ICALEPCS 2015

International Conference on Accelerator & Large Experimental Physics Control Systems

17-23 OCTOBER 2015 MELBOURNE **AUSTRALIA**

ICALEPCS 2015, hosted by the Australian Synchrotron, will bring together control systems specialists from the world's large experimental physics facilities in the vibrant city of Melbourne, Australia.

The technical program will include the latest on:

- Project Status Reports
- Systems Engineering and Project Management
- Integrating Complex or Diverse Systems
- Personnel Safety and Machine Protection
- Hardware Technology
- Timing and Sync
- Software Technology Evolution
- Experiment Control
- Process Tuning and Feedback Systems
- User Interfaces and Tools
- Data Management, Analytics, and Visualisation
- Control Systems Infrastructure
- Mobile Opportunities and Challenges

ABSTRACT SUBMISSION OPENS
19 JANUARY 2015

ABSTRACT SUBMISSION CLOSES
1 MARCH 2015

REGISTRATIONS OPEN
15 APRIL 2015

www.icalepcs2015.org

Hosted by: CSIRO | Victoria | Australian Government | ansto | Australian Synchrotron

ICALEPCS 2015

COMMITTEES

International Executive Committee

Roland Müller, HZB (Chair)
Lou Corvetti, AS
David Fernandez, ALBA
Christopher Marshall, NIF/LLNL
Ryotaro Tanaka, Spring-8

International Scientific Advisory Committee

Roland Müller, HZB (Chair)
Lou Corvetti, AS
Mark Heron, Diamond
Reinhard Bacher, DESY
Jin Dapeng, IHEP
Kuo-Tung Hsu, NSRRC
Sheng Peng, FRIB
Matthew Bickley , JLAB
Gianluca Chiozzi, ESO
David Fernandez, ALBA
Markus Janoush, PSI
Christopher Marshall, NIF/LLNL
Kazuro Furukawa, KEK
Marco Lonza, ELETTRA
Hamid Shoaee, SLAC
Kevin Brown, BNL
Philippe Gayet, CERN
John Maclean, Argonne
Young-Gi Song, KOMAC
Peg Folta, NIF/LLNL
Alain Buteau, SOLEIL
Andy Götz, ESRF
Juan Guzman, ASKAP/CSIRO
Lize Van Den Heever, Sth Africa SKA
Changbum Kim, PAL
Javier Serrano, CERN
Eric Bjorklund, LANL
Nick Hauser, ANSTO
Dennis Nicklaus, Fermilab
Karen White, ORNL/SNS
Niko Neufeld, CERN
Noboru Yamamoto, KEK
Ryotaro Tanaka, Spring-8
Jean-Michel Chaize, ESRF
Stephane Perez, CEA/LMJ

Scientific Program Committee

Markus Janousch, PSI
John McLean, Argonne
Niko Neufeld, CERN
Dennis Nicklaus, Fermilab
Karen White, ORNL
Ryotaro Tanaka, Spring-8
Jean Michel Chaize, ESRF
Stephane Perez, CEA/LMJ
Kevin Brown, BNL
Juan Carlos Guzman, ASKAP/ CSIRO
Timo Korhonen, European Spallation Source
Eric Bjorklund, LANL
Mark Heron, Diamond
Matt Bickley, JLAB
Marco Lonza, ELETTRA
Chris Marshall, NIF/LLNL
Young Gi Song, KOMAC
Andy Gotz, ESRF
Javier Serrano,CERN
Gianluca Chiozzi, ESO
Dapeng Jin, Institute of High Energy Physics
Lize van den Heever,Sth Africa SKA
Kazuro Furukawa, KEK
David Fernandez, ALBA
Allan Casey, LLNL
Gordon Brunton, LLNL
Changbum Kim, PAL
Reinhard Bacher, DESY
Noboru Yamamoto, KEK
Oscar Matilla, ALBA
Nick Hauser, ANSTO
Enrique Blanco, CERN

Local Organising Committee

Lou Corvetti (Conference chair), AS
Andrew Rhyder, AS
Kathleen Riches, AS
Maree Overall, ASN Events
Gonzalo Conesa, AS
Nick Hauser (Conference Program Chair), ANSTO
Natalia Ferreira, AS
Phil McShane, ASN Events
Annette McClellan, ASN Events
Emma Shepherd, AS

Editorial Board

Kathleen Riches, SLSA	Chief Editor
Andreas Moll, SLSA	Proceedings Editor
Marie Robichon, ESRF	Scientific Secretariat/Editor
Volker RW Schaa, JACoW/GSI	Proceedings Editor
Saji Choji, Spring 8	Proceedings Editor
Isidre Costa, CELLS-ALBA	Proceedings Editor
David Button, ANSTO	Proceedings Editor

Contents

Preface i
Committees iii
Contents v

Papers 1
MOA3O01 – SKA Telescope Manager Project Status Report 1
MOA3O02 – The Large Scale European XFEL Control System: Overview and Status of the Commissioning 5
MOB3O03 – MAX IV Laboratory, Milestones and Lessons Learned 9
MOB3O04 – The Construction Status of the SuperKEKB Control System 14
MOC3O01 – Comprehensive Fill Pattern Control Engine: Key to Top-Up Operation Quality 18
MOC3O02 – PIDTUNE: A PID Autotuning Software Tool on UNICOS CPC 22
MOC3O03 – Automatic FEL Optimization at FERMI 26
MOC3O04 – System Identification and Robust Control for the LNLS UVX Fast Orbit Feedback 30
MOC3O05 – NSLS-II Fast Orbit Feedback System 34
MOC3O06 – The Laser Megajoule Facility: The Computational System PARC 38
MOC3O07 – Low Level RF Control Implementation and Simultaneous Operation of Two FEL Undulator Beamlines at FLASH 42
MOD3I01 – Bayesian Reliability Model for Beam Permit System of RHIC at BNL 46
MOD3O02 – Continuous Delivery at SOLEIL 51
MOD3O03 – Shot Rate Improvement Strive for the National Ignition Facility (NIF) 56
MOD3O04 – Introducing the SCRUM Framework as Part of the Product Development Strategy for the ALBA Control System 60
MOD3O05 – Use of Automation in Commissioning Process of the Undulators of the European X-Ray Free Electron Laser 64
MOD3O06 – Interface Management for SKA Telescope Manager 68
MOM302 – Python Software for Measuring Wavelength at Optically Pumped Polarized Ion Source (OPPIS) 72
MOM305 – Control System for a Dedicated Accelerator for SACLA Wide-Band Beam Line 74
MOM306 – Status of the PAL-XFEL Control System 79
MOM308 – XFEL Machine Protection System (MPS) Based on uTCA 82
MOM309 – Upgrade of the Beam Monitor System for Hadron Experimental Facility at J-PARC 86
MOM310 – Nonlinear System Identification of Superconducting Magnets of RHIC at BNL 90
MOM311 – ALMA Release Management: A Practical Approach 94
MOPGF001 – Use Interrupt Driven Mode to Redesign an IOC for Digital Power Supply at SSC-LINAC 98
MOPGF002 – Magnet Corrector Power Supply Controller for LCLS-I 100
MOPGF006 – The Renovation of the CERN Controls Configuration Service 103
MOPGF008 – Embedded Environment with EPICS Support for Control Applications 107
MOPGF014 – LLRF Controls Upgrade for the LCLS XTCAV project at SLAC 110
MOPGF015 – Fast Wire Scanner Upgrade for LCLS 114
MOPGF016 – Improving the Compact Muon Solenoid Electromagnetic Calorimeter Control and Safety Systems for the Large Hadron Collider Run 2 117
MOPGF019 – Experiences and Lessons Learned in Transitioning Beamline Front-Ends from VMEbus to Modular Distributed I/O 121
MOPGF020 – Detector and Run Control Systems for the NA62 Fixed-Target Experiment at CERN 125
MOPGF021 – Database Archiving System for Supervision Systems at CERN: a Successful Upgrade Story 129
MOPGF022 – SIS18 Upgrade: The FAIR Compliant Renovation of the Data Acquisition System for Particle Detectors 133
MOPGF023 – Update of Power Supply Control System at the SAGA Light Source Storage Ring 137
MOPGF024 – Testing Framework for the LHC Beam-based Feedback System 140
MOPGF025 – Enhancing the Detector Control System of the CMS Experiment with Object Oriented Modelling 145
MOPGF026 – Laser Beam Profiling and Further Improvements to the FHI FEL 149
MOPGF027 – Real-Time EtherCAT Driver for EPICS and Embedded Linux at Paul Scherrer Institute (PSI) 153
MOPGF029 – Personnel Protection System Upgrade for the LCLS Electron Beam Linac 157
MOPGF030 – Upgrade of the Control and Interlock Systems for the Magnet Power Supplies in T2K Primary Beamline 160

MOPGF032 – Installation of a Hot-Swappable Spare Injector Laser System for the SLAC Linac Coherent Light Source . . . 163
MOPGF033 – New Developments on EPICS Drivers, Clients and Tools at SESAME . . . 167
MOPGF035 – Control System Status of SuperKEKB Injector Linac . . . 170
MOPGF036 – Control System Developments at the Electron Storage Ring DELTA . . . 173
MOPGF037 – Upgrades to Control Room Knobs at Slac National Accelerator Laboratory . . . 177
MOPGF038 – Design and Commissioning Results of MicroTCA Stripline BPM System . . . 180
MOPGF039 – TIP: An Umbrella Application for all SCADA-Based Applications for the CERN Technical Infrastructure 184
MOPGF040 – Keck Telescope Control System Upgrade . . . 188
MOPGF042 – EPICS IOC Based on Computer-On-Module for the LNL Laboratory . . . 193
MOPGF045 – MEBT and D-Plate Control System Status of the Linear IFMIF Prototype Accelerator . . . 197
MOPGF047 – Revolution Project: Progress in the Evolution of Soleil Motion Control Model . . . 201
MOPGF048 – IBEX - the New EPICS Based Instrument Control System at the ISIS Pulsed Neutron and Muon Source . . . 205
MOPGF049 – 100Hz Data Acquisition in the TANGO Control System at the Max IV Linac . . . 209
MOPGF050 – Tango-Kepler Integration at ELI-ALPS . . . 212
MOPGF051 – ELI-ALPS Control System Status Report . . . 216
MOPGF052 – A Framework for Hardware Integration in the LHCb Experiment Control System . . . 221
MOPGF056 – Synchronising High-Speed Triggered Image and Meta Data Acquisition for Beamlines . . . 225
MOPGF057 – Quick Experiment Automation Made Possible Using FPGA in LNLS . . . 229
MOPGF058 – Neutron Scattering Instrument Control System Modernization - Front-End Hardware and Software Adaption Problems . . . 233
MOPGF063 – The New TANGO-based Control and Data Acquisition of the Neutron Instrument DNS at FRM II 236
MOPGF065 – Motion Control on the Max IV Soft X-Ray Beamlines With Tango and Sardana . . . 240
MOPGF066 – Synchronized Ramping of Magnet Power Supplies for Streamlined Operation at Energy Recovery Linac (ERL) and Electron Lens (e-Lens) . . . 244
MOPGF067 – MeerKAT Control and Monitoring System Architecture . . . 247
MOPGF070 – Report on Control/DAQ Software Design and Current State of Implementation for the Percival Detector . . . 251
MOPGF071 – Sodium Laser Guide Star Emulation . . . 255
MOPGF072 – Hot Checkout for 12 GeV at Jefferson Lab . . . 258
MOPGF077 – Drift Control Engines Stabilize Top-Up Operation at BESSY II . . . 262
MOPGF079 – European XFEL Cavities Piezoelectric Tuners Control Range Optimization . . . 266
MOPGF080 – Control System of RF Stations for NICA Booster . . . 270
MOPGF087 – TPS Booster Tune Measurement System . . . 274
MOPGF088 – Integrating the Measuring System of Vibration and Beam Position Monitor to Study the Beam Stability 277
MOPGF090 – Control of Fast-Pulsed Power Converters at CERN Using a Function Generator/Controller . . . 281
MOPGF091 – White-Rabbit Based Revolution Frequency Program for the Longitudinal Beam Control of the CERN PS . . . 286
MOPGF092 – Integration of the TRACK Beam Dynamics Model to Decrease LINAC Tuning Times . . . 291
MOPGF093 – Real-time Beam Loading Compensation for Single SRF Cavity LLRF Regulation . . . 295
MOPGF097 – Architecture of Transverse Multi-Bunch Feedback Processor at Diamond . . . 298
MOPGF098 – PandA Motion Project - A Collaboration Between SOLEIL and Diamond to Upgrade Their 'Position and Acquisition' Processing Platform . . . 302
MOPGF099 – Upgraded Control System for LHC Beam-Based Collimator Alignment . . . 306
MOPGF101 – High Level Controls for the European XFEL . . . 310
MOPGF102 – The New Control Software for the CERN NA62 Beam Vacuum . . . 314
MOPGF103 – The Upgrade of Control Hardware of the CERN NA62 Beam Vacuum . . . 318
MOPGF104 – Consolidations on the Vacuum Controls of the CERN Accelerators, During the First Long Shutdown of the LHC . . . 322
MOPGF105 – Device Control Database Tool (DCDB) . . . 326
MOPGF110 – Design Strategies in the Development of the Italian Single-dish Control System . . . 330
MOPGF111 – TANGO Integration of a Specific Hardware through HTTP-server . . . 334
MOPGF112 – Measurements, Alarms and Interlocks in the Vacuum Control System of the LHC . . . 338
MOPGF113 – Controls and Interlocks for the New Elettra Super Conducting Wiggler . . . 342
MOPGF114 – Controls Interface into the Low-Level RF System in the ARIEL e-Linac at TRIUMF . . . 346

MOPGF115 – LabVIEW as a New Supervision Solution for Industrial Control Systems . . . 349
MOPGF117 – The Control System for Trim-Coil Relay-Selectors in J-PARC MR . . . 353
MOPGF119 – Design and Development of the ECR Ion Source Control System . . . 356
MOPGF120 – CAN Over Ethernet Gateways: A Convenient and Flexible Solution to Access Low Level Control Devices . . . 359
MOPGF121 – Stripping Foil Displacement Unit Control for H^- Injection in PSB at CERN . . . 363
MOPGF122 – A Fast Interlock Detection System for High-Power Switch Protection . . . 367
MOPGF123 – Upgrades of Temperature Measurements and Interlock System for the Production Target at J-PARC Hadoron Experimental Facility . . . 371
MOPGF125 – The General Interlock System (GIS) for FAIR . . . 374
MOPGF126 – A Modified Functional Safety Method for Predicting False Beam Trips and Blind Failures in the Design Phase of the ESS Beam Interlock System . . . 378
MOPGF129 – Understanding the Failure Characteristics of the Beam Permit System of RHIC at BNL . . . 382
MOPGF131 – Interlock System for Machine Protection at ThomX Accelerator . . . 386
MOPGF132 – Building an Interlock: Comparison of Technologies for Constructing Safety Interlocks . . . 389
MOPGF134 – Design of Fast Machine Protection System for the C-ADS Injection I . . . 393
MOPGF135 – Upgrade of the Trigger Synchronisation and Distribution System of the Beam Dumping System of the Large Hadron Collider . . . 397
MOPGF136 – ADaMS 3: An Enhanced Access Control System for CERN . . . 401
MOPGF137 – Interlock of Beam Loss at Low Energy Part of J-PARC Linac . . . 405
MOPGF138 – Overview and Design Status of the Fast Beam Interlock System at ESS . . . 409
MOPGF140 – Integration of PLC's in Tango Control Systems Using PyPLC . . . 413
MOPGF141 – Upgrade of Abort Trigger System for SuperKEKB . . . 417
MOPGF142 – Development of a Network-based Personal Dosimetry System, KURAMA-micro . . . 420
MOPGF143 – Integration of Heterogeneous Access Control Functionalities Using the New Generation of NI cRIO 903x Controllers . . . 424
MOPGF145 – Commissioning and Design of the Machine Protection System for Fermilab's Fast Facility . . . 428
MOPGF146 – Safety Interlock System for a Proton Linac Accelerator . . . 431
MOPGF147 – Realization of a Concept for Scheduling Parallel Beams in the Settings Management System for FAIR . . . 434
MOPGF149 – Nuclotron and NICA Control System Development Status . . . 437
MOPGF150 – Improving SOLEIL Computing Operation with a Service-Oriented Approach . . . 441
MOPGF153 – Beam Instrumentation and Data Acquisition for CRYRING@ESR . . . 446
MOPGF154 – Current Status and Perspectives of the Cryogenic Control System of EAST . . . 449
MOPGF155 – Design and Status for the Electron Lens Project at the Relativistic Heavy Ion Collider . . . 453
MOPGF158 – Sirius Control System: Design, Implementation Strategy and Measured Performance . . . 456
MOPGF160 – ARIEL Control System at TRIUMF - Status Update . . . 460
MOPGF161 – LANSCE Control System Upgrade Status and Challenges . . . 464
MOPGF162 – MaRIE - Instrumentation & Control System Design Status and Options . . . 468
MOPGF163 – Status of the Local Monitor and Control System of SKA Dishes . . . 472
MOPGF164 – Status of the EPICS-Based Control and Interlock System of the Belle II PXD . . . 476
MOPGF171 – Active Magnetic Bearings System Upgrade for LHC Cryogenic Cold Compressor, Radiations Mitigation Project (R2E) . . . 480
MOPGF172 – Bringing Quality in the Controls Software Delivery Process . . . 485
MOPGF174 – Laser-driven Hadron Therapy Project . . . 490
MOPGF175 – A Unified Approach to the Design of Orbit Feedback with Fast and Slow Correctors . . . 494
MOPGF176 – Control System Challenges from an Upgrade to the Diamond Light Source Storage Ring . . . 498
MOPGF177 – Robust Stability Analysis of Orbit Feedback Controllers . . . 502
MOPGF178 – Uncertainty Modelling of Response Matrix . . . 506
MOPGF179 – Status of the Solaris Control System - Collaborations and Technology . . . 510
TUA3O01 – Detector Controls Meets JEE on the Web . . . 513
TUA3O04 – CS-Studio Scan System Parallelization . . . 517
TUB3O01 – Advanced Workflow for Experimental Control . . . 521
TUB3O02 – Iterative Development of the Generic Continuous Scans in Sardana . . . 524
TUB3O03 – The Modular Control Concept of the Neutron Scattering Experiments at the European Spallation Source ESS . . . 529

TUB3O04 – The LMJ System Sequences Adaptability (French MegaJoule Laser) . . . 533
TUC3I01 – Machine Protection and Interlock System for Large Research Instruments . . . 537
TUC3O02 – Design, Implementation and Setup of the Fast Protection System for CSNS . . . 543
TUC3O03 – Development and Realisation of the ESS Machine Protection Concept . . . 545
TUC3O04 – Reusable Patient Safety System Framework for the Proton Therapy Centre at PSI . . . 549
TUC3O05 – NSLS-II Active Interlock System for Fast Machine Protection . . . 554
TUC3O06 – Machine Protection System for the KOMAC 100-MeV Proton Linac . . . 558
TUC3O07 – Safety Integrity Level (SIL) Verification for SLAC Radiation Safety Systems . . . 561
TUD3I01 – The LMJ Target Diagnostics Control System Architecture . . . 565
TUD3O02 – Extreme Light Infrastructure, Beamlines - Control System Architecture for the L1 Laser . . . 570
TUD3O03 – REMUS: The new CERN Radiation and Environment Monitoring Unified Supervision . . . 574
TUD3O04 – The Virtual European XFEL Accelerator . . . 578
TUD3O05 – Integrating Control Applications into Different Control Systems . . . 581
WEA3O01 – The TANGO Controls Collaboration in 2015 . . . 585
WEA3O02 – Recent Advancements and Deployments of EPICS Version 4 . . . 589
WEA3O03 – Towards Building Reusability in Control Systems - a Journey . . . 593
WEB3O01 – Open Source Contributions and Using OSGi Bundles at Diamond Light Source . . . 598
WEB3O02 – quasar - A Generic Framework for Rapid Development of OPC UA Servers . . . 602
WEB3O03 – Disruptor - Using High Performance, Low Latency Technology in the CERN Control System . . . 606
WEB3O04 – Accelerator Modelling and Message Logging with ZeroMQ . . . 610
WEB3O05 – Why Semantics Matter: a Demonstration on Knowledge-Based Control System Design . . . 615
WEC3O01 – Trigger and RF Distribution Using White Rabbit . . . 619
WEC3O02 – The Phase-Locked Loop Algorithm of the Function Generation/Controller . . . 624
WEC3O04 – New Event Timing System for Damping Ring at SuperKEKB . . . 629
WEC3O05 – Timing System for the HAPLS/L3 ELI Project . . . 633
WEC3O06 – ERL Time Management System . . . 636
WED3O01 – MASSIVE: an HPC Collaboration to Underpin Synchrotron Science . . . 640
WED3O02 – Databroker: An Interface for NSLS-II Data Management System . . . 645
WED3O03 – MADOCA II Data Logging System Using NoSQL Database for SPRING-8 . . . 648
WED3O04 – HDB++: A New Archiving System for TANGO . . . 652
WED3O05 – Big Data Analysis and Analytics with MATLAB . . . 656
WEM301 – Timing Systems for ATNF Telescopes . . . 660
WEM303 – Virtualisation within the Control System Environment at the Australian Synchrotron . . . 664
WEM304 – Status Monitoring of the EPICS Control System at the Canadian Light Source . . . 667
WEM305 – LabVIEW Interface for MADOCA II with Key-Value Stores in Messages . . . 669
WEM307 – Custom Hardware Platform Based on Intel Edison Module . . . 673
WEM308 – A Multi-Modal Human-Machine-Interface for Accelerator Operation and Maintenance Applications . 677
WEM309 – A Graphical Tool for Viewing and Interacting with a Control System . . . 681
WEM310 – How Cassandra Improves Performances and Availability of HDB++ Tango Archiving System . . . 685
WEPGF001 – The Instrument Control Electronics of the ESPRESSO Spectrograph @VLT . . . 689
WEPGF002 – A Protocol for Streaming Large Messages with UDP . . . 693
WEPGF005 – The New Modular Control System for Power Converters at CERN . . . 697
WEPGF006 – Magnet Server and Control System Database Infrastructure for the European XFEL . . . 701
WEPGF010 – Securing Access to Controls Applications with Apache httpd Proxy . . . 705
WEPGF011 – Progress of the Control Systems for the ADS injector II . . . 709
WEPGF012 – Information Security Assessment of CERN Access and Safety Systems . . . 713
WEPGF013 – Increasing Availability by Implementing Software Redundancy in the CMS Detector Control System 717
WEPGF014 – A Data Acquisition System for Abnormal RF Waveform at SACLA . . . 721
WEPGF015 – Drivers and Software for MicroTCA.4 . . . 725
WEPGF018 – Service Asset and Configuration Management in ALICE Detector Control System . . . 729
WEPGF019 – Database Applications Development of the TPS Control System . . . 732
WEPGF020 – A Redundant EPICS Control System Based on PROFINET . . . 736
WEPGF021 – Design of Control Networks for China Initiative Accelerator Driven System . . . 739
WEPGF023 – Controlling Camera and PDU . . . 743
WEPGF024 – Interfacing EPICS to the Widespread Platform Management Interface IPMI . . . 746
WEPGF025 – Data Driven Simulation Framework . . . 749

WEPGF028 – A Self-Configurable Server for Controlling Devices Over the Simple Network Management Protocol 753
WEPGF029 – High Level Software Structure for the European XFEL LLRF System 757
WEPGF030 – The EPICS Archiver Appliance 761
WEPGF031 – The Evolution of the Simulation Environment in ALMA 765
WEPGF032 – EPICS PV Management and Method for RIBF Control System 769
WEPGF034 – The Power Supply Control System of CSR 772
WEPGF036 – Data Categorization and Storage Strategies at RHIC 775
WEPGF037 – Data Lifecycle in Large Experimental Physics Facilities: The Approach of the Synchrotron ELETTRA and the Free Electron Laser FERMI 777
WEPGF038 – A Flexible System for End-User Data Visualisation, Analysis Prototyping and Experiment Logbook 782
WEPGF041 – Monitoring Mixed-Language Applications with Elastic Search, Logstash and Kibana (ELK) 786
WEPGF042 – Scalable Web Broadcasting for Historical Industrial Control Data 790
WEPGF043 – Metadatastore: A Primary Data Store for NSLS-2 Beamlines 794
WEPGF044 – Filestore: A File Management Tool for NSLS-II Beamlines 796
WEPGF045 – Large Graph Visualization of Millions of Connections in the CERN Control System Network Traffic: Analysis and Design of Routing and Firewall Rules with a New Approach 799
WEPGF046 – Towards a Second Generation Data Analysis Framework for LHC Transient Data Recording . . . 802
WEPGF047 – Smooth Migration of CERN Post Mortem Service to a Horizontally Scalable Service 806
WEPGF049 – The Unified Anka Archiving System - a Powerful Wrapper to Scada Systems like Tango and WinCC OA 810
WEPGF050 – Integrated Detector Control and Calibration Processing at the European XFEL 814
WEPGF052 – Development of the J-PARC Time-Series Data Archiver using a Distributed Database System, II 818
WEPGF053 – Monitoring and Cataloguing the Progress of Synchrotron Experiments, Data Reduction, and Data Analysis at Diamond Light Source From a User's Perspective 822
WEPGF056 – Flyscan: a Fast and Multi-technique Data Acquisition Platform for the SOLEIL Beamlines 826
WEPGF059 – The Australian Store. Synchrotron Data Management Service for Macromolecular Crystallography 830
WEPGF060 – A Data Management Infrastructure for Neutron Scattering Experiments in J-PARC/MLF 834
WEPGF061 – Beam Trail Tracking at Fermilab 838
WEPGF062 – Processing High-Bandwidth Bunch-by-Bunch Observation Data from the RF and Transverse Damper Systems of the LHC 841
WEPGF063 – Developing HDF5 for the Synchrotron Community 845
WEPGF065 – Illustrate the Flow of Monitoring Data through the MeerKAT Telescope Control Software 849
WEPGF066 – A Systematic Measurement Analyzer for LHC Operational Data 853
WEPGF068 – Formalizing Expert Knowledge in Order to Analyse CERN's Control Systems 857
WEPGF069 – Integrating Web-Based User Interface Within Cern's Industrial Control System Infrastructure . . 861
WEPGF070 – A New Data Acquiring and Query System with Oracle and EPICS in the BEPCII 865
WEPGF071 – Python Scripting for Instrument Control and Online Data Treatment 869
WEPGF072 – Parameters Tracking and Fault Diagnosis base on NoSQL Database at SSRF 873
WEPGF074 – FPGA Firmware Framework for MTCA.4 AMC Modules 876
WEPGF080 – Encoder Interface for NSLS-II Beam Line Motion Scanning Applications 881
WEPGF081 – Em# Platform: Towards a Hardware Interface Standardization Scheme 885
WEPGF083 – Single Neutron Counting Using CCD and CMOS Cameras 889
WEPGF084 – New Digitisers for Position Sensitive 3He Proportional Counters 893
WEPGF085 – The Construction of the SuperKEKB Magnet Control System 897
WEPGF089 – CERN Open Hardware Experience: Upgrading the Diamond Fast Archiver 901
WEPGF090 – Design of EPICS IOC Based on RAIN1000Z1 ZYNQ Module 905
WEPGF091 – A Formal Specification Method for PLC-based Applications 907
WEPGF092 – PLCverif: A Tool to Verify PLC Programs Based on Model Checking Techniques 911
WEPGF093 – CXv4, a Modular Control System 915
WEPGF094 – A Modular Approach to Develop Standardized HVAC Control Systems with UNICOS CPC Framework 919
WEPGF095 – Application of PyCDB for K-500 Beam Transfer Line 923
WEPGF096 – Managing a Real-time Embedded Linux Platform with Buildroot 926
WEPGF097 – Local Monitoring and Control System for the SKA Telescope Manager: A Knowledge-Based System Approach for Issues Identification Within a Logging Service 930
WEPGF100 – DRAMA 2 - An Evolutionary Leap for the DRAMA Environment for Instrumentation Software Development 934

WEPGF101 – A Modular Software Architecture for Applications that Support Accelerator Commissioning at MedAustron . . . 938
WEPGF102 – Solving the Synchronization Problem in Multi-Core Embedded Real-Time Systems . . . 942
WEPGF105 – EPICS V4 Evaluation for SNS Neutron Data . . . 947
WEPGF106 – CCLIBS: The CERN Power Converter Control Libraries . . . 950
WEPGF107 – Multi-Host Message Routing in MADOCA II . . . 954
WEPGF112 – Flop: Customizing Yocto Project for MVMExxxx PowerPC and BeagleBone ARM . . . 958
WEPGF113 – Physics Application Infrastructure Design for FRIB Driver Linac . . . 962
WEPGF115 – LabVIEW EPICS Program for Measuring BINP HLS of PAL-XFEL . . . 966
WEPGF116 – PvaPy: Python API for EPICS PV Access . . . 970
WEPGF117 – High Level Applications for HLS-II . . . 974
WEPGF118 – Use of Tornado in KAT-7 and MeerKAT Framework . . . 977
WEPGF119 – Bunch to Bucket Transfer System for FAIR . . . 980
WEPGF120 – Timing System at MAX IV - Status and Development . . . 984
WEPGF121 – Operation Status of J-PARC Timing System and Future Plan . . . 988
WEPGF122 – Real-Time Performance Improvements and Consideration of Parallel Processing for Beam Synchronous Acquisition (BSA) . . . 992
WEPGF124 – Application Using Timing System of RAON Accelerator . . . 995
WEPGF126 – Prototype of White Rabbit Network in LHAASO . . . 999
WEPGF127 – A Generic Timing Software for Fast Pulsed Magnet Systems at CERN . . . 1003
WEPGF128 – Development Status of the Sirius Timing System . . . 1007
WEPGF129 – CERN timing on PXI and cRIO platforms . . . 1011
WEPGF132 – An Update on CAFE, a C++ Channel Access Client Library, and its Scripting Language Extensions 1013
WEPGF133 – TINE Studio, Making Life Easy for Administrators, Operators and Developers . . . 1017
WEPGF134 – Applying Sophisticated Analytics to Accelerator Data at BNLs Collider-Accelerator Complex: Bridging to Repositories, Tools of Choice, and Applications . . . 1021
WEPGF135 – Using the Vaadin Web Framework for Developing Rich Accelerator Controls User Interfaces . . . 1025
WEPGF136 – Development of iBeacon Based Equipment Inventory System at STAR Experiment . . . 1029
WEPGF137 – Adopting and Adapting Control System Studio at Diamond Light Source . . . 1032
WEPGF141 – Tools and Procedures for High Quality Technical Infrastructure Monitoring Reference Data at CERN 1036
WEPGF142 – Advanced Matlab GUI Development with the DataGUI Library . . . 1040
WEPGF145 – A Structured Approach to Control System GUI Design for the Solaris Light Source . . . 1044
WEPGF146 – GUI Style Guide for Control System Applications at ESS . . . 1047
WEPGF147 – ALICE Monitoring in 3-D . . . 1049
WEPGF148 – Unifying All TANGO Control Services in a Customizable Graphical User Interface . . . 1052
WEPGF150 – A HTML5 Web Interface for JAVA DOOCS Data Display . . . 1056
WEPGF152 – Time Travel Made Possible at FERMI by the Time-Machine Application . . . 1059
WEPGF153 – Karabo-GUI: A Multi-Purpose Graphical Front-End for the Karabo Framework . . . 1063
WEPGF154 – Visualization of Interlocks with EPICS Database and EDM Embedded Windows . . . 1066
WEPGF155 – Improving Software Services Through Diagnostic and Monitoring Capabilities . . . 1070
THHA2I01 – Developing Distributed Hard-Real Time Software Systems Using FPGAs and Soft Cores . . . 1073
THHA2O02 – The LASNCE FPGA Embedded Signal Processing Framework . . . 1079
THHA2O03 – Message Signalled Interrupts in Mixed-Master Control . . . 1083
THHA3O01 – The Evolution of the ALICE Detector Control System . . . 1087
THHA3O02 – Status of the Continuous Mode Scan for Undulator Beamlines at BESSY II . . . 1091
THHA3O03 – Managing Neutron Beam Scans at the Canadian Neutron Beam Centre . . . 1096
THHB2O01 – Preliminary Design of a Real-Time Hardware Architecture for eRHIC . . . 1099
THHB2O02 – A Modular Approach to Acquisition Systems for Future CERN Beam Instrumentation Developments 1103
THHB2O03 – The Global Trigger with Online Vertex Fitting for Low Energy Neutrino Research . . . 1107
THHB3O01 – Mapping Developments at Diamond . . . 1111
THHB3O02 – Real-Time Data Reduction Integrated into Instrument Control Software . . . 1115
THHB3O03 – On-the-Fly Scans for Fast Tomography at LNLS Imaging Beamline . . . 1119
THHC2O01 – Beam Property Management at KEK Electron/Positron 7-GeV Injector Linac . . . 1123
THHC2O02 – Component Database for APS Upgrade . . . 1127
THHC2O03 – Replacing the Engine in Your Car While You Are Still Driving It . . . 1131
THHC3O01 – The MeerKAT Graphical User Interface Technology Stack . . . 1134

THHC3O03 – Effortless Creation of Control & Data Acquisition Graphical User Interfaces with Taurus 1138
THHC3O05 – National Ignition Facility (NIF) Experiment Interface Consolidation and Simplification to Support Operational User Facility 1143
THHD3O01 – Control Systems for Spallation Target in China Initiative Accelerator Driven System 1147
THHD3O05 – Standards-Based Open-Source PLC Diagnostics Monitoring 1151
THHD3O06 – Overview of the Monitoring Data Archive used on MeerKAT 1155
THHD3O08 – Upgrades to the Infrastructure and Management of the Operator Workstations and Servers for Run 2 of the CERN Accelerator Complex 1158
FRA3O01 – Past, Present and Future of the ASKAP Monitoring and Control System 1162
FRA3O02 – The Laser Magajoule Facility: Control System Status Report 1165
FRA3O03 – Overview and Status of the SwissFEL Project at PSI 1169
FRB3O01 – Commissioning of the TPS Control System 1173
FRB3O02 – Status of the European Spallation Source Control System 1177

Appendices **1183**
List of Authors 1183
Institutes List 1197

SKA TELESCOPE MANAGER – PROJECT STATUS REPORT

L.R. Brederode*, SKA SA, Cape Town, South Africa
A. Marassi, INAF-OAT, Trieste, Italy
S. Riggi, INAF-OACT, Catania, Italy

Abstract

The Square Kilometre Array (SKA) will be the world's largest radio telescope once it is complete and will use hundreds of thousands of receivers, spanning Africa and Australia to survey the sky in unprecedented detail. The SKA will be ground breaking in many respects such as image resolution, sensitivity, survey speed, data processing and size to name a few. The SKA Telescope Manager Consortium is currently designing the SKA Phase 1 (SKA1) Telescope Manager Element that will orchestrate the SKA Observatory and associated telescopes. In this paper, we report on the current status of the SKA1 Telescope Manager pre-construction project, the development process and its high-level architecture.

INTRODUCTION

Telescope Manager (TM) is one of a number of elements within the scope of the overall SKA1 pre-construction project, where the TM element has three primary responsibilities [1]:

- Management of astronomical observations;
- Management of telescope hardware and software subsystems;
- Management of data to support system operations and all stakeholders;

The TM element can thus be seen as the nerve centre largely involved in controlling and monitoring the SKA telescopes by interacting with other elements, namely:

- Dish (DISH);
- Low-Frequency Aperture Array (LFAA);
- Central Signal Processor (CSP);
- Science Data Processor (SDP);
- Assembly, Integration and Verification (AIV);
- Infrastructure Australia and South Africa (INFRA-AUS & INFRA-SA);
- Signal and Data Transport (SaDT);

The TM pre-construction project was kicked-off in November 2013 and is primarily focused on performing requirements analysis, architectural design, interface definition and prototyping activities, with the aim to produce a data pack that enables procurement contracts to be placed during 2017. Once a contract has been awarded, the development of the TM will continue with low-level design, coding and testing, integration, qualification and installation related activities ultimately culminating with a transition of the TM into a fully operational state in 2022, as per current plans.

* rbrederode@ska.ac.za

At the time of writing, the TM pre-construction project had been in progress for roughly two years since kick-off. An international consortium, led by the National Centre for Radio Astrophysics (NCRA) in India, assumes responsibility for the delivery of the TM pre-construction phase data pack.

APPROACH AND STRATEGY

The development plan for the TM is based on a tailored set of lifecycle stages and associated processes derived from the IEEE 12207-2008 and ISO/IEC TR 24748-3 standards [2]. These standards have been chosen as a basis for the development plan given that the TM Element is predominantly comprised of software, and IEEE 12207-2008 is a recognised international standard that provides a comprehensive set of lifecycle processes, activities and tasks for *software* that is part of a larger system.

The technical processes are employed in an iterative manner at various levels within the TM product hierarchy, namely the element, sub-element and assemblies \ application levels [2]. Items at each level are subjected to a requirements and design review prior to a formal system-engineering baseline being formed. The baseline indicates that there is agreement in terms of what each item in the product hierarchy is required to do, what they will consist of, and how they will interact to achieve the required functionality. A critical design review (CDR) precedes the formation of the final design baseline during the pre-construction phase. Development of the various levels proceeds largely in parallel, however it is critical that a higher level be thoroughly reviewed and baselined before a lower level can be concluded.

The TM pre-construction project aims to deliver a detailed *system* design (element & sub-elements) and a high-level *software* design (applications). The construction phase project will commence with low-level software design.

A risk management plan sees to it that risks are identified, analysed, treated and monitored on a regular basis. Prototyping activities are aimed at mitigating many of the technical risks. Exploratory prototypes are used to prove design concepts, confirm technology choices, characterise interfaces and elicit requirements. These prototypes are not developed with reuse in mind; rather they will be discarded once they have served their purpose. This reduces their cost and time to delivery. The

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

development of a system wide evolutionary prototype is outside the scope of the preconstruction phase.

WORK PACKAGES

The TM Consortium is organised into the following work packages (WPs):

- Telescope Management (TelMgt): Engineering management of the telescope;
- Observation Management (ObsMgt): Usage of the instrument for astronomical observations;
- Local Monitor and Control (LMC): Monitoring and control of the TM itself;
- Local Infrastructure (LINFRA): Computational, communications, power and facilities infrastructure for Telescope Manager;
- Prototyping (PROTO): Development of the required prototypes needed for design purposes;
- System Engineering (SE): Engineering artefacts related to requirements, architecture and interfaces;
- Project Management (MGT): Consortium coordination;

TOOLS

Various tools are used to facilitate the development of the TM, such as Alfresco© for document management, eB® Director for configuration management, JIRA© for issue tracking \ project management and Google Drive© for collaboration purposes. A Model Based Systems Engineering (MBSE) approach is employed using Systems Modelling Language (SysML) for requirements management, interface analysis and design.

PROJECT STATUS

Definition and Preliminary Design

The initial year of the TM pre-construction project focused on definition and preliminary design activities. Element level requirements were specified including functional and performance requirements, with traceability to higher-level system requirements. A top-level architecture for the TM was identified, including the next lower level assembly. Internal and external interfaces were identified and defined in the architecture. A parameterised cost model for the TM was developed, and a cost analysis report was prepared. Preliminary assembly, integration and verification (AIV) plans were prepared. A preliminary analysis with respect to hazards and safety, radio frequency interference (RFI), electromagnetic compatibility (EMC), failure-modes-effects-criticality (FMECA) was performed. Preliminary construction and maintenance plans were prepared. Draft sub-system level requirement and design specifications were prepared.

Preliminary Design Review

A preliminary design review (PDR) was conducted roughly a year into the project (Jan 2015), attended by members of the SKA Organisation (SKAO) and a panel of external experts. The primary findings of the review panel were:

- TM Consortium was found to be responsive and collaborative with a deep understanding of the key design issues;
- High quality PDR data pack had been submitted;
- Technology choices were outstanding;
- Telescope level operations concept was immature with the potential to impact the TM requirements and design;
- Scope changes as a result of construction cost constraints (referred to as 'rebaselining') needed to be considered;
- Graphical user interface (GUI) design and scripting approach was immature;

After some deliberation the SKAO indicated that a delta PDR review would be required to closeout the panel's observations.

The delta PDR review has been scheduled to take place during the 4th quarter 2015. A revised data pack addressing the primary PDR observations has been submitted to the delta PDR review panel. It is expected that the delta PDR can be concluded during 2015, with the formation of a *development* baseline.

Technology Framework Selection

The TM Consortium hosted a workshop in Trieste, Italy during March 2015 that was attended by the SKAO, representatives from all SKA element level consortia and industry experts. The primary purpose of the workshop was to work towards the selection of a monitoring and control framework technology to be implemented as an SKA wide standard. Three main candidate technologies were considered, EPICS, TANGO and Alma Common Software (ACS). Evaluation criteria included:

- Scalability and monitoring and control design concepts;
- Industrial standards and fresh module development;
- Modernity and future direction;
- User support and documentation;
- Integration and re-use of precursors;
- Risk reduction;

Although both EPICS and TANGO were deemed to fulfil the core requirements, TANGO was selected based on its product-oriented harmony versus a fragmented design in EPICS, where concerns were raised regarding the long-term support of particular modules.

The workshop concluded that TANGO was deemed to be the better choice for the SKA project with a future lifespan of several decades to consider [3].

The TM Consortium will be releasing a set of documents during 4th quarter 2015 that establish Local Monitoring and Control (LMC) interface guidelines, roles and responsibilities, and TANGO implementation

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

guidelines, endorsed by the SKA Organisation as an SKA wide standard.

Detailed Design and Prototyping

The TM Consortium officially kicked off its detailed design and prototyping activities during the course of a face-to-face meeting in June 2015. Sub-element level development plans were presented at this meeting together with progress reports in terms of closing out the PDR observations. Additional inputs were received from the SKAO, and their implications were considered. Attention was paid to risks, gaps and effort estimates.

A prototyping plan was submitted to the SKAO in July 2015 detailing the types of prototypes that will be constructed, the associated technical risks that will be mitigated, the approach, strategy and planning for each prototype. A technology baseline, detailing key technology choices, was formed and described in the prototyping plan. At the time of writing, prototyping activities had commenced with a view to producing documented results by mid 2016.

Furthermore, significant progress has been made with respect to TM sub-element development. Figure 1 illustrates aspects of a workflow that forms part of Observation Management. It denotes actors that are involved in transforming a scientific proposal through a series of workflow stages into a set of scheduling blocks (SBs) for execution. Executed SBs result in control directives being forwarded to the Telescope Management sub-element for processing.

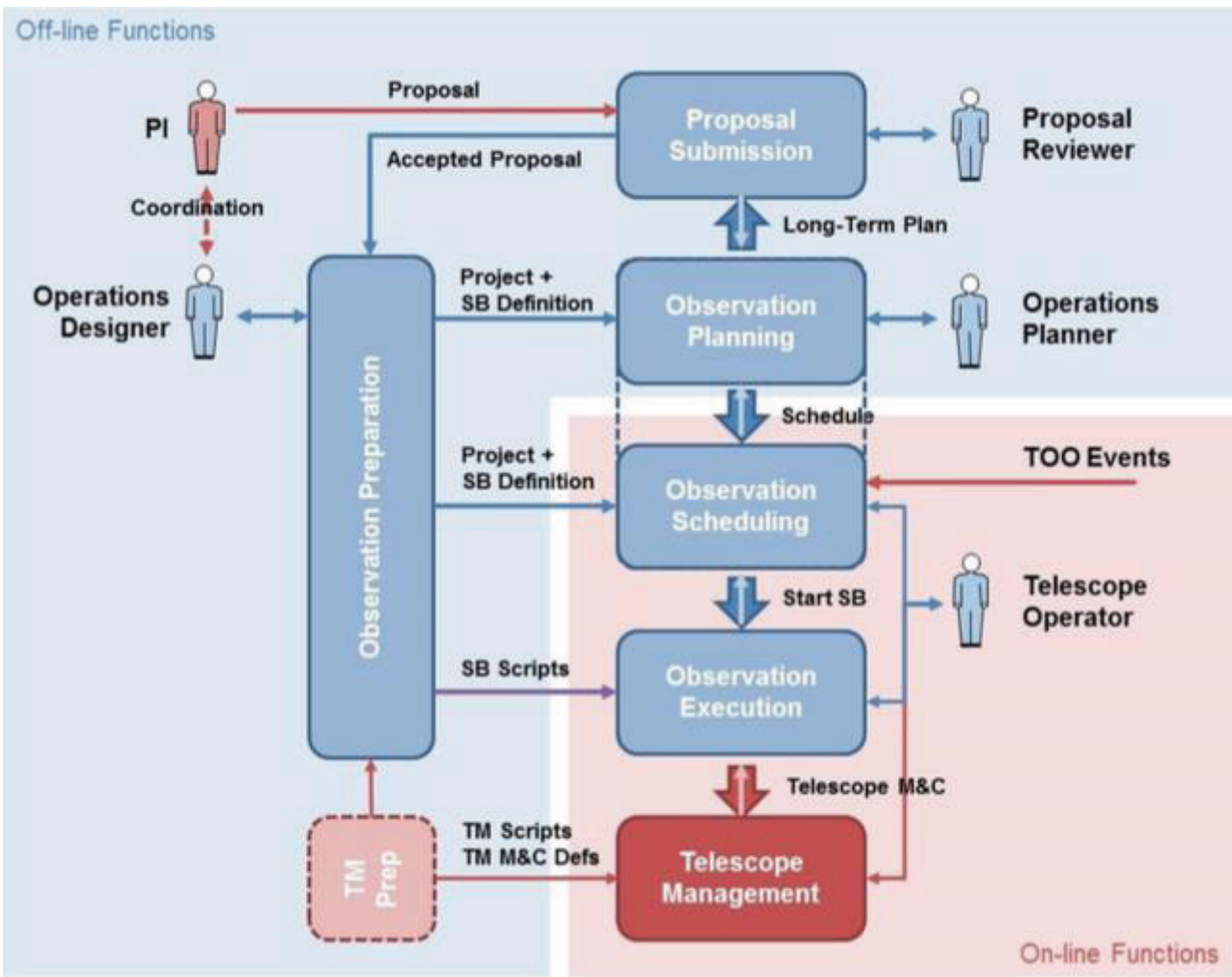


Figure 1: Observation Management High Level Functions [4].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

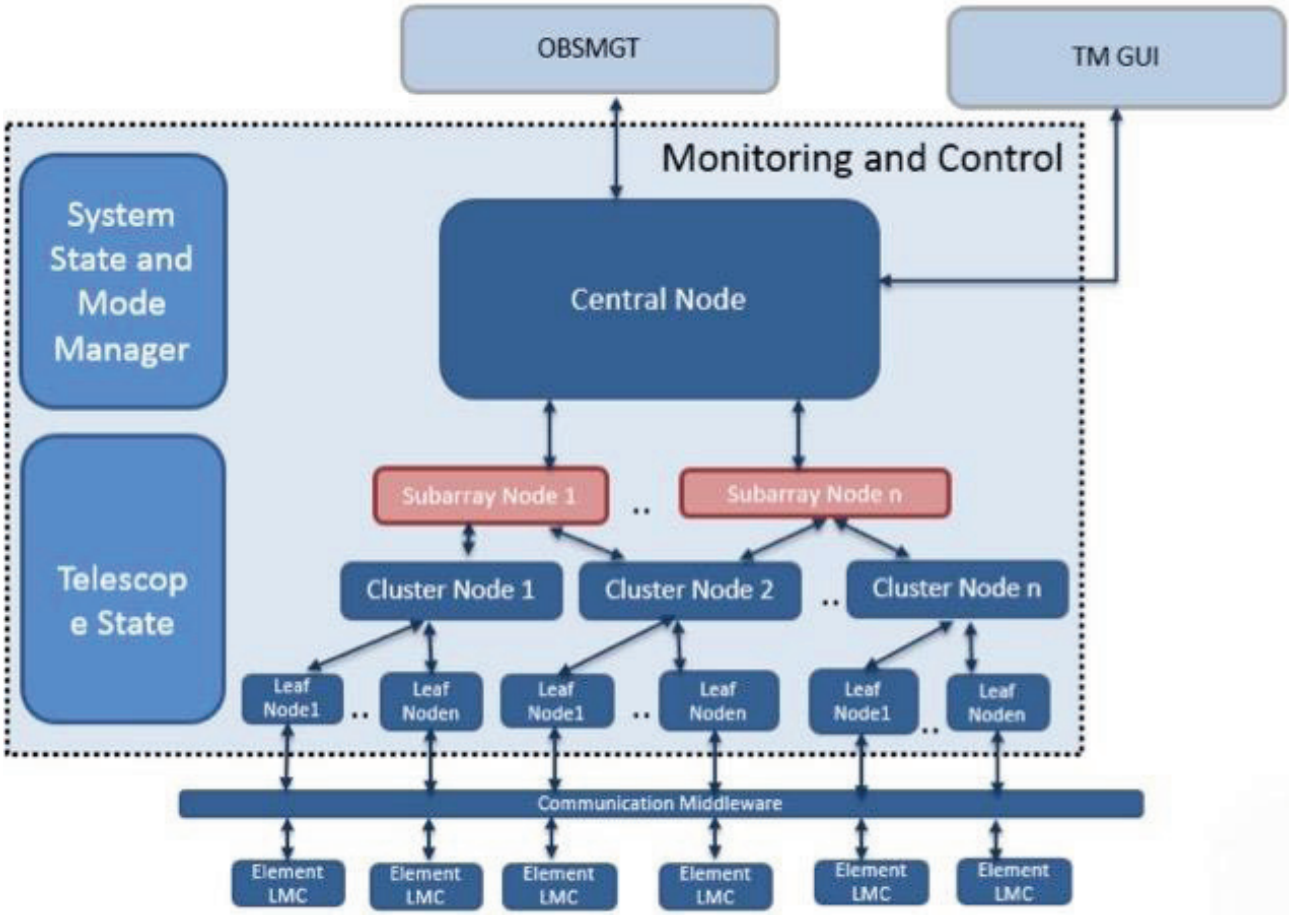


Figure 2: Monitoring & Control Interactions with Elements and Sub-elements [5].

Figure 2 depicts a hierarchy of control nodes, which form part of Telescope Management, responsible for gathering data from all Elements to obtain a real-time view of the state, health status and performance parameters of each Element, and the system as a whole. The control hierarchy enables execution of astronomical observations by configuring and controlling all Elements through commands. Interactions with other Elements are channelled via *leaf nodes* and Element LMCs.

CONCLUSION

The SKA Telescope Manager is an integral part of the SKA Observatory responsible for observation, telescope and data management functions. Significant progress has been made in developing the TM since project inception roughly two years ago. At the time of writing, a requirements and design baseline had however not yet been formed, and a schedule risk was beginning to manifest itself. Dependencies on higher-level artefacts and their level of maturity were contributing to the schedule risk. In order to mitigate the risks, the TM Consortium has adopted a strategy of making assumptions and continuing with the development activities. The next six to nine months would prove critical for the SKA1 TM project, as the assumptions would either be confirmed or shown to be false.

ACKNOWLEDGMENT

We would like to acknowledge the SKA TM Consortium partners responsible for the development of the TM, namely CSIRO (Australia), Engage SKA (Portugal), NCRA (India), INAF (Italy), NRC (Canada), STFC (UK), SKA SA (South Africa), Persistent Systems Ltd (India), SCISYS UK, and TCS (India).

REFERENCES

[1] P.S. Swart, G. le Roux, Y. Wadadekar, L. van den Heever, A. Bridger, J.C. Guzman, "SKA1 TM Requirement Specification", SKA-TEL-TM-000001 Revision F, July 2015.

[2] R. Brederode et al., "SKA1 TM Pre-Construction Plan", SKA-TEL-TM-0000045 Revision C, July 2015

[3] R. Smareglia, S. Vricic, S. Riggi, A. Faulkner, L. van den Heever et al., "LMC Standardisation Workshop Report", Revision 1, April 2015.

[4] A. Bridger et al., "SKA1 Observation Management Design Report", SKA-TEL-TM-0000017 Revision D, February 2015.

[5] S.R. Chaudhuri et al., "SKA1 TM TelMgt Design Report", SKA-TEL-TM-0000014 Revision E, July 2015.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE LARGE SCALE EUROPEAN XFEL CONTROL SYSTEM: OVERVIEW AND STATUS OF THE COMMISSIONING

A. Aghababyan, R. Bacher, P. Bartkiewicz, T. Boeckmann, T. Bruns, M. R. Clausen, T. Delfs, P. Duval, L. Froehlich, W. Gerhardt, C. Gindler, J. Hatje, O. Hensler, J. M. Jäger, R. Kammering, S. Karstensen, H. Keller, V. Kocharyan, O. Korth, A. Labudda, T. Limberg, S. Meykopff, M. Moeller, J. Penning, A. Petrosyan, G. Petrosyan, L. Petrosyan, V. Petrosyan, P. Pototzki, K. Rehlich, S. Rettig-Labusga, H. Rickens, G. Schlesselmann, B. Schoeneburg, E. Sombrowski, M. Staack, C. Stechmann, J. Szczesny, M. Walla, J. Wilgen, T. Wilksen, H.-G. Wu, DESY, Hamburg, Germany
S. Abeghyan, A. Beckmann, D. Boukhelef, N. Coppola, S. G. Esenov, B. Fernandes, P. Gessler, G. Giambartolomei, S. Hauf, B. C. Heisen, S. Karabekyan, M. Kumar, L. Maia, A. Parenti, A. Silenzi, H. Sotoudi Namin, J. Szuba, M. Teichmann, J. Tolkiehn, K. Weger, J. Wiggins, K. Wrona, M. Yakopov, C. Youngman, European XFEL GmbH, Hamburg, Germany

Abstract

The European XFEL is a 3.4 km long X-ray Free Electron Laser in the final construction and commissioning phase in Hamburg. It will produce 27000 bunches per second at 17.5 GeV. Early 2015 a first electron beam was produced in the RF-photo-injector and the commissioning of consecutive sections will follow during this and next year. The huge number and variety of devices for the accelerator, beam line, experiment, cryogenic and facility systems pose a challenging control task. Multiple systems, including industrial solutions, must be interfaced to each other. The large number of bunches requires a tight time synchronization (down to picoseconds) and high performance data acquisition systems. Fast feedbacks from front-ends, the DAQs and online analysis system with a seamless integration of controls are essential for the accelerator and the initially 6 experimental end stations. It turns out that the European XFEL will be the first installation exceeding 2500 FPGA components in the MicroTCA form factor and will run one of the largest PROFIBUS networks. Many subsystem prototypes are already successfully in operation. An overview and status of the XFEL control system will be given.

INTRODUCTION

The European XFEL is a 3.4 km long X-ray Free Electron Laser (4th-generation light source) located in Hamburg, Germany. Its linear accelerator is based on superconducting RF-technology delivering 2700 electron bunches at energies of 10.5 GeV, 14 GeV, or 17.5 GeV with a bunch train repetition rate of 10 Hz. The undulator systems of the facility will produce spatially coherent photon pulses which are less than 80 fs long showing a peak brilliance of 10^{32}–10^{34} photons/s/mm^2/mrad^2/0.1% BW in the energy range from 0.26 to 29.2 at beam energies of 10.5 GeV, 14 GeV, or 17.5 GeV. Installation work in the accelerator and beam line sections is progressing. Technical commissioning and set-up with beam will be performed in stages in 2015 and 2016. First lasing should be possible end of 2016.

The control system of the European XFEL is an effort of various distinct groups at DESY and the European XFEL GmbH. This paper describes the concepts and features of the individual solutions for accelerator, undulator, photon beam line, and cryogenic and utility controls. It addresses the challenges to be faced, discusses the interoperability between the systems and reports on the status of the whole controls project.

ACCELERATOR CONTROLS

The accelerator control system provides an environment to supervise, control, synchronize and protect the technical equipment of the accelerator. It includes among other things a large scale front-end electronics infrastructure based on the novel MTCA.4 standard [1], a timing system, a machine protection system, server software and driver libraries, communication protocols, a finite-state machine tool to facilitate process automation, and a design tool for graphical user applications. In addition, it provides a central Data Acquisition System (DAQ) aggregating and storing shot-synchronized, bunch-resolved data from distributed sources and supporting various software-based feedback systems. The environment is completed by a code repository, a repository to keep the configuration data for generating graphical user panels, various Web services, and common IT infrastructure.

Conceptual Design

Primarily the conceptual design of the accelerator control system follows the DOOCS control system ansatz in place at the FLASH facility, itself a quasi-template for the European XFEL. DOOCS-based controls cover all aspects of equipment dealing with bunch-synchronous fast data taking and algorithms. In addition, slow control such as vacuum or magnet controls is also handled by TINE-compliant control system components. TINE is the principal control system of the PETRAIII complex (3rd-generation light source). Both DOOCS and TINE are full-featured control systems and have been recently tuned to interoperate seamlessly [2].

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

System Components

More than 200 MTCA crates distributed along the accelerator host electronic plug-in modules with more than 2500 FPGAs executing tasks such as beam current read-out or low-level RF-measurements and control algorithms. Universal Linux drivers with hot-plug support are used as an interface to low-level DOOCS servers which provide board conifguration and basic read-out. Data are distributed within the crate to device-specific DOOCS servers such as the beam current server via ZeroMQ and stamped with a unique, shot-related number.

Along with management/central data switching and CPU units, an integral part of a MTCA crate is the timing-system receiver module. Synchronisation is based on a hardware timing protocol distributed through a dedicated optical fibre system. Event triggers, clock signals and bunch-related data are sent to all front-end electronic modules with accurate precision (10 ps RMS jitter). A variable bunch pattern for the individual undulator beam lines can be transmitted prior to a bunch train.

The timing system hardware is linked to the Machine Protection System (MPS). Based on well-defined error conditions the MPS can restrict or inhibit the number of bunches to be sent through a specific accelerator section (e.g. injector only). Approximately 130 MPS modules are distributed along the accelerator. Input signals are directed through a dedicated optical fibre system towards two master modules located near the injector and the dump kickers for final logical processing.

In addition to the standard control system interface a fast data-driven protocol is used to send bunch-related data to central DAQ nodes. The sustained overall data rate to the DAQ system is of the order of 1.3 GB/s. Several DAQ instances are connected to a 10 GbE backbone network and operated in parallel. The nodes synchronize the data received from the distributed DOOCS front-end servers and store the data which belong together as a common, uniquely stamped data structure in a shared memory. All data are stored on disk for a few weeks for further analysis and can optionally be saved on tape.

In order to improve application software development and to verify and tune the global system performance the so-called Virtual-XFEL test bed has been put in place. Simulated data are sent at full rate to a dedicated DAQ system which forwards the data to specific middle-layer processes running within the DAQ run time environment. These servers are dedicated to preparing data for visualization or running software loops such as the transverse or longitudinal feedback. Other servers will provide beam optics data and calculate the actual energy profile along the accelerating structures.

Operating the accelerator is based on graphical user applications. Most applications run in the so-called jddd environment which generically displays user panels based on panel-specific configuration data. In addition, rich client like applications coded in Java, MATLAB or Python will be provided by scientists for measurement purposes.

UNDULATOR CONTROLS

Three undulator systems are used to produce the photon beams. Each undulator system consists of an array of up to 35 undulator cells installed in a row along the electron beam. A single undulator cell itself consists of a 5 m long planar undulator segment and a 1.1 m long intersection housing a phase shifter, magnetic field correction coils, and a quadrupole mover. Four servo motors are used on each undulator to control the gap between girders with micrometer accuracy. The current of magnetic field correction coils as well as the gap of the phase shifter are adjustable as a function of the undulator gap [3].

System Architecture and Components

Each undulator system has a length of about 200 m and is controlled by a distributed control system. It consists of a central control node (CCN) and one local control node (LCN) per undulator cell communicating through a real-time capable (EtherCAT) network used for device and motion control and an ordinary network used for monitoring, remote access and maintenance. Both networks implement the so-called “redundant ring topology” tolerating one single point of failure. The undulator control system provides flexible and sophisticated control of the K-parameters of all involved undulators. It allows full control by the accelerator control system and control of a limited set of relevant undulator parameters (gap, taper) by the beam line users.

In operational mode the CCN receives motion commands from the accelerator control system and translates those into individual commands for each LCN. It collects status data from each LCN and sends it back to the accelerator control system. In addition, it performs various maintenance tasks (updating device configuration data and operating system software) and provides an interface to the users.

The LCN is a Programmable Logic Controller (PLC) running on an industrial PC. It controls all Beckhoff TwinCAT-compliant front-end devices belonging to each single undulator cell such as motors, beam trajectory correctors, and valve controllers. In addition, it is involved in the synchronized tuning of undulator cells to the desired K value.

A configuration database and an engineering data management system complete the undulator control system.

PHOTON BEAMLINE CONTROLS

The supervisory control and data acquisition system of the photon beam lines must integrate hundreds of distributed and very different components such as mirrors, large 2D detectors or sample-handling systems into one homogenous software framework. It interfaces to programmable logic controllers for equipment control, fast FPGA-based electronics for timing and data acquisition, CPU and GPU based algorithms for online data monitoring, distributed calibration and analysis tasks.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Conceptual Design

To cope with diverse integration needs yet still keep a homogenous interface to users and developers, a novel software framework (Karabo) [4] has been created. The remote end-points (devices) of Karabo control beam line equipment (e.g. motors, pumps, and valves) or handle concurrent data processing and analysis tasks.

Devices communicate through a message-oriented middleware using a logically central but physically clustered asynchronous and event-driven broker mechanism for message routing. Additionally, specific point-to-point connections between devices can be established and are used for shipping large and fast data as needed for DAQ and processing pipelines.

Devices are completely self-descriptive regarding their parameters and functions. Hence, once loaded (plugin technology) a device is automatically controllable through a multi-purpose graphical user interface (GUI). Parameters and commands of any device can be composed into custom expert panels by dragging and dropping. Sequencing, scanning or other high level tasks are implemented in the form of Python macros, which - within the GUI - can be edited, executed and graphically controlled in expert panels.

Programmable Logic Controller Interfaces

Real-time synchronization and equipment protection relevant hardware is controlled via TwinCAT-compliant components and PLCs. Hundreds of physical devices such as motors or temperature sensors are being integrated by device-specific field bus terminals such as digital/analog inputs or outputs, motor drivers, encoder interfaces, and serial line drivers.

Groups of functionally related terminals mounted in crates are connected via a daisy-chained EtherCAT line to a common PLC. Using a specialized TCP/IP protocol each self-descriptive hardware or IO-channel is automatically reflected in the photon beam line control system as a Karabo device instance.

Fast Electronics and DAQ Interfaces

The architecture of the beam line data acquisition system foresees five layers: detector front end electronics, front end interfaces, PC farm layer, online data cache with computing clusters, and offline data archiving with offline computing clusters.

All detectors and sensors capable of acquiring data at the pulse frequency of 4.5 MHz are provided with synchronization signals, clocks and beam related metadata (such as unique bunch train IDs) from the common accelerator timing system. Sensor signals requiring digitization are treated using FPGA based modular digitizer systems and the MTCA.4 form factor. An in-house developed firmware framework facilitates algorithm implementation of the FPGA using graphical programming tools based on Simulink [5]. In addition, register mappings and variable descriptions are exported as Karabo-readable XML files allowing instantaneous control system integration.

The large volume of data streams produced by multiple detectors (e.g. up to 10 GB/s of image data by a single 2D MPixel detector) and sensors are handled in the PC farm layer. Efficient handling of this huge data volume requires large network bandwidth (10 GbE switched links) and properly structured and tuned computing capabilities (PC layer, storage systems, and analysis clusters) interconnected by an Infiniband fabric. The PC layer nodes are highly optimized Karabo devices which periodically receive bursts of image datagrams, which need to be correlated with other fast and slow data. Data are integrated based on bunch train ID and forwarded to the storage system and to the online computing farm for further analysis. All PC layer devices are synchronized and orchestrated via the same finite state machine using a single-instance Run Controller device.

CRYOGENIC AND UTILITY CONTROLS

Control systems for cryogenic equipment are similar to process control systems for refineries or chemical plants. This is the domain of PLCs as deterministic calculations and control loops are here mandatory. PLCs are also partly used for the cryogenic system but only in those cases where machine protection or special turnkey system requirements apply. In general all process control functionalities reside in EPICS IOCs. The majority of the control functions (about 80%) consists of control loops and digital logic. The remaining 20% is implemented in complex state notation language programs used for sequential operations, state based operations as well as supervisory controls.In contrast, utility controls rely nearly 100% on PLC controls for their turnkey systems.

All parts of the cryogenic infrastructure are controlled by the same control system based on EPICS. Cryogenic and utility controls share the same control infrastructure. While the EPICS IOCs basically run basically core software with some custom extensions, the operators work at consoles running Control System Studio (CSS), which provides plugins for synoptic displays, trending, alarming, etc.

Continuous Operation and Reliability

Cryogenic systems typically run in a 24/7 mode with run periods of one year and more. This defines basic requirements for the control equipment. Sensors and actors as well as the process controllers must fulfill these requirements. Redundant sensors are in place wherever they are buried inside cryogenic boxes. Redundant process controllers are in place for every critical component. In addition redundant power supplies and networks are used to be prepared for component failures. This philosophy is in place throughout the whole control system.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The I/O System

Cryogenic systems are widely distributed. Even signals forming logical units may be spread over several hundred meters. I/O components are thus integrated into the system by means of Profibus field buses [6]. Profibus comes in two flavors: Profibus DP und Profibus PA where the latter also provides the power over the wires to the equipment. Individual Profibus nodes may consist of intelligent controllers gathering several tens of different signals as well as intelligent sensors and actors which in turn are connected to the PA variant of Profibus. The number of Profibus nodes adds up to 540 and results in 12.700 EPICS records.

Diagnostic Tools and Archiving

Widely distributed systems require good diagnostic tools with remote access to all basic information to diagnose error states and failures. In this case all Profibus lines are connected to individual diagnostic gateways which provide diagnostic information based on html (web) pages. In addition to this online information many diagnostic data are collected in the archive system for post mortem analysis.

INTEROPERABILITY

Basically all control systems used to operate the European XFEL are able to exchange a limited set of information between each other through a common Ethernet network. However, remaining issues related to naming conventions or authorization concepts might still require attention. In addition timing relevant data such as events or unique bunch train IDs can be distributed by the common accelerator timing system.

In order to facilitate device control and beam operation, both systems (DOOCS, TINE) used for accelerator control are tightly interconnected at both the client and server level. On the one hand the DOOCS client API can access any TINE server. On the other hand DOOCS servers can be configured to become defacto TINE servers. However, native TINE servers do not include DOOCS-specific DAQ system software, limiting the application of such servers to specific use cases.

The undulator control system integrates seamlessly into DOOCS. Each undulator system provides its own DOOCS-like server. Both experiment users and accelerator operators use DOOCS clients to control the undulators.

To facilitate the inter-communication with accelerator controls both Karabo and DOOCS clients will embed the others client API. In addition, Karabo must also integrate various in-kind contributions from external control systems (such as high-energy lasers, split and delay lines, etc.).

The interoperability between the cryogenic and the accelerator control system is still being discussed. Established solutions exist to exchange data between EPICS and DOOCS as well as TINE.

PROJECT STATUS

The installation of the general and in particular the novel MTCA infrastructure is progressing. Most aspects of the front-end hardware and software are in successful, routine operation at FLASH. Application development for the accelerator controls is still on-going. New software is either being tested or put into service at FLASH and the Virtual-XFEL simulation environment or directly in the gradual commissioning of already completed accelerator sections. So far the required performances have been demonstrated.

The basic concept of the undulator control system has been successfully tested at a mock-up. Extending the concept to a real undulator system and implementing the final controls software is under way.

Karabo is close to being fully featured. As of this writing it already controls a large pump-probe laser setup, several test stands (involving task like vacuum, motion, detector calibration and alignment, etc.) and will be in production for controlling the photon beamline components currently being installed in the SASE1 tunnels.

The cryogenic plant is successfully in operation for the past half year. Along with the installation of the accelerating structures the cryogenic equipment of the distribution system will be installed and tested. According to the staged technical commissioning of the accelerator sections, the cryogenic and utility control system will be put into service.

REFERENCES

[1] H. Schlarb et al., „The Case of MTCA.4: Managing the Introduction of a New Crate Standard at Large Scale Facilites and Beyond“, ICALEPCS’13, San Francisco, USA, October 2013, FMOPPC081, p. 285 (2013); http://www.JACoW.org

[2] P. Duval et al., Control System Interoperability, an Extreme Case: Merging DOOCS and TINE”, PCaPAC’12, Kolkata, India, December 2012, THIB04, p. 115; http://www.JACoW.org

[3] S. Karabekyan et al., “The Undulator Control System for the European XFEL”, IPAC’12, New Orleans, USA, May 2012, THPPR002, p 3966 (2012); http://www.JACoW.org

[4] B.C. Heisen et al., Karabo: An Integrated Software Framework Combining Control, Data Management, and Scientific Computing Tasks“, ICALEPCS’13, San Francisco, USA, October 2013, FRCOAAB02, p. 1465 (2013); http://www.JACoW.org

[5] B. Fernandes et al., High Level FPGA Programming Framework Based on Simulink”, ICALEPCS’13, San Francisco, USA, October 2013, TUPPC086, p. 776 (2013); http://www.JACoW.org

[6] M. Clausen et al.,"PROFIBUS in Process Controls", PCaPAC’14, Karlsruhe, Germany, October 2014, WCO2013, p. 16; http://www.JACoW.org

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MAX IV LABORATORY, MILESTONES AND LESSONS LEARNED

V. Hardion, Y. Cerenius, F. Hennies, K. Larsson, J. Lidon-Simon, M. Sjöström, D.P. Spruce, MAX IV Laboratory, Lund, Sweden

Abstract

The MAX IV Laboratory is a new scientific research facility based on synchrotron light being built at Lund University, southern Sweden. The accelerator consists of one full energy linear accelerator providing two storage rings at 1.5 GeV & 3 GeV and a Short Pulse Facility. Additionally, more than 13 beamlines are planned to be built among which should be operational for the first users in 2016. The current status and approach of the control system is presented from its technical and organisational point of view, including the stakeholders, as well as the lessons learned from the commissioning as part of our continuous improvement for the future.

STATUS

Since the last ICALEPCS 2013 [1], the status of MAX IV Laboratory has moved toward the official operation milestones, the 21srt of June 2016 when the inauguration will take place. In the mean time the current MAX-lab (MAX I, II, III) will be decommissioned at the end of 2015.

MAX IV Commissioning

The Linac commissioning started in 2014Q2 and for the next 6 months it was focused primary on the 3 GeV storage ring injection. This milestone was reached in February 2014 despite the occurrence of several incident including the replacement of the Thermionic Gun.

The 3 GeV commissioning has just started in August 2015 after 10 months of equipment installation and Sub System Test (SST) but, already, several intermediate milestones have been passed:

- 14th August, 2015: Beam injected
- 26th August, 2015: Beam stored
- 8th October, 2015: Beam stacked at 4mA

The 1.5 GeV installation is going to be installed from the end of 2015 but already SOLARIS[28] which follow the same design has proven the concept, the 1st of October, by achieving 20mA at 1.5 GeV.

Beamlines

In the mean time, SPECIES, a Max II beamline and, FEMTOMAX, a Short Pulse Facility beamline received their first photon with a standard MAX IV control system beginning of 2014. Both are being used as test bench for the standard control system that will be used in the rest of the MAX IV beamlines [2]. BIOMAX, NANOMAX, BALDER, VERITAS, HIPPIE, all 5 3GeV beamlines, have successfully passed the site acceptance tests for their Front Ends and part of their optics.

KONTROLL & IT SUPPORT

The “Kontroll & IT Support” (KITS) group is responsible for providing the development and the support of the standard IT and electronics equipment, spread along the accelerators, the beamlines and also the administration of MAX IV. The group is part of the Science division and divided into several teams organised by business fields, responsible to organise the resource the most efficiently possible while keeping included into different projects as well as the other group of MAX IV, a so-called Hard Matrix organisation.

The KITS project management is based on Lean[3] where the focus is to bring the user value incrementally and as soon as possible without technical overhead. The Control System (CS) has been delivered following the Kanban[4] and the Scrum[5] methods since 2012[1], while adapting continuously the rules to follow the pace and the nature of a synchrotron project.

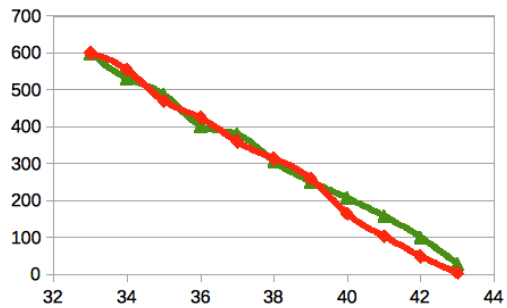


Figure 1: 3 GeV installation Burn down chart between sprint 32 and 44. Sprint estimation in red and done in green, in arbitrary unit

Network and System

The networks are based on TCP/IP v4 with a physical separation between Machine, Beamlines and offices. Each of them is divided by a set of virtual networks, usually corresponding to a subsystem, to minimise the influence of broadcast equipment. The layer 2 is based on Ethernet with 2 levels of switches in order to have a maximum network distance of 4.

All the accelerator and beamlines servers & IOC are virtual machines running by a VMWare hypervisor EsXi cluster of a redundant 36 cores and 384 GB memory. The standard OS is CentOS 6[6] as many providers support the RedHat drivers. All the Tango servers run on virtual machines apart for some specific case where a local control is needed. The definition of this infrastructure profits of the advantage that many equipment are controlled via Ethernet.

Ansible [7] is used for the configuration management of all the CS computers from the OS settings to the

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

specific role (workstation, database or IOC) as well as the software versioning.

A Continuous Delivery process has been set up with the help of the RPM system, Jenkins a continuous integration server and Ansible. A dedicated MAX IV software toolkit, DSCONFIG, is used to reconfigure the Tango system and to track any changes in version with GIT.

Hardware

The standard motion equipment ICEPAP[8] is provided by the KITS group. Around hundreds of axes have been installed to mainly control the phase shifters, some diagnostic equipment like screens, lenses, scrapers and the diagnostic beamlines. All the beamline optics are steered through this controller except one axis. The EPU and InVacuum undulators will also be steered via ICEPAP.

The first standard Input Output Controller (IOC) have been installed in the SPECIES beamline to accurately read the encoder positions. The IOC is based on Adlink IPC, ADC, DAC and National Instrument for the counter card. This equipment is intended to be used as a General Purpose Input Output acquisition system on the experimental station of the beamlines

The IT group provides also the entire timing system[9] for the facility, based on Micro Research Finland (MRF). A special development of the Libera Brilliance Plus timing card, done by ITech, allows to use them as a compatible event receiver. The last status and development are described in another article[10].

The beamline control system will benefit to the standard equipment. The number of XBPM per beamline is estimated up to 20 per beamline, controlled by a new Electrometer[11] developed in collaboration with Alba, based on the first version deployed at Alba. The development consists of assembling 4 of the current amplifiers in the same chassis, controlled and synchronised through a backplane steered by an FPGA. This will allow synchronised measurement and the possibility to embed custom algorithm for a control loop to steer, for example, a piezo fine tuning based on current (Licence Open Hardware[12] and Open Source). In location where the signal/noise ratio is expected to allow it, a multiplexer will be used to reduce the number of amplifiers. This multiplexer is currently in development at MAX IV.

User Software

The control room workstation runs the client applications from several PC with a maximum of 8 screens each. The CS is driven with 39 GUI and monitored by 3 main web applications, the archiving & alarms database front ends and the maintenance scripts executor based on Jenkins. Additionally the operators have access to a logging system, ELOG[13] from PSI, to keep track of the commissioning progress and the changes on the system.

Among the applications several piece of software have been developed by the operators themselves[14]. Their GUI allows to monitor valves, rf structure, PSS, magnets resistances... These tools have been mainly develop with Tango JDraw[15], Taurus designer and Taurus gui[16] or Igor with the Tango binding.

The GUI proposed by KITS are mainly based on the Taurus framework that allow a rapid creation of the application without development (TaurusGUI). By this way several dedicated subsystem GUI have been developed for the commissioning of the Accelerator subsystems. The composition and the layout of the panels can be easily changed by the operators.

In the overall the initial specification of the accelerator control system allowed the reuse of widgets, panels and applications coming from the ALBA Synchrotron. The best example is the VACCA[17] application for the monitoring of the 3 GeV storage ring vacuum subsystem and also the beamlines.

Some requirement could not be fulfilled with the standard framework. This is the case of the Synoptic application[18], showing the logical representation of the system like a map and the State Grid application, showing the states of the control system by a reduced Subsystem by Location matrix. These applications make use of the web technologies.

Middleware

On the back end side the Tango control system version 8[19] is used as a middleware. For each type of equipment KITS develops the Tango device, software process to establish the communication with the hardware. In some case the drivers software have been contracted within a framework agreement with the manufacturers (Scandinova RF modulator[20], Libera Single Pass e and Brilliance Plus from ITech[21]) who have developed a compatible Tango server.

Additionally MAX IV received a software API from different collaborators and manufacturers. Budker Institut have provided a complete suite of software to control and set up their pulsed power supply via a web gateway[18] and Solaris has developed the Tango Server integration. Danfysik[22] and Itest[23] have provided a C++ libraries for their respective power supplies. In total 90 different classes of devices, mainly written in python, compose the catalogue of supported hardware and services for the MAXIV CS. For example on the accelerator the control system represents:

- 106 hosts
- 1187 server instances
- 7921 devices
- 287378 control points
- 132833 attributes
- 83735 commands

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

High Level Software

Parts of the control system can now be steered in the units of interest to the user, by putting a higher layer of intelligence on top of the minimal hardware control. For example, the magnets of the accelerator can be steered by magnetic field, via software on top of the power supply devices that interpolate the current-field calibration data provided by the manufacturer. The operators use Matlab high level scripts to set the required fields. The physical magnets are connected such that one power supply may control several magnets in series. An intermediate Tango device, the magnet-circuit, uses current-field calibration data that are an average of all the magnets on that circuit. This allows the operators to set an average field (converted to a single current set point on the power supply) but each magnet device in the circuit reports its own fields based on its individual calibration. Additionally, some magnets have a trim coils with independent power supplies that can be configured in different modes selected through a switch board (corrector, quadrupole, sextupole). The magnet Tango device must report the state of all its dependencies; the main and trim coil power supplies, the switch board device, temperature interlocks. The total field is also reported as a function of the main and trim circuits. For the operator the circuit device propose more features like the cycling and a redundant software interlock by monitoring the coil resistance.

Scientific Software

Matlab and the Matlab Middle Layer (MML) [24] have been the primary choice to run the scientific software on the accelerator thankful to the Tango Matlab binding[25]. The operators can easily program scripts and applications while reducing the lead time between requirement and operation. The MML is a collection of application to monitor and steer a Synchrotron. This software has been develop for the control of SLAC and extended by several facilities world-wide.

Data Management

The user of MAX IV will access a High Performance Computing (HPC) to run the off line analysis on their data through a remote CentOS desktop solution, ThinLinc[27], running the application in seamless mode from a native OS node (no emulation). A high performance storage system GPFS/InfiniBand will be installed next year.

The current user data policy is provided for giving the ownership to the users and for keeping the data during the next following 3 months after the experimentation. The HDF5 format and Nexus when applicable has been defined as the standard format for the data storage, and the control system component like Sardana[31] or the Fast Data Archiver[26] have proved their compatibility and performance.

All the IT systems at MAX IV will enforce a security at 2 factors authentication, One Time Password foresee. The ability to connect through different ID providers like the Umbrella project is also part of the expectation. In this context MAX IV leverages the collaboration with the partners like the Lund University[28] and especially with LUNARC [29].

The Digital User Office system (PSI version adapted to our need) has been successfully tested at the current MAXLAB facility. Some experimentation has been done for the data catalogue with ISpyB[30] on one MX beamline. There is a plan to extend this application to a more general Laboratory Information Management System (LIMS) i.e to others fields than MX.

LESSONS LEARNED

Sharing Technology

A large effort was done by the beamline workgroups to use identical equipment where possible, saving by consequence the resource in development. The same strategy was applied for the storage rings where more than 80% of the equipments are reused from the Linac.

The CS team had to integrate a majority of “off the shelf” equipments. Although the level of integration was simply defined by having an Ethernet connection and an API, the choice the software API was longly discuss during the procurement with the supplier. This has paid off as several of the company sold their equipment along with Tango device and by consequence bringing Tango in the industrial scene.

Finally MAX IV benefits a lot of the experience and collaborations with other facilities. The reused software, like Sardana, reduce the pressure on the resources.

Agile and Lean

Lean give definitively the guideline for managing the MAX IV project by:

- keeping focus on the priority
- working on what is known
- bringing the user value as soon as possible and get from them a feedback on the progress
- avoiding spending resource on unused programs

The team was able to adapt quickly the organisation from SCRUM with 2 weeks iteration for the development phase to KANBAN during the 3 GeV commissioning phase to face the flow of operational issues as fast as possible. The Just In Time leads us to some trouble during the Linac installation because of the late delivery of the equipment. The simulation was not an option as it doesn't brings any value for the user representing only a perfect system. We learned at the opposite that the automatic test help to deal with this situation by securing the known part, the user interface.

The Agile method helped to handle quickly the features discovered during the testing phase. Together with the product owner the focus of the SST was changed

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

regarding the estimation of the sprint planning meeting to secure more the low level of the CS than the GUI. The daily standup meeting help to improve the communication within the team as well as the sprint planning meeting with the product owner, stakeholders.

Organisation: Conway's Law

The architecture of the system is the reflect of the company organisation. Since the MAC 2014Q2 the organisation of the project has changed to improve the communication channels with all the subsystems. Each one of the control team member is the contact person to one or several other subsystem like magnet, vacuum, PSS... A product owner, as defined by Scrum, has been settled to coordinate the overall requirement and make the priority between the hundreds of request, in synchronisation with the Scrum master. This organisation helped to be more focused and efficient on the most valuable features for the users.

In 2014Q4, a similar approach was defined for the beamlines projects. But in 2015Q1 a workshop organised by the beamline staff and the CS team revealed that the organisation should follow the technical structure instead of the company organisation. The constraint of resource would benefit on the following up of the common beamline components (monochromator, mirrors, front end, undulator...) whereas the management per beamline would cover the specific beamline requirements.

Test

The CS has make profit of the current MAX-lab Laboratory to test the system and from the operators to report an early feedback. But the Linac was a new installation involving the entire organisation. Many new issues were discovered during the installation and the start of the commissioning that the focus stayed on the basic control. For example the H/V correctors got inverted along with the polarity of the power supply or a lack of the robustness that keeping loosing the communication.

For the 3 GeV a large part of the project was spent on the subsystems tests. The goal was to deliver the system ready for the commissioning with the fewest experienced defects. The SST helped to discover the discrepancy in the connection chain. For example the steering and the protection system of every single magnet has been all tested from the power supply to the report of the correct field of the magnet on the synoptic GUI.

The configuration was indeed much more complex as it cover not only the hardware level of control but also the GUI. So, decided early in the 3 GeV planning, all subsystem had to include the CS in their test suites. The tests included the access to the functionalities but also the performance and the quality of the responses. The improvement in the configuration management allowed a very fast reconfiguration without regression for all the different subsystem.

The immediate results noticed by the operators is that they never experienced the same problem than the linac at the same phase. Only with the new 3GeV specific equipment that could not be fully tested causes trouble during the commissioning. The time and resource constraints to test more 'on the bench' the BPM as the accelerator could not be run without beam before to start the commissioning. Many false-positive alarms consider as counter productive for the commissioning were quickly disabled thanks to the high coverage of the configuration management.

Ethernet Based

MAX IV choose to participate to the development of a new Libera Brilliance Plus with the risk to use the commissioning time. But the new model simplify the integration to the timing system. The chosen architecture bring the advantage to be IOC free by relying on the Ethernet network for the primary CS. MAXIV learned a lot about industrial Ethernet integration. Even the PSS rely on the Ethernet switch it means more pressure on the reliability.

User Experience

The User eXperience (UX) has to be adapted to the culture and the context. Tango give more structure and standard compared to a home made CS. The CS representation looks like very complex but, thankful to the naming convention, very easy to navigate and to integrate new equipment. However the new comers may find difficult to deal with the control system because its complexity is still high. The heavy use of Matlab and command line, compared to the previous accelerator, give more flexibility for the experimental part of the commissioning. The user reported that the applications like the State Grid could have been helpful during the SST and especially for the Linac.

A progress has to be done on the integration of the recovery of the accelerator state. Hopefully more sequence and automation will progressively tend to push the functionalities to the GUI to have a better and simpler interface for the operators. On the other hand the Linac did not use too much scripts and leads to the development of the Synoptic GUI. The SPECIES beamline, as well, where a specific acquisition GUI has been develop to simplify the interact of their Users.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ACKNOWLEDGEMENT

The authors would like to thank the whole of the Controls and IT Group (KITs) and the Accelerator and Beamline Scientists, at the Max IV Laboratory, without the collaboration of which this work would not have been possible as well as:

- ESRF for TANGO, ICEPAP, MXCUBE
- ALBA with all the python software Sardana and Taurus, icepap, electrometer
- SOLARIS for sharing the experience and devices
- SOLEIL for the pulsed magnet, the nanoprobe, the wiggler, MxCube Web
- ESS, ELI, DESY by sharing the experience.

REFERENCES

[1] J. Lidon-Simon et al., "Status of the MAX IV Laboratory Control System", MOPPC109, ICALEPCS'13, San Francisco, USA, (2013).
[2] M. Linberg et al., "Motion Control on the Max IV Soft X-Ray Beamlines With Tango and Sardana", MOPGF065, ICALEPCS'15, Melbourne, Australia, (2015).
[3] The Lean Enterprise Institute, website: http://www.lean.org/WhatsLean/.
[4] Kanban from Wikipedia, website: https://en.wikipedia.org/wiki/Kanban
[5] Scrum, the agile software development framework, website: http://www.scrum.org
[6] CentOS, the Open Source Linux distribution, website: https://www.centos.org/.
[7] V. Hardion et al., "Configuration Management of the control system", THPPC013, ICALEPCS'13, San Francisco, USA, (2013).
[8] The ICEPAP motion controller from ESRF, website: http://www.esrf.eu/Instrumentation/DetectorsAndElectronics/icepap
[9] J. Jamroz et al, "Timing System at MAX IV", THPPC103, ICALEPCS'13, San Francisco, USA, (2013).
[10] J. Jamroz et al, "Timing System at MAX IV – Status and Development", WEPGF120, ICALEPCS'15, Melbourne, Australia, (2015).
[11] O. Matilla et al, "Em# Platform: Towards a Hardware Interface Standardization Scheme", WEPGF081, ICALEPCS'15, Melbourne, Australia, (2015).
[12] The CERN Open Hardware Licence from Wikipedia, https://en.wikipedia.org/wiki/CERN_Open_Hardware_License
[13] The ELOG Home Page, https://midas.psi.ch/elog/.
[14] V. Hardion et al., "Manage the MAX IV Laboratory Control System as an Open Source Project", MOPPC086, ICALEPCS'13, San Francisco, USA, (2013).
[15] A. Gotz et al., "The TANGO Controls Collaboration in 2015", WEA3O01, ICALEPCS'15, Melbourne, Australia, (2015).
[16] Sardana control user environment, http://www.tangocontrols.org/Documents/tools/Sardana
[17] S. Rubio-Manrique et al., "Unifying All TANGO Control Services in a Customizable Graphical User Interface", WEPGF148, ICALEPCS'15, Melbourne, Australia, (2015).
[18] J. Forsberg et al., "A Graphical Tool for Viewing and Interacting with a Control System", WEM309, ICALEPCS'15, Melbourne, Australia, (2015).
[19] Scandinova, website: http://www.scandinovasystems.com/.
[20] I-Tech Libera Brilliance Plus, website: http://www.i-tech.si/accelerators-instrumentation/libera-brilliance-plus
[21] A. Panov et al., "TANGO Integration of a Specific Hardware through HTTP-server", MOPGF111, ICALEPCS'15, Melbourne, Australia, (2015).
[22] Danfysik, website: http://www.danfysik.com
[23] Itest, website: http://www.bilt-system.com/.
[24] G. Portmann et al., "An Accelerator Control Middle Layer Using MATLAB", FPAT077, ICALEPCS'05, Geneva, Switzerland, (2005).
[25] L.S.N. Nadolski et al., "High Level Control Applications for SOLEIL Commissioning and Operation", ROPA005, ICALEPCS'05, Geneva, Switzerland, (2005).
[26] P. Bell et al., "100Hz Data Acquisition in the TANGO Control System at the Max IV Linac", MOPGF049, ICALEPCS'15, Melbourne, Australia, (2015).
[27] ThinLic, website:, https://www.cendio.com/thinlinc/what-is-thinlinc
[28] Lund Universitet, website: http://www.lunduniversity.lu.se/
[29] LUNARC website, http://www.lunarc.lu.se/.
[30] ISpyB, website: http://www.esrf.eu/UsersAndScience/Experiments/MX/How_to_use_our_beamlines/ISPYB
[31] P.P. Goryl et al., "Status of the Solaris Control System - Collaborations and Technology", MOPGF151, ICALEPCS'15, Melbourne, Australia, (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE CONSTRUCTION STATUS OF THE SuperKEKB CONTROL SYSTEM

M. Iwasaki#, A. Akiyama, K. Furukawa, H. Kaji, T. Naito, T. T. Nakamura, J. Odagiri, S. Sasaki,
KEK, Ibaraki, Japan
Y. Iitsuka, N. Yoshifuji, EJIT, Hitachi, Ibaraki, Japan
K. Asano, M. Hirose, KIS, Ibaraki, Japan
T. Aoyama, M. Fujita, T. Nakamura, K. Yoshii
Mitsubishi Electric System & Service Co. Ltd, Japan

Abstract

We have designed and upgraded the accelerator control system for SuperKEKB, the next generation B-factory experiment in Japan. The SuperKEKB control system is based on the features of the KEKB control system, while additional technologies have been implemented. In this paper, we report the construction status of the SuperKEKB accelerator control system.

INTRODUCTION

SuperKEKB is the upgrade of the KEKB asymmetric energy electron-positron collider for the next generation B-factory experiment in Japan [1]. The designed luminosity is to achieve a 40-times higher luminosity than the world record by KEKB. The KEKB operation finished in 2010 June to start the SuperKEKB accelerator construction. Currently the construction is in the final stage to prepare for the 1st operation in 2016 Feb. Here the accelerator operation will be performed in three stages, Phase 1, 2, and 3. The Phase 1 is for the vacuum scrubbing and basic machine tuning, and is the operation without the final focusing magnet system (QCS) nor the BelleII detector. After the Phase 1 operation, there will be Phase 2 and 3 operations with QCS and BelleII without (Phase 2) or with (Phase 3) the vertex detector.

The KEKB control system was based on EPICS (Experimental Physics and Industrial Control System) [2] at the equipment layer and scripting languages at the operation layer. The SuperKEKB control system continues to employ those features, while we implement additional technologies for the successful operation at such a high luminosity.

We have developed the interface modules to control thousands magnet power supplies, and introduce the EPICS embedded PLC, where EPICS runs on a CPU module. In the timing system, the new configuration for positron beams is required. The faster response abort trigger system has been developed. For the stable status monitoring, we have introduced the new alarm system based on CSS (Control System Studio) [3]. For Phase 2 and beyond, we have developed the new data archiving system based on CSS. The new VME-based FPGA module is also developed.

This paper describes the design and status of the SuperKEKB accelerator control system.

#masako.iwasaki@kek.jp

HARDWARE INTERFACE FOR SuperKEKB

For KEKB and SuperKEKB, EPICS is used as the main software to control the accelerator at the equipment layer. The EPICS framework consists of the Operator Interfaces (OPIs) and the Input/Output Controllers (IOCs). In KEKB, our system was constructed with the EPICS base version 3.13. For SuperKEKB, we're going to update to the version 3.14, however there still remain IOCs which are based on the EPICS base v3.13. For the operation layer, several script language of SAD script, python are used.

In KEKB, VME single board computers with VxWorks are mainly used as IOCs. For SuperKEKB, as well as the VME/VxWorks, the PLC (programmable logic controller) with a CPU module where Linux is running, and the Linux PC are also used as IOCs.

Figure 1 shows the picture of the PLC modules for the beam mask controller. In the CPU module (Yokogawa F3RP61), Linux is running, and we install EPICS into the CPU module to control the PLC. For SuperKEKB, the F3RP61 module is used to control the many devices of, for example, the vacuum system, LLRF, the automatic calibration subsystem of the large-type-magnet power supply, and beam collimators [4].

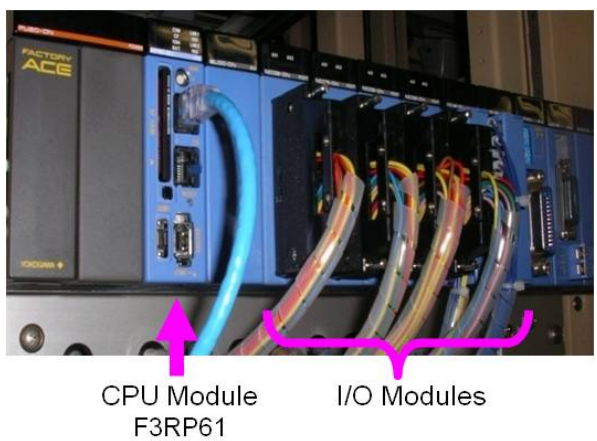


Figure 1: Picture of the PLC modules for the beam collimator controller.

There are many kinds of fieldbus in SuperKEKB accelerator components, such as Ethernet, GP-IB, serial, VXI/MXI for BPM, ARCNET for magnet power supply, and CAMAC. For the magnet-power-supply control, we have developed the PSICM (Power Supply Interface

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Controller Module) for KEKB. In SuperKEKB, significant number power supplies have been newly installed, and we have upgraded the PSICM [5], which is fully backword compatible to the previous PSICM.

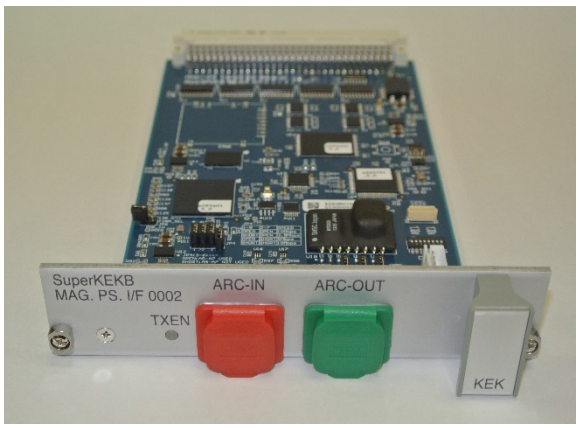


Figure 2: The upgraded PSICM (Power Supply Interface Controller Module), used for the magnet power supply control, for SuperKEKB.

Figure 2 shows the picture of the upgraded PSICM for SuperKEKB. The new PSICM has features of the faster data transfer rate of 10Mbps or 5Mbps in addition to the 2.5Mbps of KEKB, 32-bit data handling to support the 24, 20, or 18-bit resolution DAC (the previous PSICM supports 16-bit only), and redundant timing-signal inputs. For the Phase 1 operation, we use both old and new PSICM due to the budget limitation, and 426 new PSICM (out of the 2162 magnet power supplies in LER and HER) have been installed.

DATA ARCHIVING SYSTEM

In KEKB, we have used KEKBLog as a data archiving system. In SuperKEKB, we continue to use the KEKBLog which is the file based logging system, as a primary data archiving system. In addition, the new data archiving system based on the CSS [3] Archiver is also developed. CSS is originally developed at DESY, and has many general software components for the device control. CSS-based Archiver is one of the CSS tools, which collects data using the EPICS Channel Access protocol, and stores the data to the backend relational database system. Here we use PostgreSQL for the database system.

The database can be directly accessed with CSS. We have also developed the data browser program which directly accesses the database, based on the ROOT [6]. ROOT is the software framework based on C++ developed at CERN, and is used by many experiments in high-energy physics, astrophysics, etc. User's PC with CSS or the ROOT based browser can remotely access to the database for the real-time, historical, or trend data monitoring.

Currently, the CSS Archiver with the PostgreSQL backend database is running on our system. It collects the vacuum system data and the QCS cryogenic data, and works without problem so far.

NEW ALARM SYSTEM

In KEKB, we used SAD based alarm system. For SuperKEKB, we have constructed the new alarm system based on the CSS BEAST alarm handler software [7]. The alarm system monitors the severity of EPICS records, and updates the database which shows the alarm status of the monitored EPICS records. The system also uses the other database for the alarm history logging. We use PostgreSQL for both databases. Figure 3 shows the schematic view of the alarm system.

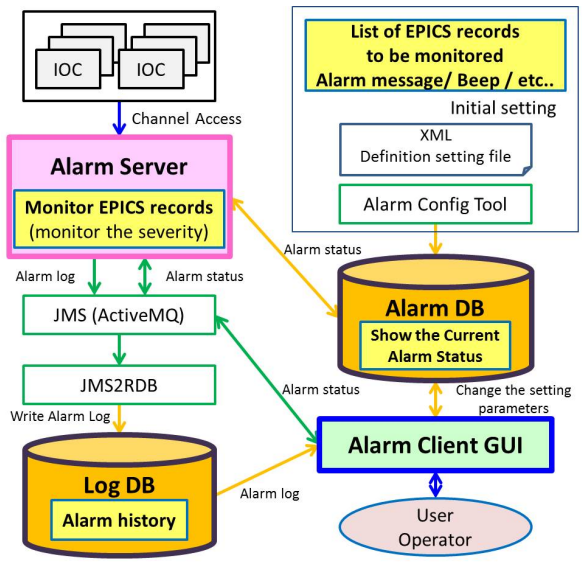


Figure 3: Schematic view of the new alarm system based on the CSS BEAST alarm handler software.

Though the CSS alarm system is already used in many accelerators, the evaluation test is important to ensure the new alarm system can stably operate under the SuperKEKB environment of several 10 thousands alarm points (~25,000 in KEKB). We did load tests using dummy records, and confirm the CSS alarm system works well under such huge number of monitoring points. We have also prepared the user interface software tools based on Python for the alarm system. Currently, the CSS alarm system is running on our system, and monitors the vacuum system data without any problem so far.

TIMING SYSTEM FOR SuperKEKB

In SuperKEKB, we construct the positron Damping Ring (DR) to produce the lower emittance beams. The KEKB timing system is based on a frequency divider/multiply and a digital delay technique. Since the KEKB ring RF (508.887MHz) is not a divisor of the Linac RF (2856 MHz), both of the KEKB ring and the Linac frequencies are locked with a common divisor frequency (10.385 MHz = 96ns cycle), which determines the injection timing. For the SuperKEKB positron injection, the injection timing of 96ns cycle for KEKB

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

becomes 11.34 ms, because of the properties of DR, if we use the similar configuration as KEKB.

Instead of this, we have developed the new timing system, which generates the injection timing signals [8]. Figure 4 shows the layout of the event timing system located at the Linac main trigger station for the SuperKEKB positron injection. The system consists with two layers of event generators (EVGs). As EVG detects

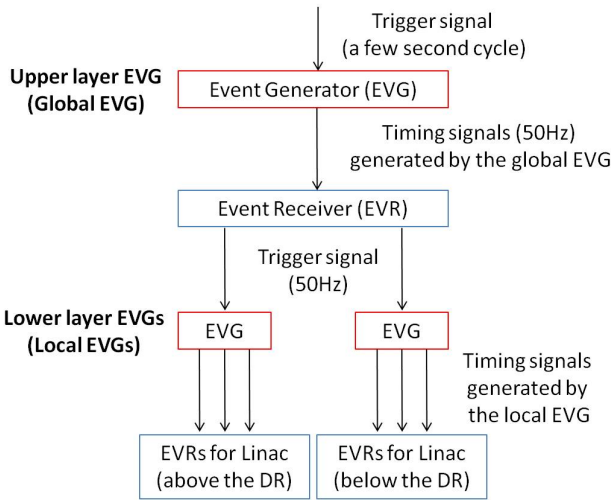


Figure 4: Layout of the event timing system at the Linac main trigger station for SuperKEKB positron injections.

the external trigger, EVG generates and sends the timing signal to the event receivers (EVRs). The trigger for the upper layer EVG (global EVG) is a coincidence between 50Hz and 11.34 ms cycle, which corresponds to a few second cycle. The global EVG sends the 50Hz timing signal for EVR. The EVR outputs 50Hz trigger signals to the lower layer EVGs (local EVGs). The local EVG for the Linac below the DR generates the timing signals, in taking account for the proper positron injections.

For SuperKEKB, we have developed the new Event modules, which are utilized to deliver triggers to the DR beam monitors. Figure 5 is a picture of the new Event Generator, VME-EVO, having 2 inputs and 8 outputs. The details of the VME-EVO module as well as the new Event Receiver VME-EVE, are described in Ref. [8].The triggers for 84 DR beam position monitors will be managed with 5 new Event modules.

Figure 5: The new Event Generator, VME-EVO.

ABORT TRIGGER SYSTEM

We have developed the faster response abort trigger system for SuperKEKB [9]. Figure 6 shows the layout of the abort trigger system.

There are over 130 points which monitor the beam status and issue the abort signal. The abort signal is E/O converted, and received by the VME abort modules located at the local control rooms. 20 local abort modules are connected to the abort module in the SuperKEKB control room, which gathers the local abort module signals, takes OR of them, and issues the trigger signal to the abort kicker.

In KEKB, these modules are connected with wires, and low pass filters which caused time delay of ~100μs were necessary for the noise reduction. For SuperKEKB, we have adopted the E/O conversion, replaced the wires with the optical cables to transfer the signal, and removed the low pass filters from the system. The total abort trigger system response time is improved to 20 μs.

Based on the feasibility tests with a prototype module, the new module design has been improved and fixed. The new system has been partially installed and has worked with the previous system.

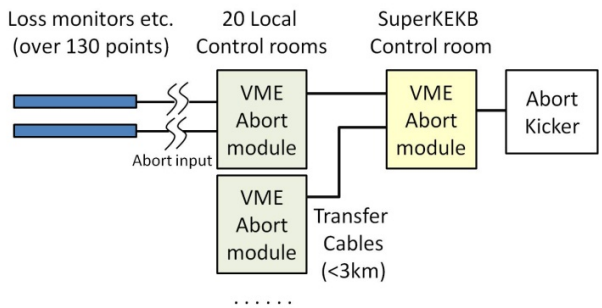


Figure 6: Layout of the SuperKEKB abort trigger system.

NEW SIGNAL TRANSFER SCHEME WITH FPGA BOARD

We have developed the new signal transfer scheme for the communication between SuperKEKB and the BelleII detector, e.g. injection control. In KEKB, we transfer E/O converted signals via ~2km length optical cables for the accelerator and detector communication. In SuperKEKB, the number of signals to be transferred increases, while the above signal transfer scheme requires the same number optical cables as the signals to be transferred.

To save the optical cable, we have developed the new signal transfer scheme using FPGA, based on digital sampling, parallel to serial, and serial to parallel conversions, shown in Figure 7. We use SFP(+) for the optical signal interface. The number of the optical cables in this system is reduced to two, for incoming and outgoing signals, as shown in Figure 8. Since SFP has the signal transfer bandwidth of 1 Gbps (~5 Gbps for SFP+), the digital sampling rate of higher than MHz, which is the higher rate than the beam revolution frequency of 100 kHz, is possible in this scheme.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

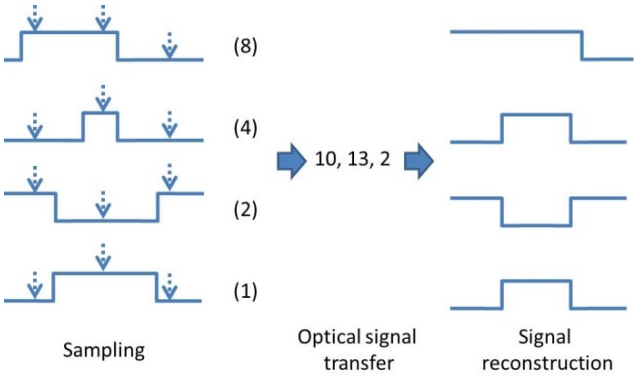


Figure 7: Injection-control signal transfer scheme for SuperKEKB.

In this scheme, we use a general purpose VME module with an FMC (FPGA Mezzanine Card) interface which is developed for Spring-8 [10], with a TTL-NIM-IO mezzanine card for NIM level and LVTTL signal IOs which is developed for SuperKEKB [11].

We also apply the combination of the VME module with a TTL-NIM-IO mezzanine card for several systems, the beam gate open/close request signal transfer between SuperKEKB and injector Linac, the software abort request system, the quench detection system for QCS, etc. The first processing of the VME module with an FMC and the TTL-NIM-IO mezzanine card was done in 2014-2015, and several evaluation tests are carried out [12].

Both the VME module and the TTL-NIM-IO mezzanine card are developed as a project of Open-It (Open Source Consortium of Instrumentation [13]).

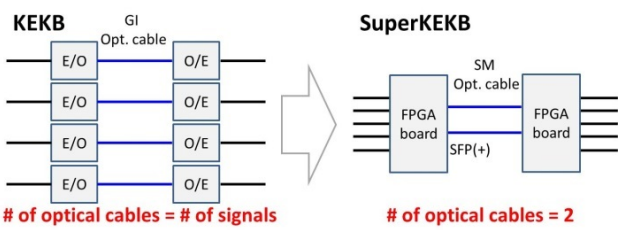


Figure 8: Injection-control signal transfer methods for KEKB and SuperKEKB.

RENOVATION OF THE CONTROL AND COMPUTING ROOMS

We did renovation of the control and the computing rooms in 2013-2014 for SuperKEKB. We removed old panel-board cabinets and server racks from the computing room. We installed new server racks and new power supply modules which are located in the new server racks. We also installed the new consoles into the SuperKEKB control room.

SUMMARY

We have improved the accelerator control system for SuperKEKB operations under the 40-times higher luminosity than the world record achieved by KEKB. Based on the KEKB accelerator control system, we have implemented the new features of the improved control hardware and software tools and interfaces, the new data archiving system, the new alarm system, the new timing system, the faster response abort trigger system, and the new signal transfer scheme based on FPGA. We have also renovated the computing and main control rooms and have been preparing for the SuperKEKB Phase 1 operation in 2016 Feb.

ACKNOWLEDGMENT

We thank T. Abe (Spring-8), M. Ikeno and M. Shoji (KEK) on the VME-FPGA module development, for their technical supports as a project of Open-It.

REFERENCES

[1] SuperKEKB Task Force, Letter of Intent for KEK Super B Factory, KEK-REPROT-2004-4, Jun. 2004. Belle II Collaboration, KEK-REPORT-2010-1, Nov. 2010.
[2] EPICS, http://www.aps.anl.gov/epics/
[3] CSS, http://www.aps.anl.gov/epics/eclipse/
[4] J. Odagiri et al., "Integration of PLC with EPICS IOC for SuperKEKB Control System", ICALEPCS2013, San Francisco, CA, USA, Oct. 2013, p. 31 (2013).
[5] T.T. Nakamura et al., "The Construction of the SuperKEKB Magnet Control System", presented at ICALEPCS2015, Melbourne, Australia, paper WEPGF085, *these proceedings*.
[6] ROOT, https://root.cern.ch/
[7] K. Kasemir, X. Chen, E. Danilova, "The Best Ever Alarm System Toolkit", ICALEPCS2009, Kobe, Japan, Oct. 2009, p. 46 (2009).
[8] H. Kaji et al., "New Event Timing System for Damping Ring at SuperKEKB", presented at ICALEPCS2015, Melbourne, Australia, paper WEC3O04, *these proceedings*.
[9] S. Sasaki et al., "Upgrade of Abort Trigger System for SuperKEKB", presented at ICALEPCS2015, Melbourne, Australia, paper MOPGF141, *these proceedings*.
[10] T. Abe et al., "Developement and Application of VME module with FMC Interface (1) - Design of the VME module and application to Spring-8 -", 12th annual Meeting of Particle Accelerator Society of Japan, Fukui, Japan, Aug. 2015, p.1299 (2015).
[11] M. Iwasaki et al., "Developement and Application of VME module with FMC Interface (2) - Application to SuperKEKB -", *ibid.*, p.1286.
[12] S. Sasaki et al., "Developement and Application of VME module with FMC Interface (3) - Evaluation tests for SuperKEKB -", *ibid.*, p.1266.
[13] Open-It, http://openit.kek.jp

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

COMPREHENSIVE FILL PATTERN CONTROL ENGINE: KEY TO TOP-UP OPERATION QUALITY*

T. Birke†, F. Falkenstern, R. Müller, A. Schälicke
Helmholtz-Zentrum Berlin für Materialien und Energie, Berlin, Germany

Abstract

At the light source BESSY II numerous experiments as well as machine development studies benefit from a very flexible and stable fill pattern. The fill pattern control engine is based on a state machine that controls the full chain from gun timing, linac pulse trains, injection and extraction elements as well as next shot predictions allowing triggering the next user experiment DAQ cycle. Architecture and interplay of the software components as well as implemented functionality with respect to hardware control, performance surveillance, reasoning of next actions and radiation safety requirements are described.

INTRODUCTION

After starting user operation in 1998, the 3rd generation light source BESSY II provided users with synchrotron radiation in decaying beam mode with three injections per day for 14 years. During this time BESSY II offered specific support for time-resolved experiments, pioneered low-α mode with coherent THz-radiation and ps-pulses and established the most advanced femtosecond slicing [1] endstation with 100 fs pulses. A purity controlled high current camshaft bunch in the ion clearing gap enables pump/probe experiments as well as pseudo singlebunch experiments at full intensity with a mechanical chopper.

Switching to top-up mode [2] extended these possibilities with a thermally stabilized machine and higher average intensity. Today, more pseudo single bunch experiments with reduced intensity are possible by using pulse picking by resonant excitation (PPRE) [3] with a selected bunch close to the end of the gap.

All these specific bunches have to be topped up with minimal variation to programmable intensities.

CONSTRAINTS

Radiation Safety

Based on analysis of facility properties and various malfunction scenarios, several restrictions and modifications have been set up to guarantee minimized losses and to make all sources of possible damages measurable and hence allow beamshutters to be kept open during injections:

- Injection efficiency must be >60% for every shot.
- Average injection efficiency must be >90% over 4 h-blocks.
- Maximum allowed injection frequency is 0.1 Hz.
- Maximum increase in stored current per injection shot: 2 mA.
- Minimum current in booster. For reliable and accurate efficiency measurement, a minimum current of 0.3 mA during an injection is mandatory.
- Minimum and maximum current of stored beam together with a corresponding minimum lifetime. To control and limit the normal average loss, a limit of max. 60 mA/h has been defined. The nominal limits are 200 mA-300 mA stored beam with a minimum lifetime of 5 h.

Top-Up Interlocks

Two separate systems are responsible for efficiency measurement and interlock. During top-up operation, both have to permanently check all relevant parameters for compliance with the top-up radiation protection restrictions and approve top-up operation to continue. The efficiency interlock systems are part of the radiation protection system.

Any violation of any of the restrictions will activate the top-up interlock system and pause or terminate top-up operation while beamshutters are free to open. Top-up injections are continued as soon as all top-up conditions are restored. This may happen automatically (e.g lifetime temporarily below limit) or may have to be accomplished by operations staff after closing beamshutters (e.g. last injection efficiency below 60%).

The fill pattern control engine operates in both top-up user-runs (beamshutters unlocked and free to open) and commissioning or machine development runs (beamshutters closed and locked). The latter is possible even without switching interlock systems to top-up mode. So the strict limitations of top-up mode are not enforced in this case and the top-up interlocks are inactive.

Injection Frequency and Timing

A proper compromise between minimizing variation in bunch currents and minimizing the number of disruptions due to injection kicker pulses had to be settled. The currently consolidated injection-scheme is, to fill a maximum of 5 bunches (reduced to e.g. 1 or 3 when filling dedicated single- or slicing-bunches) at every multibunch injection shot with the highest possible bunch currents. The decay phase between two injections currently ranges from 10 s to 200 s with an average of 120 s and every injection shot adds between 0.3 mA to 1.8 mA to the stored beam (ring current variation during top-up operation: ~0.5%).

* Work funded by BMBF and Land Berlin
† thomas.birke@helmholtz-berlin.de

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TOP-UP SERVICE AND FILLPATTERN CONTROL ENGINE

These core automation center for top-up operation at BESSY II is the "Top-Up Service and Fillpattern Control Engine" that has been developed during tests, setup and commissioning of top-up operation mode.

It is implemented as a standard EPICS state-machine together with a corresponding realtime database hosted on a dedicated Software I/O-Controller (SoftIOC).

During runtime, this system communicates via EPICS ChannelAccess (CA) with several other I/O-controllers (IOCs).

Input Signals

State and processed data of top-up interlock systems During top-up operation, the status of all top-up interlock systems is monitored to decide whether to continue top-up injections or to provide information about the reason of a top-up interruption. If efficiency decreases, a possible injection-free time at the beginning of the next 4h-block is indicated.

If top-up mode is not active and therefore the beamshutters are locked, it is not necessary to obey the top-up restrictions stringently. All this information has to be processed and displayed and may initiate state transitions.

Fillpattern Monitor To decide which positions and bunch combinations to fill on the next injection shot, the fill pattern monitor is the central source of information and core diagnostic components for proper top-up operation.

It is implemented using a stripline based high-performance ADC that delivers per bunch current measurement with high accuracy (up to 100 nA resolution) [4]. Delivering an accurate fill pattern measurement of bunchcurrents of all 400 possible bunches every second, it builds the foundation to decide where and how to place the next injection shot.

The fill pattern monitor needed to be reclassified when preparing of BESSY II for top-up operation. In the transition from an optional diagnostic element to an essential core system of top-up operation, it has been duplicated with a hot spare system and can be complemented with high-resolution photon counter based on an avalanche diode that produces an accurate measurement of the fill pattern every 15 s – 30 s.

Global beam current and lifetime measurement Used to determine whether and when an injection shot is expected to be initiated.

Other inputs like the state of extraction- and injection-pulsers as well as the overall injector status from linac to booster synchrotron are monitored and evaluated to decide whether top-up injections can be carried on or not.

Controlled Elements

After an injection, the injector is switched into suspend mode by switching off the extraction trigger pulse for the linac-gun as well as the trigger for the booster-injection-pulsers. This measure is taken to avoid activation of the septum as well as to protect the electron gun and the pulser klystrons from premature aging, because the complete injector is operated with a 1 Hz repetition rate.

When preparing for an injection, the injector is resumed, and the linac is set up to produce the determined number of pulses with the corresponding time interval in between. Currently, the number of pulses varies between 1 and 5 while the interval is typically 12 ns. This satisfies the requirements of the currently used fill pattern (induced by the timing granularity of slicing laser systems), but may change anytime.

When all preconditions are met and an injection shot is imminent, the timing system is configured to place the injected bunches at the selected position in the storage ring and the global injection trigger is switched on.

Program Structure

The main program consists of three independent parts:

The finite state machine implements the core engine that controls and reflects the state of the top-up system and the storage ring. It consists of five main states and some few transitional states (see Figure 1).

Combining all incoming status information from interlocks, measurements and configuration setup and performing the appropriate state transitions, the state machine provides full automation of the injection process for filling the machine from scratch as well as running BESSY II in top-up operation mode.

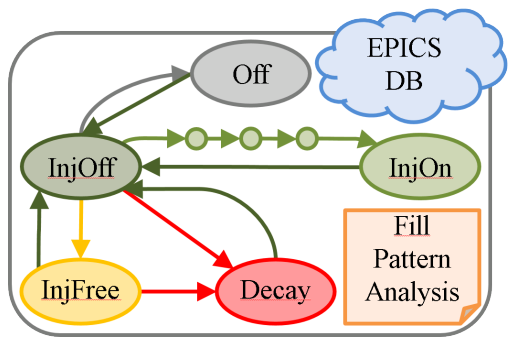


Figure 1: Simplified state diagram showing only main states and companion modules.

The fill pattern analysis asynchronously processes the measured fill pattern and decides which bunch group would be topped up with the next injection shot regardless of when it would actually happen.

This is a process that is triggered every second on incoming fill pattern measurement data. According to the current planned fill pattern, all possible positions and bunch groups to aim an injection shot at are checked, and the combination that lacks the most total current relative to the desired fill pattern is chosen.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The EPICS realtime database is the sole interface to control the state machine as well as the fill pattern analysis. It also reflects the internal status of the fill pattern control engine and holds many informational parameters for operations staff, analysis and statistics.

The advantage of using an EPICS realtime database as the sole control- and monitoring-interface to a background server process is, that all parameters, whether they are configuration or information, are provided as standard EPICS process variables (PVs) and hence integrate perfectly into the underlying control system. Other applications may access these PVs just like any other PV in the control system. They are archived and alarm-monitored and can be monitored and controlled from any control system console as well as displayed on user-information monitors and in web-based status displays.

In terms of control system infrastructure, the Top-Up Service and Fillpattern Control Engine behaves just like any other device in the BESSY II control system.

Human Machine Interface

The verbose user interface panel (see Figure 2) provides access to all configurable parameters of the control engine as well as all relevant diagnostic and interlock information and several parameters internal to the control software.

Target current and positions of all currently supported bunch-groups are configured through this interface and define the desired planned fill pattern.

A simple event/message log provides a quick overview of overall top-up operation events, and a reduced set of informational PVs like the result of the fill pattern analysis is displayed in this panel. The full set of internal status information is accessible in a separate more verbose panel.

Links to a few auxiliary software modules like injector standby/resume, linac controls and RF-knockout gap control are also provided for easy access.

The fill pattern diagram shows the current live fill pattern as well as the underlying desired fill pattern calculated from the configuring parameters. It also displays the position and scaled up shape of the latest increase in the measured fill pattern, which is the last seen injection shot. Optionally the range cleared by RF-knockout during every shot for 50 ms, to maintain a high purity of the camshaft bunch, is also displayed.

The top-up service determines some further diagnostic information that are not of immediate relevance for the top-up interlock systems, but might point to subsequent problems if not taken care of. An example is the per bunch booster current, that prevents injections on camshaft- or PPRE-bunch if it does not exceed the minimum limit.

Fillpattern Control Engine — Configuration

Development over the last three years lead to the current scheme of defining the desired fill pattern. The overall total current and four groups of bunches can be configured to match the current needs:

Multibunch fill The part of the ring that will be populated with equally filled bunches is defined by exclusion. The length and position of the ion-clearing gap are configured and hence define the multibunch fill, which can in turn be setup to have every single or every n-th bunch filled.

Camshaft bunch A single bunch at a fixed position in the purity-controlled gap. Only the current stored in this bunch can be configured. This bunch is used for standard pump/probe experiments, but also for single-bunch experiments at full rate and intensity using the phase locked MHz pulse selector [5], a chopper wheel synchronized to the circulating beam opening a 70 ns window for the light pulse at 1.25 MHz.

PPRE bunch For a different scheme of pseudo single bunch experiments, a dedicated single bunch in configurable distance, but close to the end of the gap, is filled.

Slicing bunches For femtosecond experiments, a group of three bunches, usually opposite of the camshaft bunch, are filled with higher currents than the standard multibunch bunches. These are to be sliced one at a time with a fs-laser at a 6 kHz rate.

All these four groups or any subset may be configured, defining the desired fill pattern for the next user-run.

Error Handling

Problems that may occur during top-up mode are divided into five classes:

Top-up interlock Causes pausing or termination of top-up operation (includes full or partial beamloss).

Positioning mismatch A shot that is not appearing at the position it was targeted at may be caused by a timing failure, or a diagnostic problem. No further reliable exact positioning is possible until resolved. Immediate fallback to round-robin positioning of injection shots.

Injector failure Resulting in insufficient booster current will suspend injections for some or all bunch groups.

Depending on the actual available booster current, the fill pattern control engine may leave out any bunch-groups that could not be filled without violating the minimum booster current limit for a proper injection efficiency measurement.

I^2-limit exceeded Impedance related power deposit in components scales with the sum of the squared bunch currents. To protect sensitive components, an appropriate limit is defined.

Persistent efficiency problems Since top-up interlocks are inactive during beamscrubbing, the fill pattern control engine itself will suspend injections to avoid activation of components and damages.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

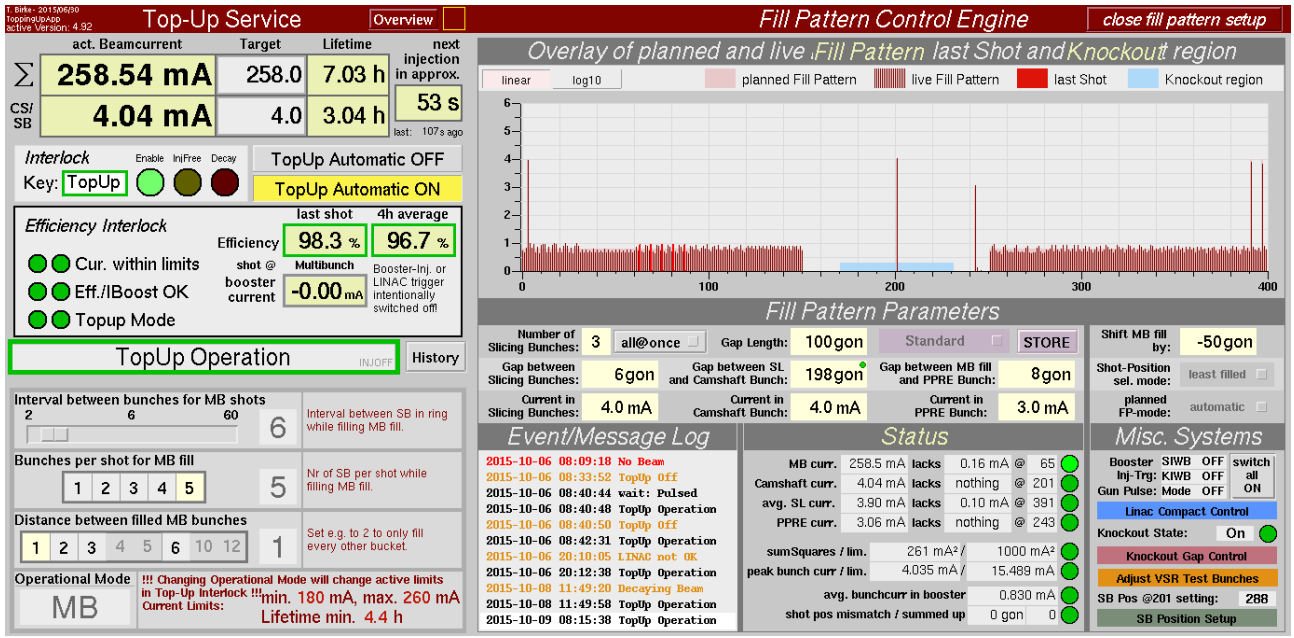


Figure 2: Top-Up Service and Fillpattern Control panel in standard multibunch hybrid mode. It contains a multibunch train, a purity controlled high current camshaft bunch in an ion clearing gap for pump/probe experiments and a mechanical pulse picking chopper, three high current bunches for femtosecond slicing and a specific bunch at the end of the gap for PPRE.

All these error conditions are handled properly by either suspending injections until the error-condition disappears or falling back into decay mode until problems are fixed manually by operations staff.

Reliable countdown

Sensitive experiments require a reliable prediction of start and end of the decay-phases between two injections.

In top-up operation mode, after each shot an estimation is made on when the next injection shot will be necessary depending on actual current and lifetime of the stored beam. Users can rely on this countdown, so even if beam-lifetime is decreasing e.g. due to insertion devices moving at low gaps, no injection is triggered before the reliable countdown has elapsed.

Fillpattern Control Engine – Next Steps

An update to the fill pattern control engine is currently in development, that has a more general approach. It will enable configuration of an arbitrary number of bunch-groups with a few general parameters:

- Affected bunches – range definition in MATLAB notation: *startpos:endpos:stepwidth*.
- Current to be stored in every bunch of this group.
- Additional flags e.g. whether the given current may be adjusted to match the desired overall target current.
- Prioritization of bunch groups.

Besides the currently supported fill patterns, this scheme will enable special fill patterns like alternating bunches with high and low current which is used as a test setup for BESSY-VSR [6] studies, or additional bunches with varying currents (down to 5 µA) for even shorter pulses in low-α mode.

CONCLUSION

A fully automated injection control software is absolutely essential for top-up operation with a fill pattern tailored to the users needs at BESSY II.

It has also completely replaced the manual injection procedure during low-α runs, machine- and beamline-commissioning and machine studies as well as for beam-scrubbing at high stored currents.

REFERENCES

[1] S. Khan, et al., “Femtosecond Undulator Radiation from Sliced Electron Bunches”, Phys. Rev. Lett. Vol. 97, Iss. 7 (2006).

[2] P. Kuske, et al., “Preparations of BESSY for top-up operation”, EPAC’08, Genoa, Italy (2008).

[3] K. Holldack, et al., “Single bunch X-ray pulses on demand from a multi-bunch synchrotron radiation source”, Nature Communications 5, Article number 4010 (2014).

[4] F. Falkenstern, et al., “BunchView / a fast and accurate bunch-by-bunch current monitor”, DIPAC’09, Basel, Switzerland (2009).

[5] D. F. Förster, et al., “Phase-locked MHz pulse selector for x-ray sources”, Optics Letters 40, (10), 2265-2268 (2015).

[6] “BESSY VSR, the variable pulse-length storage ring proposed by Helmholtz-Zentrum Berlin”, [Online], `http://www.helmholtz-berlin.de/zentrum/zukunft/vsr/summary_en.html`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PID_TUNE: A PID AUTOTUNING SOFTWARE TOOL ON UNICOS CPC

R. Martí, B. Bradu, E. Blanco Vinuela, CERN, Geneva, Switzerland
L. Frutos, R. Mazaeda, C. Prada, University of Valladolid, Valladolid, Spain

Abstract

PID (Proportional, Integral and Derivative) is the most used feedback control algorithm in the process control industry. Despite its age and thanks to its simplicity in terms of deployment and its efficiency in most of industrial processes, this technique still has a bright future. The major challenge in using PID control is to find the optimal set of parameters to tune the controller. This may be a complex task as it mostly depends on the dynamics of the process being controlled. In this paper a tool able to provide the engineers a set of PID parameters in an automated way is proposed. The tool offers auto-tuning methods, both in open and close loop, and others can be added as it is designed to be flexible. The tool is fully integrated in the framework UNICOS (CERN Standard Control framework) and can be used to tune multiple controllers at the same time, directly from the supervision layer.

INTRODUCTION

The largest majority of controllers found at the regulatory layer in the process industry are of PID (Proportional, Integral and Derivative) type. At CERN, around 4900 PID controllers are employed on the LHC cryogenic control system, 870 are employed on cooling and ventilation installations. In total, more than 8000 PID controllers are used at CERN in control systems based on the UNICOS (Unified Industrial Control System) framework [1]. All these controllers are implemented in more than 300 different PLCs (Programmable Logic Controller). Therefore, a tool to automatically tune the PID parameters would be a valuable asset in terms of controller robustness and hence in improving uptime and operation of the CERN industrial plants.

Although the PID controller has only three parameters, it is not straight forward, without a systematic procedure, to find suitable parameters. The main goal of this work is to provide an auto-tuning tool where different methods are available. The initial implemented methods were selected to cope with real industrial process scenarios. Two main features are searched: easiness of use, to allow operators to perform the task, and flexibility to extend the tool with new methods to cope with complex processes.

The article is organised as follows: first, the real implementation of the tool is depicted, second, the different tuning methods included in the tool are described including their suitability on different processes. Then, real industrial use cases with different auto-tuning methods employed are depicted along with a performance comparison between them. Finally, the paper concludes with the most important ideas and future work.

TOOL IMPLEMENTATION

The PID structure taken into consideration to design the auto-tuning tool is the ISA (International Society of Automation) version used in the UNICOS framework. The control law can be seen in the formula (1)

$$K_c \cdot (e(t) + \frac{1}{T_i} \cdot \int e(t)dt + T_d \cdot \frac{d}{dt}e(t)) \qquad (1)$$

The tool is fully integrated in the UNICOS framework at the supervision layer, the WinCC OA SCADA (Supervisory Control and Data Acquisition) from SIEMENS. It has been implemented in the WinCC OA control scripting language based on ANSI-C. The PID algorithm remains untouched in the PLC (control layer) and only its parameters are set once the tuning process is validated. The architecture and data flow is depicted in Figure 1.

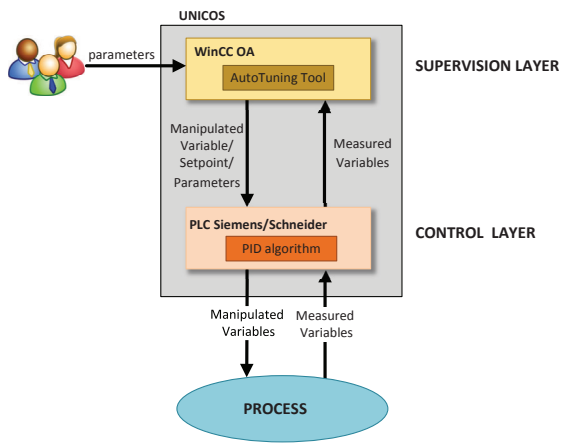


Figure 1: PID auto-tuning tool architecture.

The human interaction is made through dedicated interfaces embedded in the UNICOS controller faceplate. The user can select the method, configure its options, initiate the auto-tuning, evaluate the behaviour and ultimately set the new parameters to the controller. An example of such interface is shown in the Figure 2.

AUTOTUNING: BASIC CONCEPTS AND METHODS

The PID auto-tuning concept refers to the capability to automatically compute the parameters of a PID connected to the real plant. Many different approaches are available. This work focused on developing an auto-tuning tool, PID_TUNE, launched by the operators or the control engineers on their own initiative, performing the chosen tuning algorithm and

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

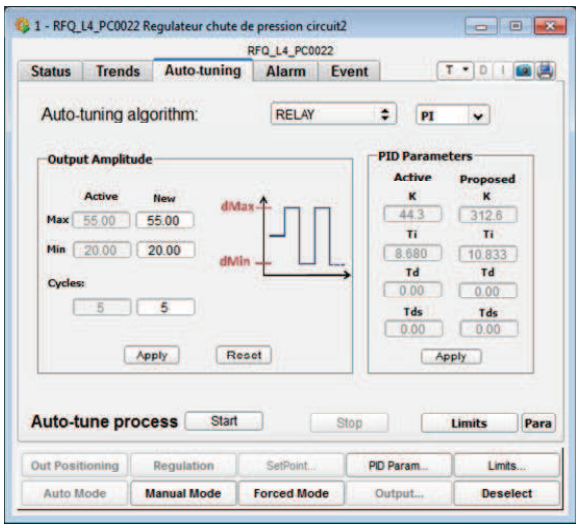


Figure 2: UNICOS auto-tuning user interface: Relay method.

presenting the results to the user who ultimately decides to apply them to the controller.

It is of general consensus that any skilled professional can tune a control loop better than any auto-tuning mechanism, however this task may be very time consuming. However the number of loops and the availability of experts can condition this assessment to the point that an auto-tuning tool can be of great help for the day to day operation of the plants.

The behaviour of an auto-tuning algorithm is based on what an expert would do when tuning that loop by his own. First a phase of observation, then in function of the control objectives selection of the method and finally applying the PID parameters. The first phase includes process data taking and this is done by stimulate the process by either moving the manipulated variable or the controller setpoint. Evidently this settles a first trade off of stimulating enough the process to get meaningful experimental data versus creating a process perturbation which can be fatal for the process itself. To make this feasible, several constraints must be taken into account: e.g. controlled and measured variable limitations, noise.

The selection of a method depends on the control needs which, in turn, depends on the analysis of the control problem. There is no a general rule although the methods available in the tool can cope with the majority of the processes as they are rather universal. Three methods has been included: Relay, SIMC and IFT.

Relay Method

The relay auto-tuning method was proposed to automate the Ziegler-Nichols ultimate cycling tuning method [2], where the critical gain (K_{cr}) and period (P_{cr}) are determined with a proportional-only regulator. The controller gain is gradually increased until an oscillation is obtained. The procedure concept is rather simple but difficult to automate ensuring the safety of the installation as the oscillation amplitude must be always kept under control. To overcome possible instabilities the relay incorporates a hysteresis [3].

The relay is connected in the feedback control loop instead of the PID controller (Fig. 3). It alternates the output between 2 values when the difference between the set-point and the controlled variable changes sign. This is the way it obtains a limited cycle oscillation.

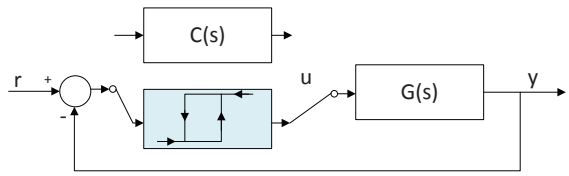


Figure 3: Relay tuning.

Some practical issues must be addressed when implementing this method. First, the measurement noise may give errors in detection of peaks and zero-crossing, thus again a hysteresis is used. It is a common practise to start the method when a steady state [4], of the process is identified to facilitate the sustained oscillations although the tool includes an algorithm to automatically identify steady state [5].

The advantage of this method is the simplicity of its use. The user only has to introduce the amplitude of the hysteresis and the number of cycles to detect sustained oscillations (Fig. 2). The method fits in processes with unknown dynamics.

SIMC

Another well known group of auto-tuning methods is based on plant models [6]. Basically, two different steps compose the methodology, first a process identification phase and then the application of the tuning rules. Obviously this approach is heavily constrained by the availability of the process model.

The developed tool allows users to identify a first-order model by applying a step on the manipulated variable. In addition to this, a steady-state automatic identification algorithm has been included to fully automate the identification phase. Then the tool suggest the best parameters according to well known tabulated values.

The SIMC method was chosen due to its advantages regarding others model based techniques [7]. The main, the simplicity in terms of parameterisation: SIMC has just a single tuning parameter, the desired performance. Its selection is a trade-off between performance (“tight” control) and robustness (“smooth” control). Therefore, the operators or control engineers can easily choose what kind of response they desire in the feedback control loop. The method is applicable to stable and low order systems with not complex dynamics.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

IFT: Iterative Feedback Tuning

The previous two methods have some notably drawbacks: on one side unpleasant disturbances are introduced to the process and, on the other, the need of a model of the plant. Also the algorithms need to disable the PID controller while the auto-tuning process is running. These justified the inclusion of a new method in the tool that ideally does not require the knowledge of any explicit model of the plant and its execution would not disturb much the nominal loop dynamics.

The IFT [8] is a technique inspired by the iterative parametric optimisation approach. It is entirely working in closed-loop and making random but contained perturbations in the loop to find the optimal parameters. It computes a cost function to minimise the current value of the measured value and a desired first order response.

The IFT algorithm is particularly suitable for the industrial use as it fulfils the requirements of being a model-free and closed-loop procedure which does not demand a very severe modification of the nominal conditions. The algorithm has the capability of rejecting efficiently disturbances, crucial for the regulatory control widely deployed in process industries. The IFT tuning algorithm included in the tool has been developed to ease the use of the IFT method, low parameterisation but still offers a minimum set of practical safeguards constraining the freedom of the underlying algorithm during the tuning procedure.

CASE STUDIES

In order to commission and validate the auto-tuning tool together with its algorithms, the tool was deployed in the LHC (Large Hadron Collider) cryogenics simulator. The simulator provides the same UNICOS engineering environment (SCADA and PLCs) as in any production system but connected to a simulated process. The second phase was to deploy it to a real case with complex dynamics, a cooling plant of the LHC.

Auto-tuning on the LHC Cryogenic Simulator

This real-time simulator is able to imitate the real transients of the LHC cryogenics and it embeds a identical version of the real control system. A flow control loop has been selected as candidate to perform the auto-tuning. This PI based control loop regulates the amount of gaseous helium circulating in LHC to maintain the thermal shielding of superconducting magnets around 80 K. It was originally too sluggish, generating flow oscillations and overshoots when disturbed by the circuit pressure changes.

Al three methods (Relay, SIMC and IFT) were applied on this control loop to find its PI parameters. The parameters are compared with the original ones (Table 1).

A significant pressure disturbance was applied in the simulator (t = 2 min) in order to compare the controller performance with the different parameter sets (Figure 4). The three auto-tuning parameter sets are resembling and then provide a controller similar response, but in any case faster and more stable than the original one.

Table 1: Auto-tuning Results on the Cryogenic Simulator

Tuning	Kc	T_i	Overshoot	Oscillation
Original	1	200	22 %	6 %
Relay	9	12	0.5 %	1 %
SIMC	4.5	7	1 %	1.3 %
IFT	13	11	0.5 %	0.5 %

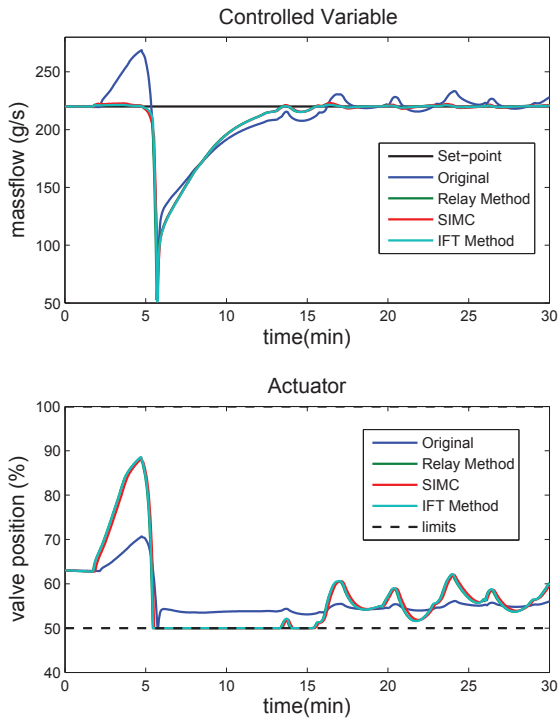


Figure 4: PI controllers performance in simulation.

Auto-tuning on a LHC Cooling Plant

As the first real test on a production system, the tool was used on a chilled water production unit providing water at 5 $°C$ for LHC. The operation team observed instabilities on the condenser output temperature of the chiller at 25 $°C$. The tuning of the associated PI controller was troublesome because this process is always disturbed and unstable due to a noticeable delay time. The temperature deviation to operate the chiller should be less than 1 $°C$ and indeed it was oscillating by around 2 C with the valve moving wildly between 0 % and 100 %. The relay method was the first method used. This allowed to stabilise the temperature with a reasonable control effort on the valve. However there were important disturbances all the time, so the temperature was still oscillating too much (Fig. 5). Then, an auto-tuning in close loop using the IFT method was applied as disturbance

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

rejection is one of its features. The results were satisfactory regarding the operation requirements (Fig. 6).

The different tuning parameters found are compared in Table 2 with the corresponding oscillations measured on the controlled temperature ($|\Delta y|$) and on the valve ($|\Delta u|$).

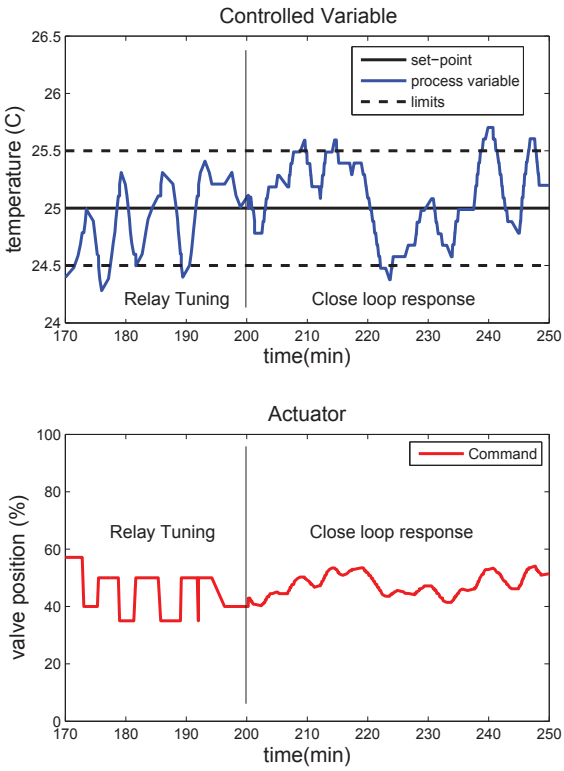


Figure 5: Relay method results on a LHC cooling unit.

Table 2: Auto-tuning Results on a LHC Cooling Unit.

Tuning	Kc	T_i	$\|\Delta y\|$	$\|\Delta u\|$
Original	8	0.5	2 °C	100 %
Relay	8.3	311	1.3 °C	10 %
IFT	50.9	5666	0.5 °C	20 %

CONCLUSIONS

An auto-tuning tool for PID controllers is presented in this paper. It is fully integrated, at the supervision level, in the CERN control framework (UNICOS). The tool does not imply any PLC control logic modification. This tool allows operators and control engineers to apply different auto-tuning methods on the regulation loops depending on the process and on the operational constraints. Three initial methods have been implemented and tested: Relay, SIMC and IFT. The tool has been validated in a real scale simulator and then applied in a production system, a LHC cooling plant. It has demonstrated its usability and its performance where the proposed control loops have been properly tuned. In the coming months, this auto-tuning tool will be deployed on all CERN control systems using UNICOS. The tool is designed in a flexible and open way so in the future, other methods could be easily added.

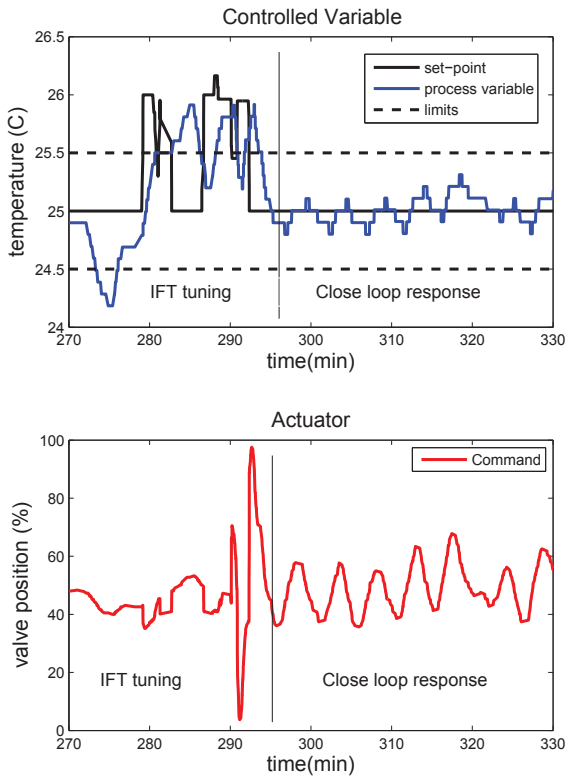


Figure 6: IFT method results on a LHC cooling unit.

REFERENCES

[1] Ph. Gayet and R. Barillère. “UNICOS a framework to build industry like control systems: Principles Methodology”. 10th ICALEPCS Int. Conf. on Accelerator and Large Expt. Physics Control Systemns. Geneva (Switzerland), 2005.

[2] J.G. Ziegler and N.B. Nichols. “Optimum setting for automatic controllers”.Transactions of the ASME,64, 1942.

[3] K. Aström and T. Hägglund. “Automatic tuning of simple regulators on phase and amplitude margins”.Automatica, 20, 1984.

[4] S. Dormido and F. Morilla. “Auto-tuning and Antiwindup methods in PID controllers”.UNED Press, 1995.

[5] R. Russell Rhinehart.“Automated Steady and Transient State Identification in Noisy Processes”. American Control Conference (ACC), Washington (USA), 2013.

[6] A. O’Dwyer. “Handbook of PI and PID Controller Tuning Rules”.Imperial College Press, 2006.

[7] Sigurd Skogestad and Chriss Grimholt. “The SIMC Method for Smooth PID Controller Tuning”. PID Control in the Third Millennium.Springer. 2012.

[8] H. Hjalmarsson, M. Gevers and O. Lequin. “Iterative feedback tuning: Theory and applications”. IEEE Control Systems Magazine 18(4),1998.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

AUTOMATIC FEL OPTIMIZATION AT FERMI

G. Gaio, M. Lonza, Elettra Sincrotrone Trieste, Trieste, Italy

Abstract

FERMI is a seeded Free Electron Laser (FEL) located in Trieste, Italy. The machine setup and optimization is a non-trivial problem due to the high sensitivity of the FEL process to several machine parameters. In particular, the electron bunch trajectory and its spatial overlap with the seed laser beam represent one of the key aspects to optimize and then preserve during machine operation.

In order to ease the FEL tuning and to guarantee a long term stability of the photon beam, a software process integrated into the feedback systems performs automatic trajectory optimization of both the seed laser and the electron beams. The algorithm implementation, the results and the operational issues are presented.

INTRODUCTION

The FERMI machine setup is a non-trivial task. Although a deeper knowledge of FEL mechanisms in parallel with new diagnostic tools have reduced the time spent for machine tuning, new and more challenging requests coming from the users community have significantly increased the complexity of the accelerator setup and its operation during the experiments.

One of the priorities is the preservation of the FEL radiation quality in the long term. Over the years a number of feedback loops have been added in order to cope with subsystem instabilities [1]. However, even thermal drifts of only a few hundredths of degrees or requested changes of the machine working point tend to move the machine away from the optimum in terms of intensity, stability and spectral radiation purity.

In order to tune the machine for its best performance, a well established sequence of operations has to be carried out. At first, the photon-injector and seed laser systems have to be prepared, transversal and longitudinal electron beam dynamics in the linear accelerator (linac) have to be tuned, and then the FEL tuning process along one of the two undulator chains has to be performed. At the end, the photon beam is driven into the photon beam transport system and aligned in the experimental chamber. Before each users beam time one day is usually dedicated to machine preparation and tuning but, in case of non-standard experiments, up to five days have to be reserved for this task.

The tuning process which is performed by the physicists, in most of the cases consists of a number of manual or automatic scans of actuators versus machine variables, leaving to the physicist the final duty to set actuator to the optimum value. Instead, the optics matching, which imposes the design values of the Twiss functions to the electron beam [2], is completely automatic. There are two main reasons why the optics tuning is performed using the matching program: the first is that the problem is difficult to be managed (up to 6 dimensions in the linac optics). The second reason is that the model in terms of relations between quadrupole strengths and Twiss functions is well known. The availability of a system model reduces the optimization problem to a task that a proper algorithm quickly solves.

However, a system model cannot be always taken for granted. During the commissioning of a new system (e.g. a particle accelerator), the difference of the real system from the model could drive the model based optimization to frustrating results. In order to obtain a correct system model, software based simulators with adaptive and diagnostic capabilities have been developed and are currently supporting operations in several particle accelerators.

Although this kind of software could be applied to the entire FEL from the linac gun to the photon beam in the experimental chamber [3], its usage as an online tool has still to come. In particular, the multiplicity of the systems involved (photo-injector and seed laser beams, electron beam and FEL beam transport) and their nonlinearity have limited the adoption of these tools to only single well known subsystems, leaving to the manual intervention the rest of the machine preparation.

However, the manual tuning, which is a complex task, tends to be inefficient because the expertise of the personnel involved in machine operations is not homogeneous.

FEL PROCESS

The FERMI FEL design is based on an external seeding scheme. Two undulator chains, FEL-1 and FEL-2 operated one at a time, provide radiation in two different ranges, 100-20nm and 20nm-4nm respectively. In the modulator (first undulator) of both undulator chains, an external seeding laser in a tuneable range of 228-265 nm, overlaps in time and space the electron bunch provided by the linac. The resultant energy modulation of the electron bunch is then converted into charge density modulation by a chromatic dispersive section following the modulator. The electron bunch then passes through a series of undulators tuned in magnetic field so that coherent FEL radiation is emitted at one of the harmonics of the seed laser.

In FEL-2 the process is made in two stages. The radiation emitted by the first stage, that is similar to FEL-1, is used as seed in the second stage. In this way the seed laser wavelength is downshifted twice reaching shorter wavelengths.

The undulators and the seed laser are the most important devices involved in the tuning process. A software supervisor [4] taking into account the relations between the undulator magnetic strength, the electron beam energy and the seed laser wavelength, sets the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

undulator gaps and the seed laser energy according to the user request in terms of FEL polarization and wavelength.

Other two key variables, the dispersive magnet strength and the seed laser power, are optimized by performing scans versus the FEL spectrum. Whereas the mentioned variables are changed according to the requested FEL parameters (power, wavelength and bandwidth), other variables have to be kept constant during this process. In particular, controlling the overlap of the seed laser and the electron beam in the undulators is one of the most important and challenging tasks due to the high number of variables involved.

Seed Laser Alignment

The alignment procedure consists in measuring the electron beam position before and after the modulator by means of two YAG screens. After inhibiting the electron beam, the position of the seed laser beam is acquired by the same couple of screens and moved in order to overlap the electron beam. The procedure, which is carried out manually, has the drawback of being destructive (a screen has to be inserted) and complex because the seed laser has to be steered on the two screens one at a time. At the end of the procedure, which could take several cycles of screen insertion/extraction, the screens are removed and a shot-to-shot feedback loop is activated to keep the seed laser trajectory stable on two CCDs. The first CCD is located about 8 m upstream the modulator. The second should have been located inside the modulator but unfortunately the laser beam is measurable only intercepting both the seed laser and the electron beam. In order to have an estimation of the laser beam size and transverse position, the seed laser beam is split before the modulator. A small portion of the beam follows an equivalent optical path inside a closed box and its transverse profile, the so called “virtual undulator”, is acquired by a CCD.

Although this solution gives a good estimation of the seed laser position and shape in the modulator, the virtual modulator is very sensitive to thermal drifts due to the multiple beam reflections in the box. Moreover, a mirror installed just in front the modulator, used to steer the laser beam into the modulator, increases the dependence of the whole system on thermal stability since its movement is not compensated by the feedback.

Another critical point is the dependence of the seed laser optical transport on wavelength change. Peaks of absorption of the mirrors at 250 nm cause an intensity drop of the profile acquired by the CCD, which can only in part be compensated by the automatic gain control of the CCD itself. As a result the estimation of the centroid of the laser profile at this wavelength becomes inaccurate thus reducing the benefit of running a trajectory feedback based on these CCDs.

Electron Beam Alignment

The mechanical misalignments of undulators and magnets have been fixed during the initial FEL commissioning. Instead, the electron beam steering based on the photon beam emitted by each radiator is carried out before each machine run. In case the radiator emissions are not well superimposed, the electron beam trajectory is adjusted inside the undulators by changing the trajectory feedback set-points.

Although the optimal electron beam trajectory is then maintained constant by a feedback loop, offsets of a few tens of microns are often empirically applied to maximize the intensity of the FEL emission.

The alignment of the first stage of FEL-2 is one the most challenging task because no available diagnostics is able to discriminate the emission of the first and the second stage [5]. In this case the insertion of a screen acquired in single shot (to avoid high radiation doses) is the only available option. Once the screen is extracted, the FEL-2 first stage is virtually a black box leaving to the operator just the possibility to optimize blindly the electron beam trajectory.

AUTOMATIC TRAJECTORY OPTIMIZATION

Optimizing a multidimensional system is a difficult task when the relations between the system inputs and the objective function are uncertain. Humans tend to explore all the inputs in order to create a model. Random scans are essential to find the inputs that are most correlated with the objective function; these are then scanned one at a time to find best values and the optimization process is iterated until the objective function reaches a satisfying value.

A similar approach is also adopted by stochastic optimization (SO) algorithms. Contrary to deterministic optimization where the relation between the objective function and the input values is well defined, stochastic optimization is based on the random exploration of the system input space. Stochastic approximation, simultaneous perturbation, simulated annealing, random search and evolutionary algorithms are the most promising optimization methods. Their strategies slightly differ one from the other on how they introduce randomness in the search process and on how they define the climbing step to reach the optimum. Formally it has been demonstrated that the best search algorithm does not exists (see No Free Lunch theorems [6]) and the success of an optimization algorithm depends on how it fits its problem.

The search of the optimal seed laser and electron beam trajectories (array of horizontal/vertical beam positions) presents all the characteristics of an optimization problem that could be solved by this kind of algorithms. The system is multidimensional, non-linear and uncertain. In FERMI the objective function used so far is the FEL radiation intensity measured by a gas monitor or by a CCD camera. A more complex objective function to be used in the future could be a combination of FEL parameters (power, bandwidth and spectral purity) opportunely weighted.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The algorithm implementation takes advantage of a software communication protocol developed in house, called NRM (Network Reflective Memory) [7] that is used to distribute across the control system a shot-to-shot timestamp, the so called "bunch number". Moreover, the NRM transmits all the variables involved in the feedback loops and the photon beam transport diagnostic values. This real-time network, which was mainly developed for the shot-to-shot feedbacks, replicates snapshot of all FERMI important parameters in all the computers of the control system, allowing shot by shot data analysis that go beyond the feedback system purposes.

The optimization process is strictly integrated into the trajectory feedback loops; it changes at each shot the feedback set points, acquires sensors and actuators and couples them with the objective function.

The algorithm consists of six steps:

- Collects N trajectories and the corresponding objective function values (ex. FEL output intensity);
- Sorts the trajectories according to the objective function value in descending order;
- Calculates a "**golden**" trajectory by averaging the first M trajectories (M is usually 10% of N);
- Calculates a "**reference**" trajectory by averaging the remaining N-M-P trajectories where P is the number of the "**worst"** trajectories (P usually 10% of N);
- Sums the difference between the golden and the reference trajectory to the feedback set point;
- Go back to the first step until a reasonable result is obtained.

The optimization algorithm can run in active or passive mode. In the active mode the trajectory feedback injects a perturbation in the system by changing the set-point of a selection of sensors. In order to speed up the scan accordingly to the feedback system response, the perturbation applied to each horizontal/vertical position usually resembles a 2D square spiral. The spiral scans have to be opportunely desynchronized between different positions in order to maximize the number of trajectory combinations. The maximum spiral dimension is normally determined a priori according to the physical magnitudes of the underlying process. It can be constant during the whole optimization process or decrease in time according to simulated annealing principles. In case the initial spiral dimension cannot be determined, a warming-up process adjusts the spiral size until the absolute value of the Pearson correlation between the horizontal or vertical position and the objective function reaches a predetermined value.

The reduction of the convergence time is the main advantage of injecting a controlled perturbation. The drawback is an additional perturbation of the beam trajectory that could affect user experiments.

In passive mode the exploration is allowed by the shot-to-shot noise present on the beam, making this optimization process virtually transparent to the experiments. A trick for increasing the noise magnitude when it is not sufficient to improve the objective function consists in increasing the trajectory feedback gain.

In FERMI the alignment of the seed laser with the electron beam is based on the active optimization mode. The trajectory feedback starts rotating the seed laser spot on two CCDs and four iterations of 130 shots are sufficient to maximize the FEL intensity (see Fig. 1).

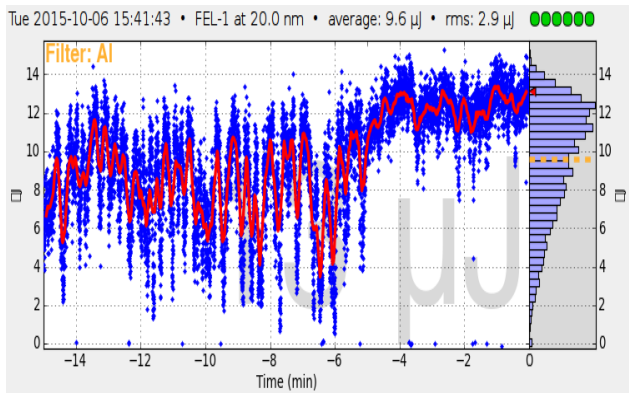


Figure 1: Increase of the FEL output intensity and stability achieved by running the seed laser alignment optimizer.

Usually the optimal electron beam trajectory in the undulator is found manually and then the optimization algorithm in passive mode is launched to see whether there is room for further FEL improvements. At least five iterations of 600 shots are necessary to get significant results. In order to avoid the optimization statistics been biased by residual dispersive trajectories, the matching between the electron energy beam and undulators configuration has to be verified in advance.

Alternatively to the gas monitor, a CCD in the experimental station can also be used to measure the FEL intensity. The advantage is that the optimized parameter is closer to the experiment, thus taking into account also effects occurring downstream the gas monitor (Fig. 2).

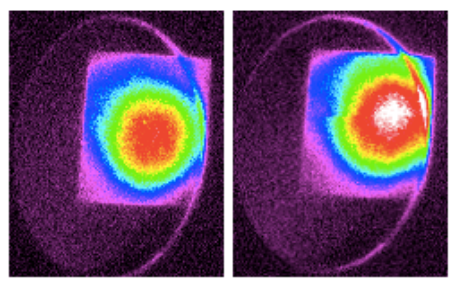


Figure 2: Profile of the FEL radiation on a YAG screen in the experimental chamber before (left) and after (right) the optimization of the electron beam trajectory in the undulators (15% of intensity increase).

CORRELATOR

The automatic detection of the machine parameters that most affect the noise of the FEL beam and the launch of the optimization procedure on those parameters allow a faster achievement of the optimal working point.

The number of the variables that have to be monitored and correlated with the FEL parameters is very large and

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

inhomogeneous, so at the moment the search is restricted to those involved in the FERMI beam based feedbacks.

In order to evaluate the best noise detecting strategies a software application written in Matlab, acquires every second the last N shots (N between 100 and 2000) of 214 sensors/actuators, correlates them with the FEL output intensity and finally sort them in descending order according to the correspondent correlation value.

In an ideal world the correlation between feedback variables and FEL should be as low as possible. Sensors with high correlation values indicate a displacement from the optimal values, whereas high correlation in the actuators could suggest a feedback malfunction (too high feedback gain, feedback loops crosstalk, sensor malfunction, etc.).

It has been empirically noticed that a gap of more than 20% between the top shots of the list and the rest is a clear indication of malfunction. For example, this tool can identify with a very good reliability the temporal and transversal misalignment of the seed laser with respect to the electron beam (see Fig. 3), glitches on the RF plants and in general a misconfiguration of any feedback system.

Figure 3: List of sensors/actuators most correlated with the FEL intensity; seed laser vertical positions on the two CCDs on top of the ranking indicate a seed laser transverse misalignment.

CONCLUSION

In order to cope with system uncertainties a procedure based on stochastic optimization principles has been developed and integrated in the seed laser and electron beam feedbacks. Beam nonlinearities and thermal drifts affecting the feedbacks systems could be well recovered by automatic optimization procedures based on stochastic algorithms. In order to proactively detect the arising of system drifts that affect FEL radiation, a tool that correlates all the feedback sensors/actuators with the FEL output power has been developed. Recurrent patterns in correlation values are useful to detect noise sources and implement the appropriate countermeasures.

Being able to easily optimize the accelerator and the FEL will be a great value for the future FERMI experiments.

REFERENCES

[1] G. Gaio et al., "Evolution of the FERMI beam based feedbacks", THPPC129, ICALEPCS2013, San Francisco CA, USA.

[2] S. Di Mitri et al., "Electron beam optics and trajectory control in the Fermi free electron laser delivery system", Phys. Rev. ST Accel. Beams 15, 012802.

[3] G. De Ninno et al., "Start-to-end time-dependent study of FEL output sensitivity to electron-beam jitters for the first stage of the FERMI@Elettra project", MOPPH058, FEL 2006, Berlin, Germany.

[4] C. Scafuri, B. Diviacco, "Automation of the Wavelength Change for the FERMI Free Electron Laser", TUPPC052, ICALEPCS2013, San Francisco CA, USA.

[5] L. Giannessi, "First lasing of FERMI FEL-2 (1°stage) and FERMI FEL-1 recent results", MOOB06, FEL2012, Nara, Japan.

[6] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Optimization", IEEE Transaction on Evolutionary Computation, Vol. 1, No. 1, April 1997

[7] L. Pivetta et al., "The FERMI@Elettra distributed real-time framework", THDAUST03, ICALEPCS2011, Grenoble, France.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SYSTEM IDENTIFICATION AND ROBUST CONTROL FOR THE LNLS UVX FAST ORBIT FEEDBACK

D. O. Tavares#, LNLS, Campinas, Brazil
D. R. Grossi, São Paulo University, São Carlos, Brazil

Abstract

This paper describes the optimization work carried out to improve the performance of the LNLS UVX storage ring fast orbit feedback system. Black-box system identification techniques were applied to model the dynamic behavior of BPM electronics, orbit correctors, communication networks and vacuum chamber eddy currents. Robust control techniques were employed to analyze and optimize closed-loop performance and robustness.

INTRODUCTION

The LNLS storage ring FOFB system has been operating for users since March 2013. RMS stability integrated from 1 Hz to 500 Hz is 1% beam size in the horizontal plane and 6% for vertical in worst case (1% coupling is assumed). Disturbances originating on floor vibrations account for less than 0.2% RMS in horizontal plane and 2.5% RMS in vertical plane. The dominant disturbance source to be attenuated by the FOFB system is Elliptically Polarizing Undulator (EPU) gap/phase variations, while orbit disturbances caused by power supply ripple (up to 5% beam size, in vertical plane) should not be significantly amplified. The following sections describe the approach adopted to optimize the system performance and robustness.

HARDWARE DESCRIPTION

The LNLS FOFB system is a multiple-input multiple-output feedback control system comprising 48 sensors (24 beam position readings per transverse plane) and 42 actuators (18 horizontal orbit correctors + 24 vertical orbit correctors) running at 3.125 kHz update rate.

The data acquisition and control system hardware is composed of one central real-time controller (PXI-8108) and 12 data acquisition nodes (CompactRIO-9144 EtherCAT chassis) with analog and digital interface modules. Each acquisition/actuation node has an FPGA available for digital filtering, currently used for decimation (from 100 kS/s oversampling to 3.125 kS/s) and dynamic response compensation for each BPM reading and orbit correct current setpoint. Two daisy chain EtherCAT networks are used as deterministic network for sensor and actuator data distribution to/from the controller and provides synchronization between all nodes below 1 µs. Bergoz MX-BPM electronics are used for RF signals processing and delivers analog signals proportional to the beam position. In-house developed power supplies with analog interface and analog current regulator (Proportional-Integral type) are used to drive the orbit corrector magnets [1].

#daniel.tavares@lnls.br

SYSTEM IDENTIFICATION

Figure 1 depicts the general structure adopted for modeling the LNLS FOFB system using Laplace and Z transforms.

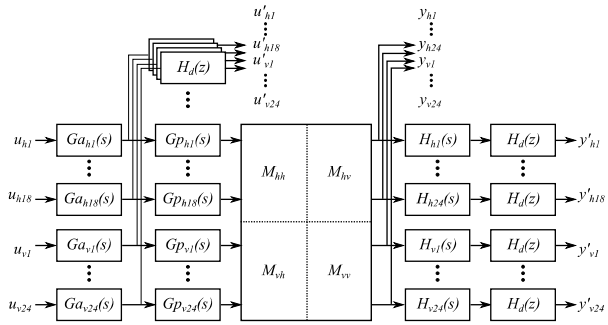


Figure 1: LNLS FOFB model structure.

The description of transfer functions and signals are given below:

- $Ga_{pj}(s)$: power supply + magnet impedance response of the j-th corrector of the p transverse plane (h = horizontal, v = vertical).
- $Gp_{pj}(s)$: magnet core + vacuum chamber magnetic field response (eddy currents) to coil current variations on the j-th corrector of the p plane.
- $H_{pi}(s)$: BPM electronics response of the i-th BPM of the p plane.
- $H_d(z)$: decimation filter response (discrete-time) at lower data rate (FOFB update rate) + communication network delay. Identical response for both BPM position reading and orbit corrector current reading.
- M_{p1p2}: static orbit response matrices ($p1$ plane BPMs and $p2$ plane orbit correctors). Horizontal and vertical plane matrices are represented by M_{hh} and M_{vv}, respectively. Crosstalk matrices (off-diagonal) are represented by M_{hv} and M_{vh}.
- u_{pj}: orbit corrector current setpoint (j-th corrector, p plane).
- u'_{pj}: orbit corrector current reading (j-th corrector, p plane).
- y_{pi}: actual beam position (i-th BPM, p plane).
- y'_{pi}: measured beam position (i-th BPM, p plane).

For black-box system identification the available ports for experiments comprise the discrete-time input signals u_{pj} and output signals u'_{pj} and y'_{pi}, with 320 µs sampling period. $H_d(z)$ transfer function is accurately known *a priori*, since it is fully determined by CIC decimation filters (factor 32 decimation rate, 2 sections and 1 differential delay) implemented in the nodes' FPGAs and one-cycle time delay due to the EtherCAT network data distribution.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Static orbit response matrices are readily obtained through the traditional method of small amplitude (tens of μrad) bipolar steps on each orbit corrector. For all purposes of this paper, M_{plp2} will be assumed to be known, including their variations due to known optics changes: ±2% tune shifts and extreme values of insertion device gap and phase configurations during normal operation.

Experiments

After experimenting with different types of excitation signals (white noise, sum of sinusoids, PRBS), a choice has been made for the pseudo-random binary signal (PRBS) due to its appropriate spectral content, easiness of use and periodicity enabling time-domain average of the experimental data. Special care has been taken to choose the sequence length in such a way that no major spectral line of the beam position readings and orbit corrector current readings (mainly due to magnets power supply ripple) aligned with some of the excitation harmonics. Figure 2 shows the chosen excitation pattern (62 points sequence, 750 Hz bandwidth, 9.2 μrad peak-to-peak excitation) in time-domain and frequency domain. For illustration, the beam position spectrum of a particular BPM is plotted in background.

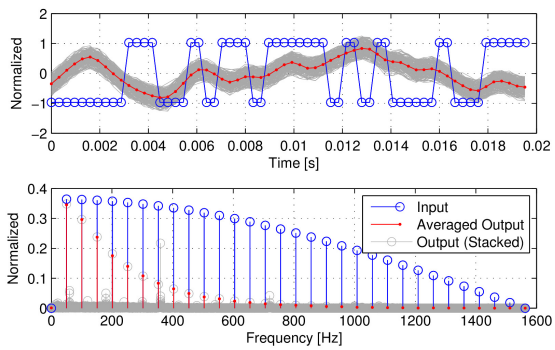


Figure 2: PRBS excitation signal of an orbit corrector and corresponding reading in a beam position monitor.

Each corrector was excited independently in order to simplify the identification process.

Input-Output Datasets

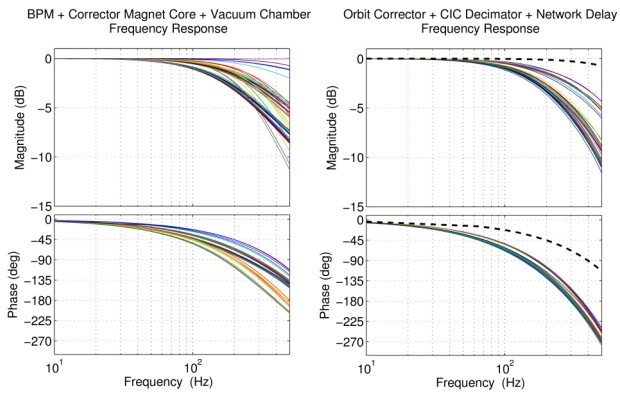
For orbit correctors, 42 datasets with 1 input (current setpoint) and 1 output (current reading) each were built from the acquired data. These datasets allowed black-box identification of discrete-time transfer functions in the form $Ga'_{pj}(z)=Ga_{pj}(z)H_d(z)$, that is, the combined response of power supply, magnet impedance, decimation filter and network delay.

For BPMs, 48 datasets with 1 input (filtered corrector current setpoint) and 1 output (beam position reading) each were built, where the orbit corrector to be excited was chosen as the corrector which produced the maximum static response at the BPM to be identified. The input data was then filtered by the identified model of the chosen orbit corrector resulting in the identification of the transfer function $H'_{pi}(z)=H_{pi}(z)Gp_{pj}(z)$, that is, the combined response of the i-th BPM and the magnet core plus vacuum chamber response of the j-th orbit corrector. Since the magnet core and vacuum chamber of all correctors were known to have bandwidth higher than 1.25 kHz, that is above the frequency range of interest (500 Hz), the $Gp_{pj}(z)$ dynamics were neglected.

Black-box Identification

Among several existing parametric black-box system identification methods, the Autoregressive model with Exogenous Inputs (ARX) was chosen [2]. In order to avoid biasing on the identification parameters, each dataset was averaged in time-domain: the output signals were segmented in 161 periods of 62 samples (approximately 3.2 s of data), being the first segment discarded to remove the transient dynamics. From the remaining segments, 80 segments were averaged and resulted in 62-sample long estimation dataset. Another 80 segments were averaged to form the validation dataset. Orbit corrector models of order 8 (16 coefficients) and delay of 3 sample times were used. BPM models of order 2 (4 coefficients) and delay of 1 sample time were used.

Figures 3a and 3b show the frequency response of the identified models for BPMs and orbit correctors, respectively.



(a) (b)

Figure 3: Frequency responses of identified (a) BPM models, and (b) orbit corrector models. $H_d(z)$ frequency response is plotted in dashed black curve for comparison.

The identified models presented normalized root mean squared errors (NRMSE) above 91% and 97% for BPMs and orbit correctors, respectively (100% meaning perfect fit and 0 meaning the fit is as good as of a straight line equal to the mean of the data). Residual analysis showed no statistically significant correlation between estimation errors (residues) and the excitation signal for all the identified models.

CONTROL DESIGN

A signal-based H_2/H_∞ optimal control approach has been adopted when designing the FOFB controller. A fixed-structure controller requirement was imposed by the available hardware, making the synthesis of an optimal controller not possible by classical tools or not computationally efficient [3]. Robustness was analyzed using uncertain plant subsystem transfer functions and worst-case singular values frequency response plots.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Weights

The key element of a signal-based control design approach is to establish inputs, outputs and transfer function weights which directly express realistic disturbances, sensor noise, actuator noise and performance objectives [4]. Closed-loop optimization is then performed based on an augmented plant as illustrated in Fig. 4.

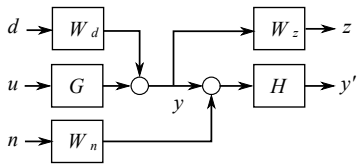


Figure 4: Augmented plant for signal-based control.

The blocks *G(z)* and *H(z)* are the plant and sensor matrix transfer functions, respectively, and $W_d(z)$, $W_n(z)$ and $W_z(z)$ are the weights for disturbance, noise and performance objectives.

In the case of the LNLS FOFB system, the $W_d(z)$ transfer function matrix was chosen to reflect the main electron beam disturbances on the UVX storage ring: (i) variation of EPU gap (ramp type); (ii) variation phase of EPU phase (ramp type); (iii) mains ripple from 60 Hz to 360 Hz (second order peak filters). By means of experiments, the orbit distortion profiles of each of these disturbances were obtained.

$W_n(z)$ was chosen as a diagonal transfer function matrix where each element is the worst-case (lowest beam current on operation conditions) standard deviation of the corresponding beam position measurement broadband noise integrated over the entire bandwidth.

$W_z(z)$ was chosen also as a diagonal matrix transfer function where each element is a first-order low pass filter with a given bandwidth and gain given by the inverse of the nominal beam size, meaning all beamlines are considered to be equally sensitive to electron beam orbit disturbances up to the bandwidth.

Actuator noise and amplitude range limitation were both considered to be negligible.

Performance and Robustness Metrics

FOFB performance and robustness were assessed by means of H_2 and H_∞ closed-loop system norms of the augmented plant, with negative feedback from $\boldsymbol{y'}$ to $\boldsymbol{u}$ by a controller with *C(z)* transfer function. The chosen performance channel was the summation of disturbance rejection and sensor noise transfer functions, with inputs $\boldsymbol{d}$ and $\boldsymbol{n}$ and output $\boldsymbol{z}$, denoted by $\|T_{d,n\to z}\|_2$, for which the H_2 norm has direct correspondence with the RMS-integrated criterion typically used to analyze FOFB system performance. The chosen robustness channel was the sensitivity transfer function at $\boldsymbol{y}$, denoted by $\|S_y\|_\infty$, for which the H_∞ norm simply means the worst-case multivariable gain from the disturbances to the actual beam position outputs. Despite being an indirect measure of robustness, it has straightforward practical meaning.

The H_2 and H_∞ norms were scaled in such a way that values below unity were said to satisfy the FOFB requirements. To this end, W_z was divided by a factor 0.1 to reflect 10% beam size stability requirement and by the square root of the number of beam position measurements (24, for each plane) in order to convert the 2-norm of the output vector to RMS value. Each input of W_d is divided by its expected maximum value, so that an unitary input means the worst-case disturbance. The sensitivity transfer function was divided by the allowed maximum amplification gain, $\sqrt{2}$ (3 dB).

Model Uncertainty

Despite adopting black-box system identification when modeling each BPM and orbit corrector dynamics, the transfer functions were assumed to have some level of uncertainty. In order to take this information into account when analyzing the control loop, classes of similar responses have been established from visual inspection followed by the determination of lower and upper bounds of the frequency responses through an input multiplicative uncertainty model of the type:

$$G_{uncertain}(z)=G_{nominal}(z)(1+W(z)\Delta(z)) \qquad (1)$$

where Δ(z) is a norm-bounded uncertain complex transfer function ($\|H(\Delta)\|_\infty<1$) and *W(z)* is a weighting transfer function of order 1. Figures 5a and 5b show the established frequency response classes.

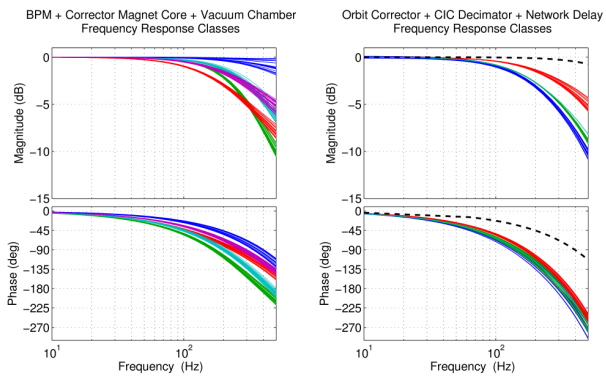


Figure 5: Classes of frequency responses of (a) BPM models, and (b) orbit corrector models. $H_d(z)$ frequency response is plotted in dashed black curve for comparison.

Control Optimization

Although mature frameworks for designing optimal controllers based on H_2 and H_∞ norms minimization exist (*e.g.* LQG, LMIs, μ-synthesis), these methods are not directly suitable for the LNLS FOFB system since the controller has a fixed-structure. For such class of controllers, optimizing the system norms would typically require nonconvex and nonsmooth optimization techniques. Although an efficient and scalable nonsmooth optimization method for solving this class of problems exists [3] and is available in Matlab Robust Control Toolbox since version 2010b, the present work has taken a simpler approach of exhaustive computation of system

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

norms for a grid of controller parameters. The next section describes simulation results which were obtained following this approach.

SIMULATION RESULTS

The feedback controller *C(z)* on the simulated control loops had the following structure:

$$C(z)=M_C\frac{K\cdot T_s}{z-1} \quad (2)$$

where M_C is a decoupling matrix, K is the controller gain and T_s is the sampling time (320 μs). M_C is the pseudo-inverse of the static orbit response matrix applying Tikhonov regularization as described in [5, 6]. The pseudo-inverse singular values $\hat{\sigma}_i$ have the following relation with the singular values of the orbit response matrix σ_i :

$$\hat{\sigma}_i=\frac{\sigma_i}{\sigma_i^2+\mu} \quad (3)$$

where μ is a real parameter to be tuned.

The closed-loop norms $\|T_{d,n\to z}\|_2$ and $\|S_y\|_\infty$ of the augmented plant (nominal case, with no uncertainty) were calculated for different values of K and μ for three different bandwidths in the $W_z(z)$ transfer function. Orbit correction on horizontal and vertical planes were treated independently. Figure 6 shows the results for each value of K, μ and bandwidth. The optimal controllers are those with minimum value of $\|T_{d,n\to z}\|_2$ meeting the constraint $\|S_y\|_\infty<1$, as summarized in Table 1.

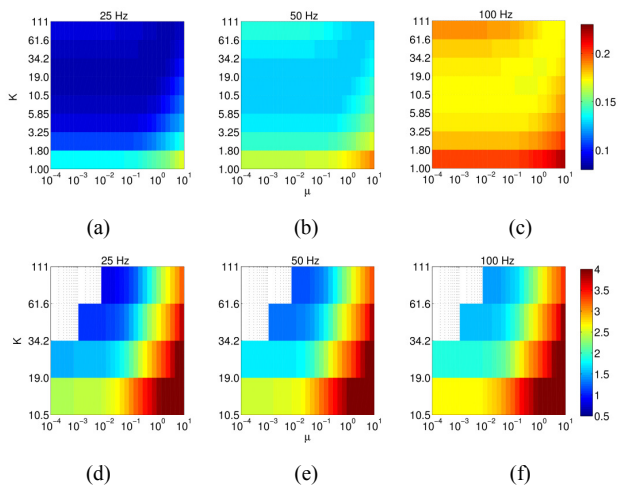


Figure 6: $\|T_{d,n\to z}\|_2$ costs meeting the $\|S_y\|_\infty<1$ constraint for each plane and beamline bandwidth: (a) horizontal, 25 Hz; (b) horizontal, 50 Hz; (c) horizontal, 100 Hz; (d) vertical, 25 Hz; (e) vertical, 50 Hz; (f) vertical, 100 Hz.

Table 1: Summary of optimal controllers.

Plane	BW	$\|T_{d,n\to z}\|_2$	$\|S_y\|_\infty$	K	μ
H	**25 Hz**	0.0855	0.7467	19.0	4.17e-1
H	**50 Hz**	0.1271	0.7294	10.5	2.81e-1
H	**100 Hz**	0.1710	0.7293	10.5	4.17e-1
V	**25 Hz**	0.7837	0.9746	111	1.17e-2
V	**50 Hz**	1.160	0.8585	61.6	5.30e-3
V	**100 Hz**	1.466	0.8480	61.6	7.88e-3

Figure 7 shows the frequency-dependent singular values of the sensitivity transfer function of the system for each controller in Table 1. The uncertainty on dynamics and orbit response matrix is taken into account. No closed-loop sensitivity function of the optimized loops significantly exceeds the 3 dB amplification reference line, indicating the uncertainties do not substantially degrade robustness. This was expected since the differences on dynamics are significant only for frequencies above the achieved closed-loop bandwidths.

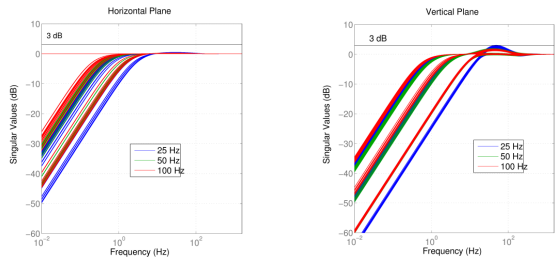


Figure 7: Singular values plot of sensitivity functions with uncertain plant: (a) Horizontal plane (b)Vertical plane.

CONCLUSION

A complete approach to identify the performance limits of the LNLS FOFB system has been presented, from the system identification to the control loop optimization. From this study it has been concluded that the LNLS FOFB system is limited in about 15 Hz closed-loop bandwidth (~25 Hz 0 dB crossover on sensitivity function), the main limitation being a ~1.5 ms delay due to communication network, decimation filters and BPMs responses. Due to the effect of noise amplification for frequencies above the crossover, higher control loop bandwidths improves performance only when beamlines are less sensitive to electron beam motion at higher bandwidths. The simulation results herein presented still need to be validated against experimental data.

REFERENCES

[1] L. Sanfelici et al., "LNLS Fast Orbit Feedback System", MOP263, PAC'11.

[2] L. Ljung, *System Identification: Theory for the User, 2nd edition*, (New Jersey: Prentice-Hall, 1999).

[3] P. Apkarian et al., "Nonsmooth optimization algorithm for mixed H_2/H_∞ synthesis", Conference on Decision and Control 2007.

[4] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design, 2nd edition*, (Chichester: Wiley, 2005).

[5] J. Rowland et al., "Status of the Diamond Fast Orbit Feedback System", RPPA10, ICALEPCS'07.

[6] S. Gayadeen et al., "Fast Orbit Feedback Control in Mode Space", THMIB07, ICALEPCS'13.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

NSLS-II FAST ORBIT FEEDBACK SYSTEM *

Y. Tian#, K. Ha, L. Yu, W. Cheng, J. DeLong, L. Dalesio, BNL, Upton, NY 11973, USA
W. Levine, University of Maryland, College Park, MD 20742, USA

Abstract

This paper presents the NSLS-II fast orbit feedback (FOFB) system, including the architecture, the algorithm and the commissioning results. A two-tier communication architecture is used to distribute the 10kHz beam position data (BPM) around the storage ring. The FOFB calculation is carried out in field programmable gate arrays (FPGA). An individual eigenmode compensation algorithm is applied to allow different eigenmodes to have different compensation parameters. The system is used as a regular tool to maintain the beam stability at NSLS-II.

INTRODUCTION

NSLS-II is a third generation 3GeV storage ring with ultra-low emittance [1]. The low emittance requires a very stable electron beam orbit. Applying the common rule (beam stability < 10% of beam size), NSLS-II needs to hold submicron beam orbit stability. The stringent orbit stability requires the orbit feedback system to be able to suppress various noises from low frequency ground motion to high frequency mechanical vibration [2].

The NSLS2 BPM electronics generates high resolution turn-by-turn, fast acquisition (10kHz) and slow acquisition (10Hz) data. Fast acquisition data is used for global fast orbit feedback. To ensure sufficient bandwidth of the feedback loop, 10kHz data from all storage ring BPMs are distributed within a short period of time. A two-tier communication architecture is implemented to ensure the time budget for the FOFB system is satisfied. The first tier is the communication between local BPMs and the cell controller at each cell. The second tier is the communication between the cells around the whole storage ring.

The FOFB system is a typical multiple-input and multiple-output (MIMO) system. One common feature of the traditional singular value decomposition (SVD) based orbit feedback algorithm is that it applies the same controller dynamics (such as a PID controller) to all eigenmodes. Since each eigenmode has a different frequency response, it is desirable to apply a different controller for each of the eigenmodes and thus each eigenmode has different compensation in the frequency domain [3]. The challenge for such individual eigenmode compensation is that the feedback system needs to carry out much larger calculations within the time budget of the FOFB system. In this paper, we present the implementation and commissioning results of the FOFB system at NSLS-II.

*Work supported by DOE contract No: DE-AC02-98CH10886
#ytian@bnl.gov

TWO-TIER COMMUNICATION

One common task for a global orbit feedback system is to deliver BPM data to the calculation unit, and to send the calculated result, i.e. the corrector setpoints, to the power supplies. Many communication protocols are used in different labs.

To design the communication protocol at NSLS-II, we take advantage of the geographic locations of BPMs and correctors at NSLS-II. The NSLS-II storage ring is separated into 30 cells. There are 6 BPMs, 3 fast correctors, and 6 slow correctors in each cell. To effectively deliver the BPM and corrector data, we designed a hardware unit in each cell to collect local BPM data, deliver the BPM data around the ring, and to send corrector setpoints to the correctors. We call this unit the cell controller.

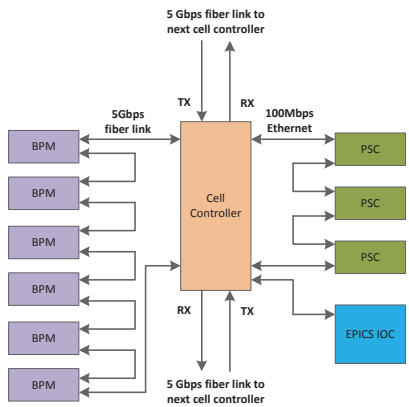


Figure 1: Two-tier communication.

By using cell controllers, two-tier communication is established, as shown in Figure 1. The first tier is the communication of the local data in one cell. In one cell, the cell controller and up to 12 BPMs are connected using fiber optic cables to form a ring structure. Currently, FOFB is only using 6 RF BPMs in the loop and it can be expanded to include 6 ID BPMs or X-ray BPMs. Each BPM sends its position data to its left and right neighbours, and passes through its neighbours' data. The data finally reach the cell controller and the cell controller saves the 6 local BPMs' data in its buffer. This communication protocol is called a serial device interface (SDI). On the other side, the corrector setpoints are sent from the cell controller to the local fast correctors via a similar SDI link. The only difference is that the power supply SDI link uses Ethernet PHY.

The second-tier communication is between the 30 cell controllers. The 30 cell controllers are connected using fiber optic cable to form a ring structure. Each cell

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

controller sends the local BPM data to its neighbour cell. Within 14us, each cell controller has all the BPM data. It is worth mentioning that the SDI protocol includes a clockwise and counter clockwise ring structure. The data communication has built-in redundancy.

COMPENSATION IN EIGENSPACE

The NSLS-II FOFB system applies a novel approach of feedback compensation in eigenspace [4]. Figure 2 shows the diagram of eigenspace decoupling and compensation in each eigenmode.

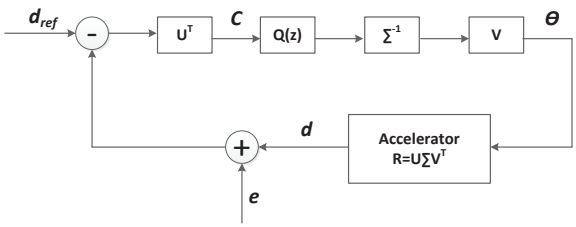


Figure 2: FOFB with eigenspace compensation.

In Figure 2, $\vec{e}$ is used to represent the noise in the system and $\vec{c}$ is used to represent the eigenspace projection of each input

$$\vec{c} = U^T(\vec{d} - \vec{d}_{ref} + \vec{e})$$

For each of the components of $\vec{c}$, an individual digital controller can be designed for the compensation of each eigenmode. The MIMO problem is converted to many signal-input signal-output (SISO) problems, for which control theory has many standard solutions. We use $Q_i(z)$ to represent the digital controller applied only on *ith* eigencomponent of $\vec{c}$.

The above eigenspace compensation algorithm allows us to apply different compensation parameters (for example, PID values) to each eigenmode and thus provide a flexible controller according to each eigenspace's transfer function. On the other hand, such an algorithm needs much many calculations than the traditional algorithm.

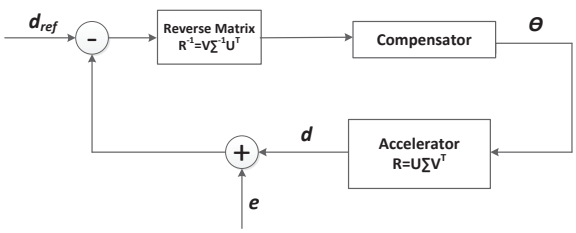


Figure 3: A typical SVD-based FOFB system.

Figure 3 shows a typical SVD-based FOFB algorithm. It shows that to calculate one corrector setpoint, the system needs to multiply one row of the reverse response matrix by the orbit error. This needs M multiply–accumulate (MAC) operations, where M is the number of BPMs. A typical PID compensator adds another 3 MACs. The total calculation amount is M+3 MACs.

For eigenspace compensation, as shown in Figure 2, the orbit error needs to first multiply each row of the U^T matrix to decouple into eigen components in eigenspace. Then a compensator is applied for each eigenmode. The third step is to combine all eigen components into corrector space by multiplying the eigen components by the V matrix. Apparently, due to the eigenspace decoupling, the amount of calculation increases by N times, where N is the number of eigenmodes. For NSLS-II, N is 180. Such a large number of calculation needs to be finished within the 100us FOFB timing budget. This is a challenge for NSLS-II FOFB system.

We solved the challenge by using hardware for the FOFB calculation. Instead of using a general CPU to do the eigenspace compensation, we used FPGAs for the calculation. Figure 4 shows that most of the calculation is in the decoupling process, where each row of the U^T matrix multiplies the orbit difference. We noticed that this process can be done in parallel. Compared to serial computing using a CPU, FPGAs are well known for their parallel computation capability.

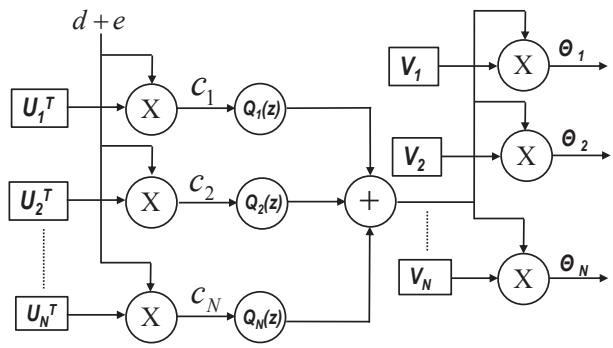


Figure 4: FOFB calculations with FPGA.

Figure 4 shows the basic calculation unit of FOFB inside FPGA. It includes the parallel decoupling in eigenspace together with the corrector setpoint calculation in parallel. Floating point calculation is used to make the calculation more accurate.

COMMISSIOING RESULTS

The NSLS-II FOFB system was commissioned and proved to meet the requirements of the orbit stability. Since June 2015, the FOFB system has been a regular tool to maintain beam stability.

Beam Stability Requirement

There are 30 double-bend achromat (DBA) cells in NSLS-II storage ring. The insertion devices are located at the straight sections of 15 high beta (long) and low beta (short) sections. Three damping wigglers are installed to reduce the bare lattice horizontal emittance from 2nm.rad

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

to 0.9nm.rad. Figure 5 shows the RMS beam size in one super cell.

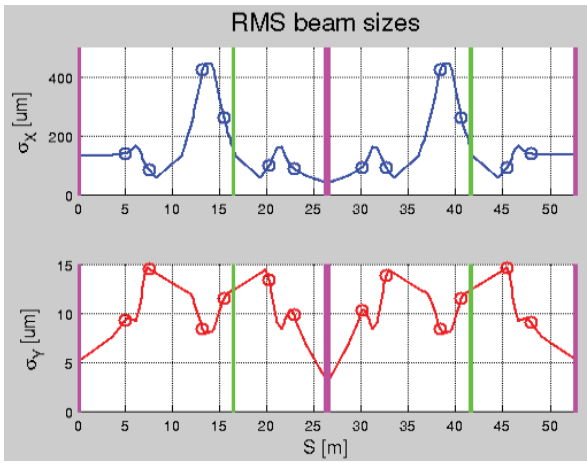


Figure 5: RMS beam size in one supercell using emittance of $\varepsilon_x/\varepsilon_y$ = 0.9/0.008 nm.rad and energy spread of 0.09%.

The photon beamline source points are marked with green and pink vertical lines. The beam stability requirement for FOFB is that the beam movement is less than 10% of beam size. One can see that the most stringent requirement is to keep beam motion less than 300nm at s value about 27m, where the RMS beam size is 3us.

Beam Stability Achievement

Figures 6 and 7 show the beam stability with FOFB on and off. Figure 6 is the power spectrum density (PSD) plot for FOFB on and off cases. The power spectrum of each BPM is calculated from 10 second synchronized 10kHz fast acquisition data. Different BPMs at different locations have different spectra. Figure 6 shows the average spectrum for all non-dispersion BPMs. There are three panes in Figure 6: the upper one shows the spectrum for the horizontal and vertical planes, the middle one shows the integrated spectrum where the integration starts from low frequency and the lower pane shows the integrated spectrum where the integration starts from high frequency. Figure 6 shows that the FOFB system is able to supress noises up to 250Hz in the vertical plane and up to 50Hz in horizontal plane. During the commissioning, most of the effort was spent on vertical plane noise suppression. Feedback parameters can be further optimized in the future developments.

Figure 7 shows the RMS motion of 12 BPMs in one supercell (C02-C03) for both the horizontal and vertical planes. The data is calculated from the integration of PSD from 1 Hz to 500Hz. The dashed lines are drawn to indicate 1% of horizontal beam size and 10% of vertical beam size. In the horizontal plane, the RMS motion is suppressed within 1% of beam size when FOFB is on. This is well within the requirement. In the vertical plane, when FOFB is off, the beam motion is about 20% of beam size. With FOFB on, the beam motion is suppressed down to about 5% of beam motion, which is also within the 10% of requirement. The HXN (Hard X-ray Nano Probe) beamline has the most stringent requirement for orbit stability. C03 IVU, which locates in the middle waist, is the insertion device for the HXN beamline. From the two BPM readings at the two ends of the insertion device, position and angle spectrum at the IVU center can be calculated. In the vertical plane, the integrated (from 1Hz to 500Hz) position and angle motions are 0.216 μm and 0.136 μrad, within the 10% beam size and 10% of beam angle divergence requirement.

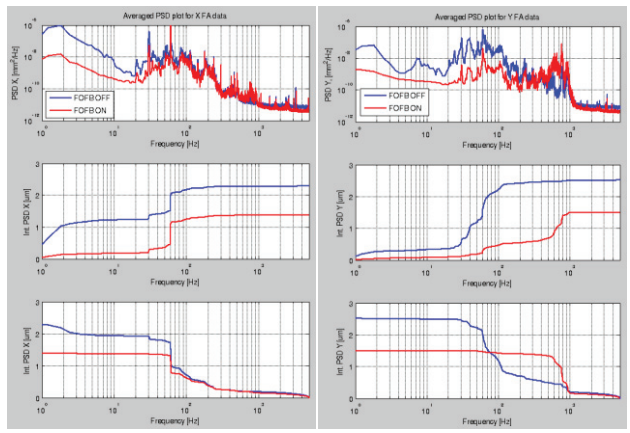


Figure 6: BPM 10kHz FA data PSD spectrum.

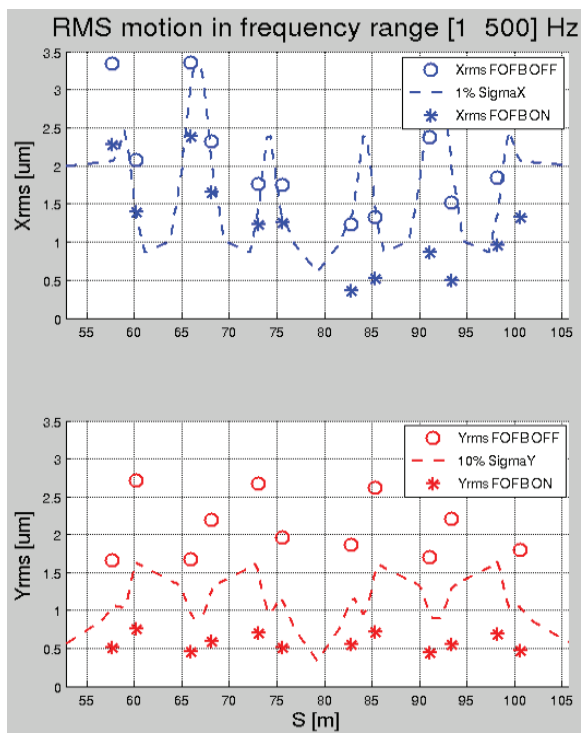


Figure 7: Integrated RMS motion(1-500Hz) for 12 BPMs in one supercell (C02-C03).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DISCUSSIONS

FOFB on During Injection

Before the top-off operation, NSLS-II had been operated with regular injection. Usually the beam was injected when the stored beam density drops to some level. During our beam study, we found that the beam orbit change during beam injection was small. This is due to the relatively small injection beam charge and the matched optics design between the storage beam and injection beam. The small disturbance allowed FOFB to stay on during injection.

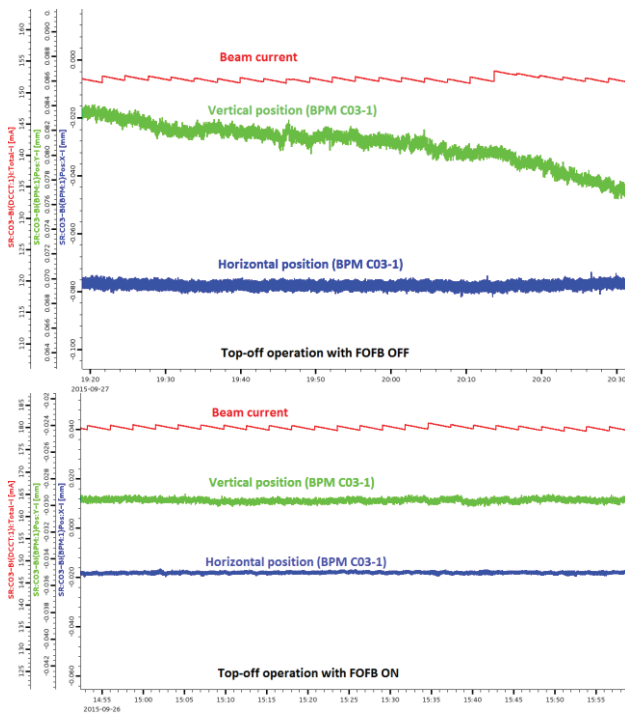


Figure 8: BPM position (10Hz data) during top-off operation with FOFB off (top) and FOFB on (bottom)

NSLS-II top-off operation was commissioned in September 2015. Figure 8 shows the BPM 10Hz reading during top-off operation with FOFB off and FOFB on. It shows the noise suppression effect of FOFB system. With FOFB on, the horizontal and vertical orbit disturbance during beam injection is negligible. Starting from October 1, 2015, NSLS-II is running with both FOFB on and top-off mode to provide users a stable beam current and stable beam orbit.

FOFB on with Insertion Device(ID) Cap change

Light source users regularly change the insertion gaps. The change brings some orbit disturbance. To keep the orbit disturbance small, lookup tables are carefully designed to compensate the disturbance caused by ID gap changes.

During commissioning, we verified that if ID lookup tables are designed well, the leaking orbit disturbance is small and the FOFB can compensate such disturbances. One exception is the damping wiggler. The movement of the damping wiggler causes large orbit change and FOFB can't easily compensate it. Fortunately, the change of damping wiggler doesn't happen often. The FOFB can be turned off before the movement, and turned on after the damping wiggler movement.

FOFB with Local Bump Creation

During the NSLS-II beam line commissioning stage, beam line users are allowed to create a local bump for a specific angle and offset at ID center. A high level application has been designed to create such local bump using multiple local slow correctors. If FOFB stays on during local bump creation, the two systems fight with each other: FOFB is trying to keep the orbit to its own reference orbit and the local bump program is trying to create a new orbit. We have tested several methods to solve this problem. The principle is to quickly change the reference orbit of the FOFB system. We are in the process of testing different methods to change the FOFB reference orbit.

SUMMARY

This paper discussed the architecture, the calculation algorithm and the commissioning results of the NSLS-II FOFB system. It shows some of the novel approaches of the fast orbit feedback system: two tier communication protocol, eigenspace compensation, and FOFB calculation using FPGA. These novel approaches provide flexibility for FOFB control and they demonstrate the performance required to meet all NSLS-II sub-micron beam stability design goals with noise suppression up to 250Hz in the vertical plane. The FOFB system is commissioned and used as a regular tool for user operation. We also discussed various issues during FOFB commissioning.

ACKNOWLEDGEMENTS

Many critical subsystems are required to work well to make the FOFB system work. We thank O. Singh, K. Vetter, J. Mead, A. Dellapenna and Y. Hu for their efforts of BPM development. We also thank G. Ganetis, W. Louie and J. Ricciardelli for fast corrector power development.

REFERENCES

[1] S. Ozaki, et.al, "Philosophy for NSLS-II Design with Sub-nanometer Horizontal Emittance", Proceedings PAC07, p.77 (2007).

[2] L. Yu, "Performance Calculation on Orbit Feedback for NSLS-II", Proceedings PAC05, p.1036 (2005).

[3] M.G.Abbott, et.al, "Performance and Future Development of the Diamond Fast Orbit Feedback System", Proceedings of EPAC08, p.3257 (2008).

[4] Y. Tian, et.al, "NSLS-II Fast Orbit Feedback with Individual Eigenmode Compensation", Proceedings PAC11, p.1488 (2011).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

THE LASER MEGAJOULE FACILITY: THE COMPUTATIONAL SYSTEM PARC

S. Vermersch, J-P. Airiau, V Higonencq, J Pascal, X Julien, E Bordenave CEA, CESTA, F-33116 Le Barp, France
C. Lacombe, L. Burgy, J. Terral, A. Lemaire, Thales Services, Mérignac, France

Abstract

The Laser MegaJoule (LMJ) is a 176-beam laser facility, located at the CEA CESTA Laboratory near Bordeaux (France). It is designed to deliver about 1.4 MJ of energy to targets, for high energy density physics experiments, including fusion experiments. The assembly of the first line of amplification (8 beams) was achieved in October 2014.

A computational system, named PARC, has been developed and is under deployment to automate the laser setup process, and accurately predict laser energy and temporal shape. PARC integrates the simulation software MIRO. For each shot on the LMJ, PARC determines the characteristics of the injection laser system required to achieve the desired main laser output, checks the machine fine-tuning for equipment protection, determines the required diagnostic setup, and supplies post-shot data analysis and reporting.

This document gives the first results obtained with PARC. We also describe results obtained with the PARC demonstrator during the first experience on the LMJ facility.

PARC FUNCTIONALITIES

PARC must guarantee the laser performance according to a shot request. To achieve this goal, PARC is based on the simulation software MIRO which models each laser beam. PARC manages and updates one data model for each beam and validates them at the end of each shot.

Contribution in Experiment's Chronology

The LMJ monitoring system orchestrates the whole sequence of operations. It invokes PARC at each step of the sequence to deal with specific process.

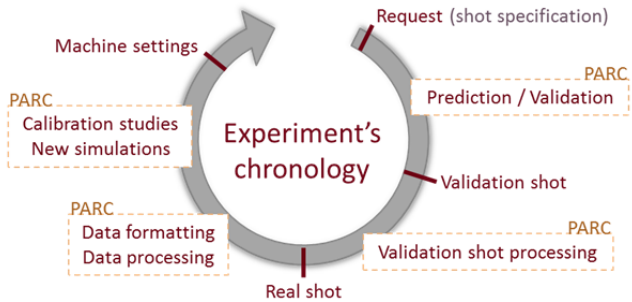


Figure 1: PARC in experiment's chronology.

The first step in the sequence is the production of a request that identifies all the desired laser output characteristics.

Predictive simulation is the first set of operations in the sequence chronology. It determines the LMJ equipment configuration and settings required to achieve the desired main laser output. The behaviour is performed by the software MIRO.

Validation is performed to check the ability of the machine to perform the shot in security and safety. Validation is also based on a simulation realized by MIRO.

Once the prediction and the validation are accepted, the sequence is composed of one or more validation shots to fine-tune data and equipment settings. A validation shot is like a real shot without the main amplifiers.

Data-processing of validation shot follows each validation shot. PARC performs data processing and provides parameters to check if the settings are compatible with the real shot. We also check that equipment's and diagnostics are operating normally.

The next step in the sequence is the real shot.

Data-processing of the real shot consists of supplying post-shot data analysis and reporting. Among other things, PARC compares the measurements to the request, to the results of simulations, performs new MIRO simulations from shots results to fine-tune calibration.

Calibration of computation models is performed at the end of each shot to correct the drift. The calibration uses the results and reports of several shots.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Autonomous Working

PARC provides an autonomous mode that allows operators to perform simulations and processing apart from the shot sequence.

Operators can prepare new shot campaigns and check the ability of a request thanks to dedicated prediction and validation algorithms.

They can calibrate MIRO simulations and PARC setting with specific data-processing codes from shots results.

PARC STAKES

PARC Position

Interfaces are a major issue in PARC. PARC has three main interfaces (Figure **2**): the LMJ monitoring system, the users and the MIRO software.

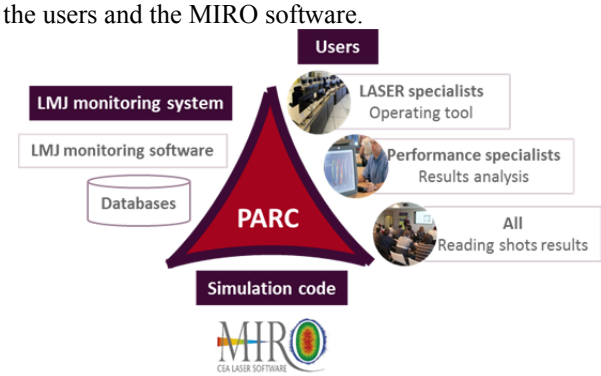


Figure 2 : PARC interfaces.

The LMJ monitoring system software (also called Supervisory system) invokes PARC, which provides reports in return. The LMJ monitoring system relies on several databases where raw data, shots results, settings and characteristics of the equipment are stored. PARC extracts and translates these data into its own formalism. The data are hence civilized to be used in processing and simulations.

Several types of users are interested in PARC functionalities.

Laser specialists, the main users, use PARC as an operating tool. They execute new simulations, study the results and reports, integrate new algorithms and modify existing codes and configuration data.

Performance specialists export the results processed by PARC to perform more analysis on separated systems.

All users want to see the reports on separated systems to show them in meetings for example.

The Miro software simulates the Laser propagation with numerical models. PARC prepares data and simulation characteristics, executes MIRO and reads and processes the results.

LMJ Structure and Variables

The particularities of LMJ structure have an important part in PARC algorithm structure. The LMJ is characterized by:

- 4 sections (Figure 3): pre-amplification, amplification, conversion and target,

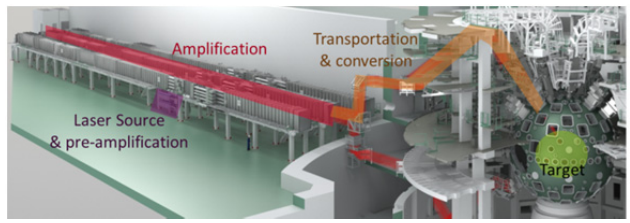


Figure 3 : LMJ sections.

- 4 levels of granularity: beam, quad (4 beams), line (2 quads) and the whole system,
- 3 main characteristic variables: energy, temporal shape and spatial shape,
- 3 levels of data: prediction results, request, measurement results.

These four components form four dimensions in the architecture of PARC processing codes.

ARCHITECTURE

Hardware Architecture

PARC is a complete ecosystem of tools and technologies working together to achieve its objectives. The hardware platform consists in three components.

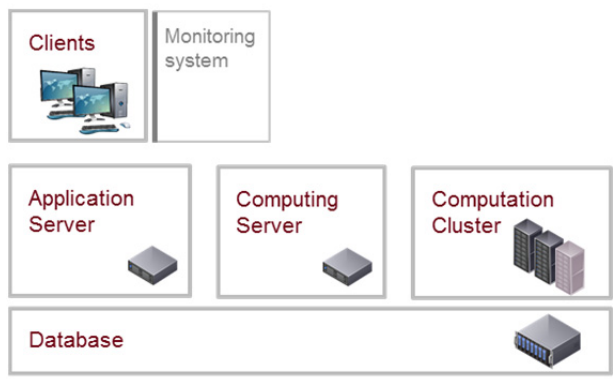


Figure 4 : Hardware architecture.

The Application Server is a server accepting connections from the various clients,

The Computing Server is a server orchestrating algorithm executions. It distributes tasks among the cluster nodes,

The Computation Cluster is a cluster of nodes performing the high-demand computation tasks such as MIRO beam propagation code. The modular structure of the cluster in racks allows high flexibility to gradually add laser lines with their installation and high performance regardless of the number of laser beams (a prediction simulation takes about 20 minutes).

These hardware components access the same distributed file system to share the data. The data are previously converted by a dedicated software civilizing

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

the various data types (file formats, units) injecting these into the PARC database.

Software Architecture

PARC software architecture is threefold:

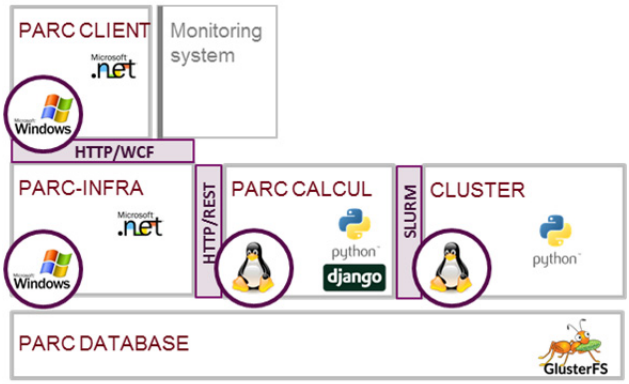


Figure 5 : Software architecture.

- PARC-INFRA: application server dealing with all the contextualization of scenario execution. Web application hosted by an IIS Web server, it exposes a set of WCF web services mainly to fine-tune data and initiate scenarios,
- PARC-CALCUL: Web Frontend to the computation cluster. Python Django application, it exposes a Rest web service interface to segregate the computing from the application management, running task orchestration, and perform task distribution,
- PyParc: Python library underpinning the execution of the elementary computation tasks.

FUNCTIONS COMPONENTS

Algorithm Components

In PARC, the algorithm associated with a function is named scenario. Scenarii are Python scripts based on two computation libraries.

The modules library is composed of elementary computation codes. We distinguish several kinds of modules:

- Basic calculation codes (two scalars relative comparison for example),
- Generic codes which implement generic functions as the execution of MIRO simulation for instance,
- Reading / writing data in PARC database,
- More complex codes which implement a part of a scenario algorithm. These modules manage data and elementary calculation codes.

The PyParc library is composed for generic functions which manage the scenario execution and provide tools for the implementation of modules. For instance, the distribution of modules is managed by PyParc functions.

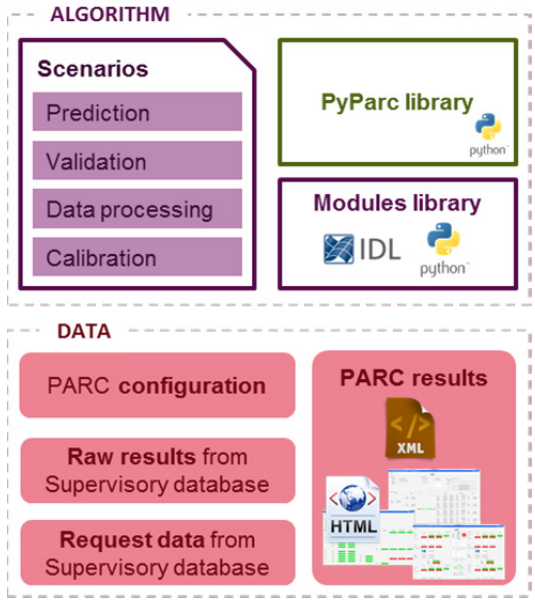


Figure 6 : PARC components.

Data Structure

PARC scenarios are run from different types of data files. PARC has its own configuration file including algorithm parameters and simulation data. But the main data is recovered from the monitoring system : the request data with equipment's settings and the raw results of the shots.

The scenarii results are stored in PARC database in XML (Extensible Markup Language) files and in the form of specific reports in HTML (Hypertext Markup Language).

PARC RESULTS

Prediction

Prediction algorithm performs simulations with MIRO to determine settings and equipment configuration for reaching the request. A report is generated at the end of the simulations using a specific GUI (Graphical User Interface). This report is composed of three synthetic views which gather the results. Each view corresponds with a level of granularity: system, quad and beam.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

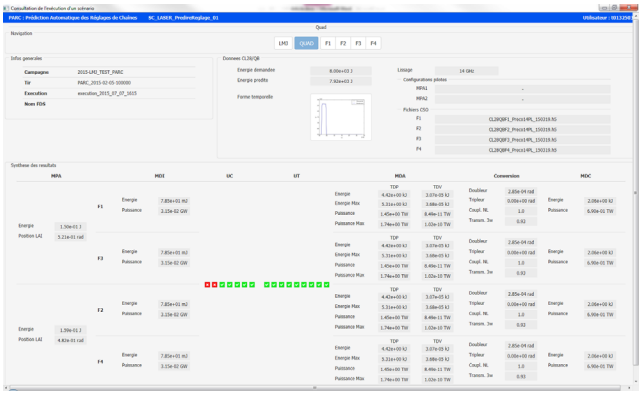


Figure 7 : Prediction quad results.

Figure 9 : Data processing quad results.

Validation

Validation algorithm checks the ability of equipment to perform the shot in security with the settings calculated by the prediction algorithm. A report is generated using a specific GUI. This report is composed of two synthetic views which gather the results for the system and for each beam.

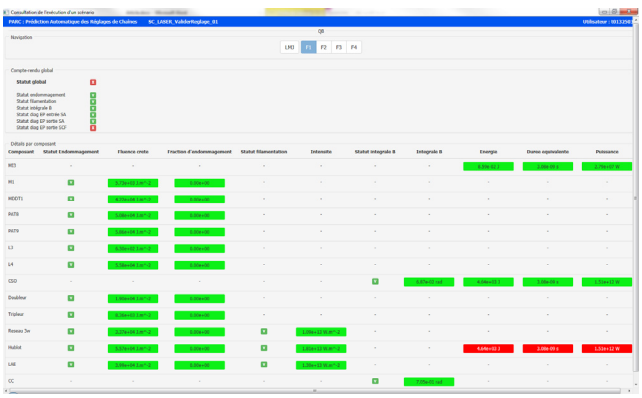


Figure 8 : Validation beam results.

Data Processing

Data processing algorithms perform calibration corrections, comparisons between measurements and prediction or between measurements and request, new simulations from the shot's results. A report is generated using a specific GUI. This report is composed by three synthetic views which gather the results. Each view corresponds with a level of granularity: system, quad and beam.

First Experiment

We use the PARC demonstrator during the first experiment on the LMJ facility. This tool uses computation codes which are now integrated in the modules library of PARC. The Quad Energy for the October 17th 2014 shot is illustrated on the figure bellow. The request was 10kJ at 3ω for each quadruplet. We measure respectively 10,2 et 9,8 kJ.

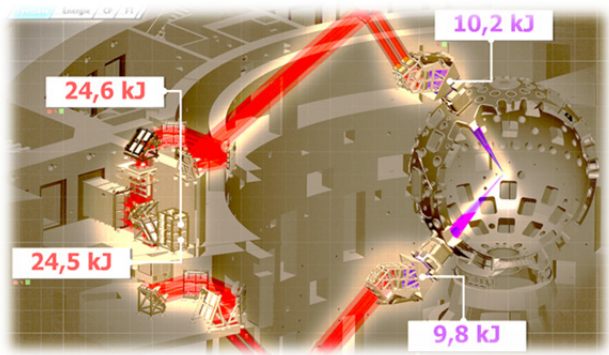


Figure 10 : First experiment.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

LOW LEVEL RF CONTROL IMPLEMENTATION AND SIMULTANEOUS OPERATION OF TWO FEL UNDULATOR BEAMLINES AT FLASH

V.Ayvazyan*, S.Ackermann, J.Branlard, B.Faatz, M.Grecki, O.Hensler, S.Pfeiffer, H.Schlarb, C.Schmidt, M.Scholz, S.Schreiber, DESY, Hamburg, Germany
A.Piotrowski, Fast Logic, Lodz, Poland

Abstract

The Free-Electron Laser in Hamburg (FLASH) is a user facility delivering femtosecond short radiation pulses in the wavelength range between 4.2 and 52 nm using the SASE principle. The tests performed in the last few years have shown that two FLASH undulator beamlines can deliver FEL radiation simultaneously to users with a large variety of parameters such as radiation wavelength, pulse duration, intra-bunch spacing etc. FLASH has two injector lasers on the cathode of the gun to deliver different bunch trains with different charges, needed for different bunch lengths. Because the compression settings depend on the charge of bunches, the low level RF (LLRF) system needs to be able to supply different compression for both beamlines. The functionality of the controller has been extended to provide intra-pulse amplitude and phase changes while maintaining the RF field amplitude and the phase stability requirements. The RF parameter adjustment and tuning for RF gun and accelerating modules can be done independently for both laser systems. Having different amplitudes and phases within the RF pulse in several RF stations simultaneous lasing of both systems has been demonstrated.

INTRODUCTION

FLASH has been in operation as a user facility since summer 2005 [1]. In the mean time as a test facility, FLASH is in use for testing the superconducting accelerator technology for European XFEL [2] and ILC [3] projects. The first effort to operate accelerating modules at different gradients with alternating RF pulses was motivated by the ILC and European XFEL and initially has been demonstrated in year 2008 [4].

FLASH2 [5] is the second undulator beam line with variable gap built in a separate tunnel. It will make full use of the existing FLASH accelerator. Part of the bunch trains are kicked from the main beamline (FLASH1) into the new undulator beamline. In order to double the beamtime, users of both beamlines would need the 10 Hz repetition rate. A fast kicker in combination with a DC septum is used to deflect the beam into the second undulator line. In addition, the large variety in beam parameters should be available at both beamlines independently in order to ensure a maximum flexibility in planning of the beamtime. For this reason two cathode lasers are in use, each with its own bunch train. A variable delay between the two lasers within the RF pulse gun and accelerating modules ensures that users from both beamlines get their own set of parameters.

* valeri.ayvazyan@desy.de

In addition, the start time of the kicker pulse is shifted with respect to the start time of the laser pulse and the RF amplitude and phase of the gun and each of the modules can be tuned for optimal conditions for both users. During the initial tests it was shown that the RF system is able to handle the flexibility needed to compress the beam independently for FLASH1 and FLASH2 beamlines.

LOW LEVEL RF CONTROL FOR MULTI-BEAMLINES

RF Control Specifics for Multi-beamline Case

During the multi-beam line operation, within limits given by the beam delivery system, the bunch pattern and beam energy should be adjusted independently for each beam line, suggesting a time-sliced operation. For the FLASH2 beamline RF amplitude and phase changes within the pulse are required within a short time (less than 50μs). Different beam loadings are foreseen for different beamlines. Particularly the ability of gradient tuning of the last two RF stations is needed for wavelength scans for the FLASH1 beamline and the ability of phase tuning at injector for variation in compression at FLASH1 and FLASH2. From the operation point of view, the most important requirement is to have the ability of independent RF operational parameters adjustments for both beamlines.

LLRF System Overview

The layout of the FLASH facility including the new beamline is described in [6] and shown in Fig. 1. Its accelerator comprises a normal conducting RF gun, a first 8-cavity cryomodule, a third harmonic cryomodule with 4 cavities, a first bunch compressor, a second accelerating RF station (16 cavities), a second bunch compressor, and another two RF stations, with 16 cavities each. FLASH is operated in pulsed mode with repetition rate 10Hz. RF pulse duration is 1.3ms, 500μs for filling and 800μs flattop (beam acceleration). The RF power coming from 10MW klystrons is equally distributed to all cavities through the waveguide distribution system. The current stage of the LLRF control is implemented based on the MTCA.4 system [7]. The goal of the system is to control the accelerating gradient in amplitude and phase for each RF station based on vector sum control of cavity gradients [8]. The main components of the LLRF system are depicted in Fig. 2. For every cavity, the forward (PFWD), reflected (PREF) and transmitted signals or probes (PRB) are first down-converted to an intermediate frequency (IF) by the down-converters (uDWC) and then digitized

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

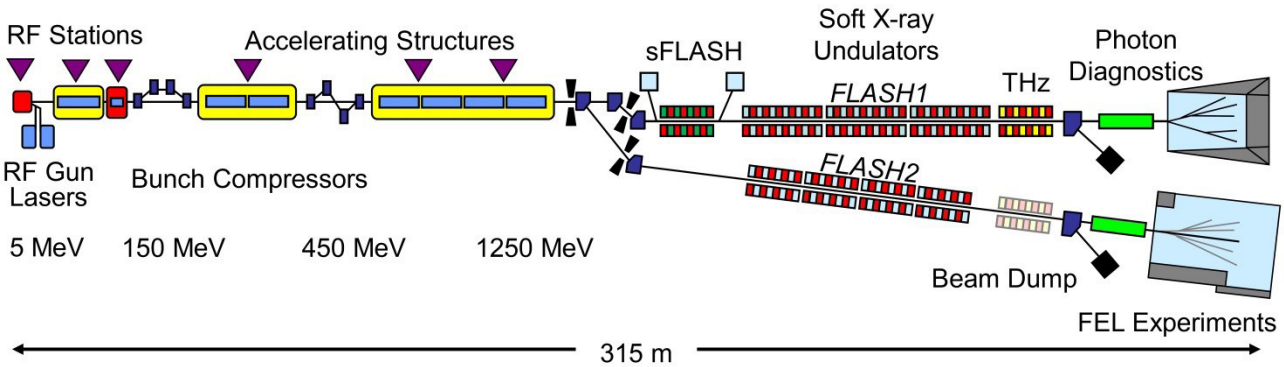


Figure 1: Schematic layout of the FLASH facility. The electron gun is on the left, the experimental hall on the right. Behind the last accelerating module, the beam is switched between FLASH1 and FLASH2 beamlines.

(uADC). The sampled signals are pre-processed by the uADC and then sent over the MTCA.4 backplane to the main LLRF controller (uTC) which performs all control computations. The uTC then generates the drive signal which is up-converted to RF frequency by the vector modulator (uVM). The LLRF drive signal is pre-amplified and then sent to the klystron (KLY). The master oscillator (MO) provides the 1.3 GHz reference signal (RF), required by the local oscillator generation module (LOGM) to generate the LO and clocks (CLK) signals used by the down-converters, and to distribute the reference signal to the uVM. The power supply module (PSM) provides DC voltages to external modules. Finally, the piezo driver module (PZ16M) digitizes the piezo sensor data and drives the piezo actuator for detuning and microphonics compensation. Communication between the PZ16M and the MTCA.4 system is performed through an optical link to the main LLRF controller, the uTC. This setup can be easily extended to control the sum of 16 cavities.

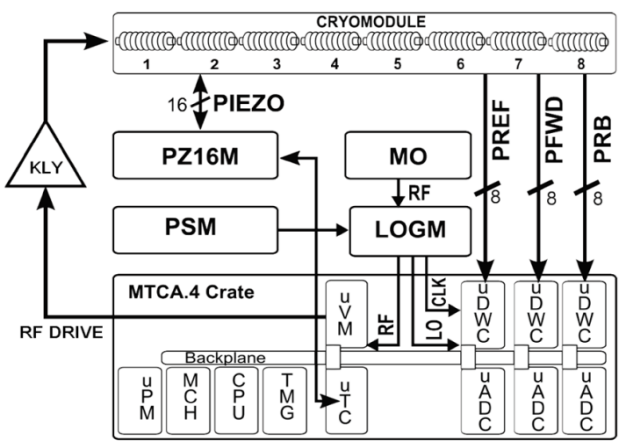


Figure 2: LLRF system block diagram for one cryogenic module.

Brief Description of Control Algorithm

The feedback algorithm is implemented in FPGAs firmware and DOOCS [9] control system server. The control algorithm employs tables for feed-forward, set-point and feedback gain settings to allow time varying of those parameters. High frequency probe signals are used to measure the accelerating field in the individual cavities. These 1.3 GHz (3.9 GHz) signals are down-converted to 54 MHz and sampled by uADCs at 81.25 MHz. The digitized signals are going to the digital field detector which extracts the real and imaginary parts of the cavity field vectors from the input stream. The resulting field vector of each cavity is multiplied by a rotation matrix to calibrate amplitude and phases. Then the sum of individual field vectors is calculated and rotated to adjust the loop phase. The vector sum of the cavities 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 is used to scale the feed-forward table to provide fast feed-forward corrections if the beam current varies. The controller server software handles: generation of control tables from basic settings, rotation matrices for the cavity field vectors, start-up configuration files, feedback and exception handling control parameters, etc. The interrupt service routines are used to start the data reading from the controller boards. The parameters of the feedback algorithm are modified by the FPGA programs in the time slot between pulses.

Application Software

A set of generic and especially devoted programs provides the tools for the operators to control the RF system. Some of them are created based on the MATLAB, others as 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 vector sum calibration, automated operation of the frequency tuners, phasing of cavities, adjustment of various control system parameters, etc. Extensive diagnostics inform the operator about cavity quenches, cavity detuning, and an excessive increase in control power.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

RF control functionality extension for FLASH2

The RF pulse is shared between the electron bunch trains for FLASH1 and FLASH2. For the full RF pulse length, the total maximum number of bunches, with bunch repetition rate of 1 MHz is 800. The bunch pattern (number of bunches and intra-train repetition rate) and bunch charge can be different for FLASH1 and FLASH2, which is realized by using two independent injector lasers. Timing events from FLASH accelerator are used to synchronize all accelerator subsystems and are managed by DOOCS timing server. Programmable timers are triggered by these events to generate the start pulses for the klystrons, FPGAs or uADCs. A timer unit provides several independent output channels. Some machine parameters that change from macro pulse to macro pulse are delivered to run all digital feedback loops in parallel. This information is available for the LLRF controller server. The control tables are generated through the LLRF library from the operational setting according to the provided timing information. Parameters such as beam start time, number of bunches, bunch repetition rates are extracted from the timing server for both beamlines. Other parameters like RF transition time, offsets with respect to beam pulse, etc. are adjustable. LLRF control software follows any changes of the timing parameters. Fig. 3 illustrates the timing setup for two beamlines.

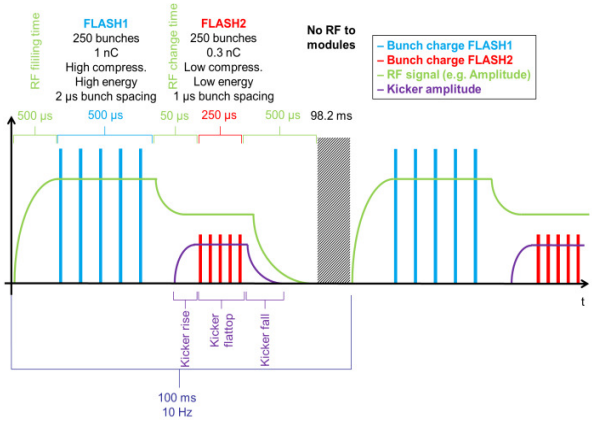


Figure 3: An example of FLASH timing settings.

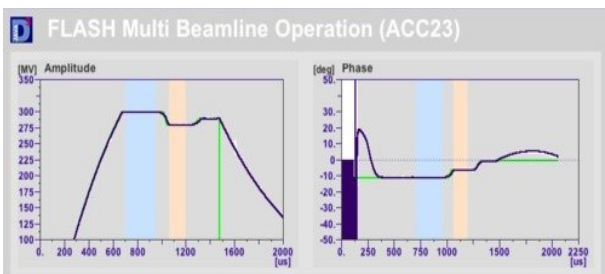


Figure 4: Energy and phase variation within RF pulse.

Energy and phase variations are possible to allow for charge dependent compression and wavelength fine tuning. A future extension with an additional beamline is already foreseen, as can be seen by the three different levels of RF pulse (Fig. 4). Background colours show the beam start time and bunch train duration time for FLASH1 and FLASH2 beamlines. In order to avoid operational settings changes in unacceptable ranges, limits for FLASH2 operational parameters with respect to FLASH1 are implemented as well. An adaptive learning algorithm which minimizes repetitive control errors tends to limit these transition changes. In order to avoid oscillations, it is possible to deactivate adaptations for certain regions.

PERFORMANCE OF SIMULTANEOUS OPERATION OF FLASH1 AND FLASH2

In the initial stage of the project several tests have been performed to show that simultaneous operation of both beamlines is possible [10]. Fig. 5 shows lasing of two bunch trains which were generated with separate injector lasers, separated by 80µs: a) with the same charge of about 0.5 nC and b) with a factor of 2 difference in charge, e.g. of 0.5 nC for the first and 0.25 nC for the second bunch train. The blue line indicates the actual SASE pulse energy produced by each individual electron bunch in the macro-pulse. The green line is the time average of this signal. The yellow line indicates the maximum SASE level which occurred since the measurement was started. Both bunch trains have a repetition rate of 1 MHz. The number of bunches in this case was 30 and 20 respectively. During this test only RF parameters were changed within the RF pulse and orbit was adjusted behind the FLASH2 extraction point. Because the FLASH2 beam line was under construction in that time, this test has been performed at FLASH1 beamline. Careful adjustment was done to make sure that both injector lasers hit the cathode under the same angle to make sure that the electron beams have the same trajectory. This condition was relaxed in the later situation, since the orbit in the FLASH1 and FLASH2 undulators can be adjusted independently to optimize lasing.

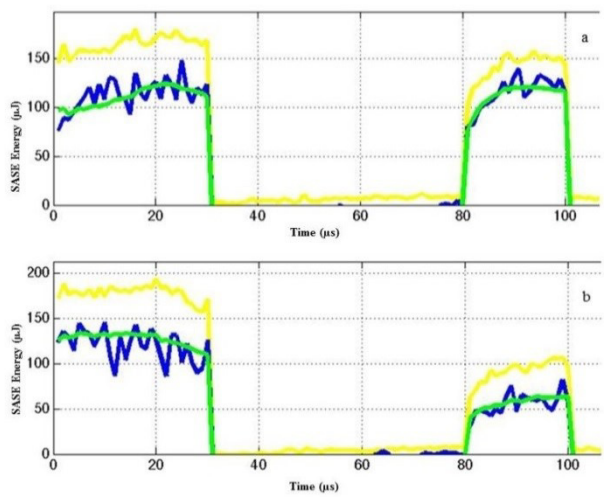


Figure 5: Lasing of two independent bunch trains with different charges and a variable delay in time.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Since 2014 tests have been performed with FLASH1 and FLASH2 in operation. First SASE operation in FLASH2 was achieved at wavelength 40 nm on August 20, 2014 [11] during FLASH1 operation with 250 bunches at 13.5 nm.

After the demonstration of the first lasing at FLASH2 the SASE operation was established at various wavelengths. So far, the maximum number of bunches per burst during a parallel SASE operation of both beamlines has been 400 bunches in FLASH1 and 30 bunches in FLASH2, both with a bunch repetition rate of 1 MHz. Fig. 6 shows the SASE pulse energy along the bunch trains in FLASH1 and FLASH2. During parallel operation achieved pulse energy at FLASH1 was about 200µJ and about 100µJ at FLASH2 in the period from June 2015 to August 2015.

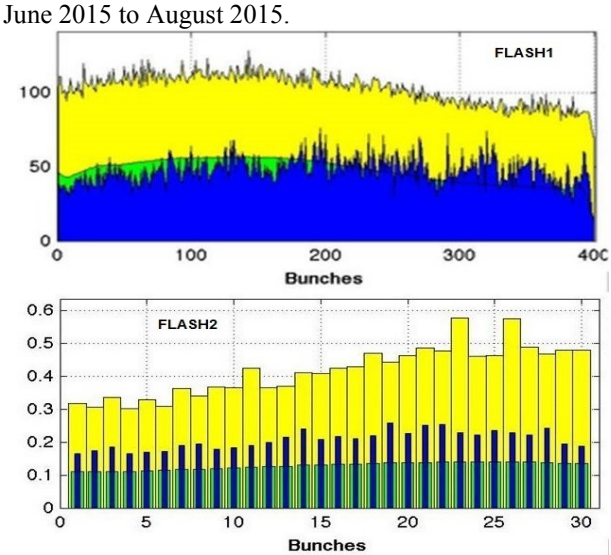


Figure 6 : SASE pulse energy per bunch (in a.u.). Top: 400 bunches in FLASH1. Bottom: 30 bunches in FLASH2. Blue: actual value, green: average, yellow: maximum.

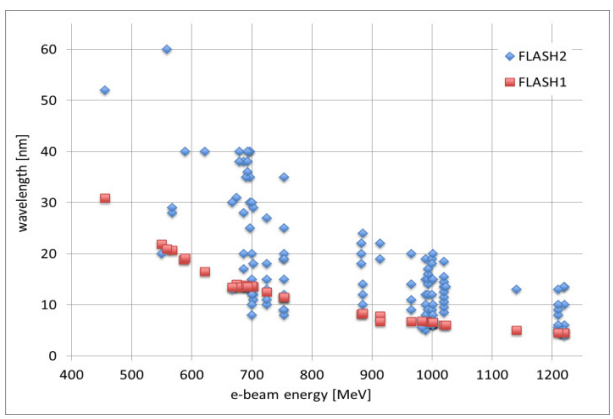


Figure 7: Photon wavelengths achieved in the FLASH1 and FLASH2 beamlines during parallel SASE operation.

In Fig. 7 [12] are shown photon wavelengths achieved in the FLASH1 and FLASH2 beamlines during parallel SASE operation in the period from August 2014 to August 2015. The photon wavelength in FLASH1 is determined - due to the fixed gap undulator - by the electron beam energy. Variable gap undulators in FLASH2 allow different photon wavelength at fixed beam energies.

SUMMARY AND OUTLOOK

Low level RF control functionality has been extended which makes simultaneous RF operation of multi-beamlines possible. Changes in RF settings within RF pulse can be achieved which allow different compression settings for different charges (and therefore different bunch lengths) while maintaining the RF field amplitude and the phase stability requirements.

Simultaneous operation of two beamlines and lasing of FLASH2 at the wavelength 40 nm was achieved, while FLASH1 was lasing simultaneously with multi-bunch mode at different wavelength (13.5 nm). SASE operation at various wavelengths was established.

Gained experience with simultaneous operation of two beamlines at FLASH is a good basis for successful commissioning of multi-beamline facility European XFEL which is foreseen for the second half of 2016. At XFEL, operation of alternating RF pulses (from pulse to pulse) is foreseen as well. This operation mode requires additional changes in the firmware/software structure, e.g. extending the amount of control tables. Furthermore, requirements to the timing system are increased to reliably trigger the correct mode of operation.

REFERENCES

[1] V.Ayvazyan et al., Eur. Phys. J. D 37 (2006) 297-303.
[2] http://www.xfel.eu
[3] https://www.linearcollider.org/ILC
[4] V.Ayvazyan et al., “Alternating Gradient Operation of Accelerating Modules at FLASH”, TUPP001, EPAC’08, Genoa, Italy (2008).
[5] B.Faatz et al., “FLASH II: Perspectives and chalanges”, Nucl. Instr. Meth. A 635, S2 (2011).
[6] M.Vogt et al., “Status of the Soft X-ray Free Electron Laser FLASH”, TUPWA033, IPAC’15, Richmond, USA (2015).
[7] J.Branlard et al., “Equipping FLASH with MTCA.4-based LLRF system”, THP085, SRF’13, Paris, France (2013).
[8] T.Schilcher, “Vector Sum Control of Pulsed Accelerating Fields in Lorentz Force Detuned Superconducting Cavities”, PhD Dissertation, DESY Hamburg, Germany, 1998.
[9] http://doocs.desy.de
[10] S.Ackermann et al., “Simultaneous Operation of Tow Undulator Beamlines FEL Facility”, TUPD32, FEL’12, Nara, Japan (2012).
[11] S.Schreiber, B.Faatz, “First Lasing at FLASH2”, MOA03, FEL’14, Basel, Switzerland (2014).
[12] M.Scholz, B.Faatz, S.Schreiber, J.Zemella, “First Simultaneous Operation of Tow SASE Beamlines in FLASH”, TUA04, FEL’15, Daejeon, Korea (2015).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

BAYESIAN RELIABILITY MODEL FOR BEAM PERMIT SYSTEM OF RHIC AT BNL*

P. Chitnis#, Stony Brook University, NY, USA
K. A. Brown, Brookhaven National Laboratory, NY, USA

Abstract

Bayesian Analysis provides a statistical framework for updating prior knowledge as observational evidence is acquired. It can handle complex and realistic models with flexibility. The Beam Permit System (BPS) of RHIC plays a key role in safeguarding against the faults occurring in the collider, hence directly impacts RHIC availability. Earlier a multistate reliability model [1] was developed to study the failure characteristics of the BPS that incorporated manufacturer and military handbook data. Over the course of its 15 years of operation, RHIC has brought forth operational failure data. This work aims towards the integration of earlier reliability calculations with operational failure data using Bayesian analysis. This paper discusses the Bayesian inference of the BPS reliability using a two-parameter Weibull survival model, with unknown scale and shape parameters. As the joint posterior distribution for Weibull with both parameters unknown is analytically intractable, the Markov Chain Monte Carlo methodology with Metropolis-Hastings algorithm is used to obtain the inference. Selection criteria for the Weibull distribution, prior density and hyperparameters are also discussed.

INTRODUCTION

The Beam Permit System (BPS) of Relativistic Heavy Ion Collider (RHIC) monitors the health of RHIC subsystems and takes active decisions regarding beam-abort and magnet power dump, upon a subsystem failure. The reliability of BPS thus directly impacts the RHIC downtime, and hence its availability. A Monte Carlo reliability model based on exponential competing risks was developed for BPS previously [1]. This model simulated the progression of basic component failures to system level catastrophic events. This work together with a quantitative fault tree analysis [2] helped characterize the failure rate and structural importance of each basic component of the BPS. RHIC has been operational for 15 years, and has gathered hardware failure data over this time period. This data represents the actual BPS failure attributes from a top level perspective. Bayesian analysis is a good candidate for combining these two information sources to get a combined inference about the system failure characteristics.

BAYESIAN PARADIGM

Bayesian statistics is branch of mathematics that deals with updating current knowledge about a system or process when new information is acquired. Statistical analysis follows two major approaches, namely frequentist and Bayesian. In the widely used frequentist approach, the probability distribution of an event is calculated by observing its occurrence over a large period of time, and the distribution parameters are assumed to be constant over time. In contrast, Bayesian approach keeps updating the probability distribution as new data arrives. The parameters of the distribution are treated as random variables that are modified according to the new information gathered. This becomes quite important when there are two sources of information about a system or process that indicate different results, and both sources hold significance to the inference.

The underlying framework for Bayesian analysis is Bayes theorem. Bayesian analysis involves the continuous form of the Bayes theorem [3], which is represented as

$$\pi(\theta|x) = \frac{L(\theta|x) \times \pi(\theta)}{f(x)}$$

The unknown parameter is θ, which defines the probability distribution of any process and is subject to change with the arrival of new information. Variable x is new source of information in the form of data observations. The term $\pi(\theta)$ is called as the prior distribution of θ, which can be elicited by using another parameter(s) called the *hyperparameter(s)*. $L(\theta|x)$ is the likelihood function for θ which is calculated by the gathering the new data. The term $\pi(\theta|x)$ is called the posterior distribution of θ, which is a combination of both prior and data likelihood function. $f(x)$ is the unconditional distribution of the variate x that acts as a normalizing factor in the equation. Because $f(x)$ is independent of θ:

$$\pi(\theta|x) \propto L(\theta|x) \times \pi(\theta) \quad (1)$$

This equation forms the foundation for Bayesian analysis discussed in this paper. We will discuss the selection of the prior distribution and data distribution (likelihood function) in subsequent sections.

PRELIMINARY ANALYSIS

Preliminary analysis is needed to find the suitability of Bayesian analysis to our problem and for choosing the distribution appropriate to the information sources we have. We analyze two the sources of information, the results from a Monte Carlo (MC) model [1] and the historical failure data obtained from the RHIC hardware maintenance records.

*Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy.
#prachi.chitnis@stonybrook.edu

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Source 1: Monte Carlo Results

The MC model defines the propagation of component failures to a system level, depending upon the states of other components and the structure of the system. As the component failures were taken from MIL-HDBK 217F [4] and the manufacturers' data, they provided point estimate for the hazard rate λ, with exponential survival distribution. These exponential failures might evolve as a different distribution on the system level. We combine the four system level failures modes in the MC model to a single failure for applying Bayesian analysis. Next we analyze the failure rate pattern of this single failure of BPS.

First, we check if the failure rate follows the Non Homogenous Poisson Process (NHPP) [5], for which specialized Bayesian analysis is needed. We plot the interval failure rates to see if it is time varying, and whether it follows a typical distribution. The number of failures in an interval of 10^7 hours is calculated, and the failure rate is plotted for each interval in Fig. 1. The rate is noisy but has a constant mean over the time. Thus we deduce that the MC model failure is not a NHPP.

Figure 1: Detection for NHPP failures.

Next, we find a suitable failure distribution function for the MC model. We run the simulation for 1.3E9 iterations and the total failures are recorded as point processes. The times between failures are fitted with exponential, Weibull and gamma distributions with the forms specified in [6] using MATLAB® [7]. The goodness of fit is estimated by Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) [8].

Table 1: MC Model Failure Distribution

Distribution	Para meter	Point estimate	AIC	BIC
Exponential	λ	8.831e-5	3141125.52	3141136.15
	α	1		
Weibull	λ	8.829e-5	3141127.47	3141148.72
	α	1.00046		
Gamma	λ	8.84e-5	3141127.44	3141148.66
	α	1.00106		

As seen in table 1, the AIC and BIC are smallest for the exponential distribution, which indicates that the exponential distribution is the best fit to overall system failure. The failure rate is λ and the shape parameter is α.

Although the military handbook is quite old (1995), its applicability to the BPS can justified as RHIC has been running since 1997, which was contemporary to the release of this version of military handbook

Source 2: Historical Failure Data

We analyze the past 15 years of hardware failure data of BPS, and select the system level failures that are similar to the ones analyzed by the MC model. We find overall 16 data points for the time between failures owing to high reliability of BPS. To analyze the distribution of this data, we fit Exponential, Weibull, Gamma and Lognormal distributions using MATLAB® and goodness of fit is estimated using AIC and BIC.

Table 2: Historical Failure Data Distribution

Distribution	Para meter	Point estimate	AIC	BIC
Exponential	λ	1.20E-04	322.9279	323.7005
	α	1		
Weibull	λ	0.000171	317.6438	319.189
	α	0.627457		
Gamma	λ	6.03E-05	318.1511	319.6963
	α	0.503074		
Lognormal	μ	7.7676	319.0121	320.5573
	σ	1.99141		

Looking at the table 2, we see that the AIC and BIC are now smallest for the Weibull distribution, asserting that the BPS has a Weibull survival distribution with decreasing failure rate. The historical data distribution represents the actual failure characteristics of the system, even if the MC model suggests that the system follows exponential survival distribution.

BAYESIAN RELIABILITY MODEL

The parameters of the posterior distribution reflect the tradeoff between the prior distribution and the data distribution (incorporated using likelihood function). This tradeoff level is determined by the relative strength of prior and data distributions. The influence of either can be changed by altering the hyperparameters. It is often desirable to choose a prior of a form such that the posterior distribution calculated is mathematically tractable. One of the techniques is to employ a "conjugate prior" that yields a posterior of the same form as the data distribution [3], but with different parameters. The posterior parameters specify the adjustment between prior and data.

For our Bayesian model, the prior information is an exponential distribution and the data is a Weibull distribution with shape parameter less than 1. We thus assume that the prior information is also a Weibull distribution with shape parameter equal to 1. Also scale parameters of both the information sources is different. Thus we need to choose a Bayesian model for Weibull distribution with both shape and scale parameters unknown. To implement this, we first need to choose a conjugate prior distribution that is suitable to the Weibull distribution of the data. There is no best way to define a prior distribution for Bayesian analysis. The following

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

sections outine the development of the Bayesian model step by step.

Data Distribution

Vidakovic [9] suggests a model for the Bayesian inference of Weibull distribution that is unknown in both shape and scale parameters. We follow this model throughout our analysis. Following is the Weibull distribution used by Vidakovic, with α as the shape parameter and $\eta^{-1/\alpha}$ as the scale parameter. Variable x is the times between failures data in years. This form does not have explicit posteriors for α and η. We will draw Bayesian inference for these two parameters.

$$f(x|\alpha,\eta) = \alpha\eta x^{\alpha-1}e^{-\eta x^{\alpha}}$$

The likelihood function is then equal to

$$L(\alpha,\eta|x) = \prod_{i}^{k} \alpha\eta x_i^{\alpha-1}e^{-\eta x_i^{\alpha}}$$

The α and η are treated as variables in the likelihood function, and have *[0, ∞)* support. The most probable values are given by

$$\alpha = 0.6275, \eta = 1.2904 \qquad (2)$$

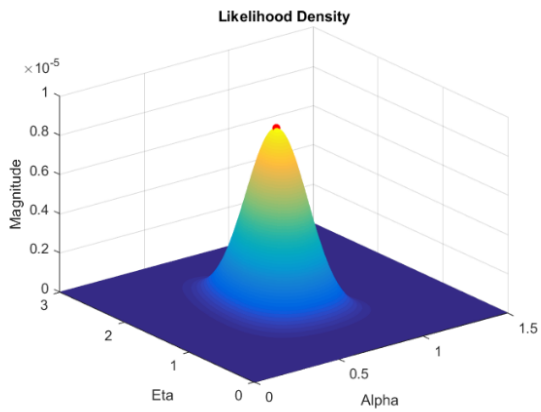


Figure 2: Likelihood function plot with α, η.

These values are calculated by transforming the parameters from the Weibull distribution of data from table 2. Fig. 2 plots the 3D likelihood function with α and η as variables. Note that α and η values from Eq. 2 correspond to the maximum likelihood point.

Conjugate Prior Distribution

A conjugate prior distribution is proposed for the Weibull distribution in [9]. This is a joint distribution for α and η, with a hyperparameter β.

$$p(\alpha,\eta) \propto e^{-\alpha}\eta^{\beta-1}e^{-\eta\beta} \qquad (3)$$

Note that we only use the kernel (omitting the proportionality constant) of the prior distribution. This is explained later in the posterior inference. The prior parameter λ (and α=1) representing the exponential distribution (Weibull with shape as 1) in table 1 are converted to α and η as

$$\alpha = 1, \eta = 0.7741 \qquad (4)$$

These are the point estimates. We need to define the joint distribution of α and η so that it best represents our beliefs about the prior information. We choose the hyperparameter β equal to 3. The reason for choosing β is explained by the figures below. The 3D prior density from Eq. 3 is plotted with α and η as variables in Fig. 3. Note that the magnitude on the plot is unnormalized. The point estimates from Eq. 4 are plotted as a red dot on the same figure. Note that this point does not correspond to the peak of magnitude in the plot.

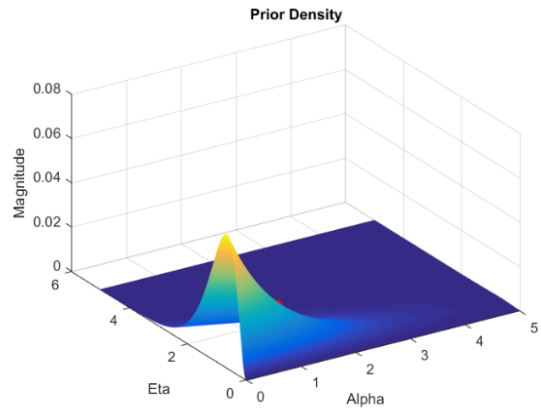


Figure 3: Prior density plot with α, η.

To get a clear picture we plot the 2D prior density for η with constant $\alpha = 1$ in Fig. 4a and the 2D prior density for α with constant $\eta = 0.7741$ in Fig. 4b.

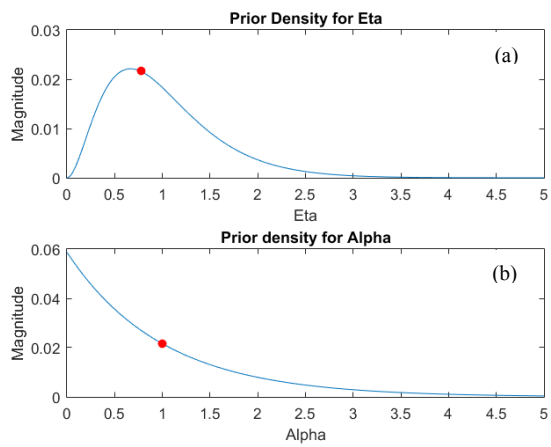


Figure 4: 2D Prior densities for α, η.

In Fig. 4a, the point estimate lies almost on the peak of the density distribution which signifies our confidence in the value of scale parameter of the prior obtained from the MC model. But if we look at Fig. 4b, the point estimate

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

does not lie on the peak, rather much lower than the highest magnitude point. This is chosen because the shape parameter is a system characteristic, which is less likely to change. For the data distribution we saw that α is much less than one in Eq. 2. So we express more confidence in the lower values of α than the one obtained from the MC model. Thus the prior density is higher for smaller values of α in Fig. 4b.

Posterior Inference

The posterior distribution is a fusion of the prior and data distributions that contains all the information of the parameters of the system, in our case α and η. From Eq. 1 we get the proportionality equation (or kernel) for posterior as:

$$p(\alpha,\eta|x) \propto \alpha^k \eta^{k+\beta-1}\left(\prod_i^k x_i\right)^{\alpha-1} e^{(-\eta\sum x_i^\alpha-\alpha-\eta\beta)}$$

Here k is the total number of data points. As this is a complicated joint distribution of α and η, it is not possible to obtain independent and identically distributed samples directly from this unnormalized kernel. We use the Metropolis Hastings (MH) algorithm which is a type of iterative Markov Chain Monte Carlo (MCMC) technique [3]. The parameters α and η are calculated as sample averages of realizations of Markov chains, so one has to ensure that the Markov chain has converged before drawing the samples. To generate the random samples of α and η, a "proposal density" is used, given the samples from previous iteration. Following proposal density is suggested in [9].

$$q(\alpha',\eta'|\alpha,\eta) = \frac{1}{\alpha\eta} e^{\left(\frac{\alpha'}{\alpha}-\frac{\eta'}{\eta}\right)}$$

Here α',η' are the new samples and α,η are the previous samples. In MH algorithm first we draw samples from the proposal density. This sample is then accepted or rejected as per the acceptance probability given by:

$$a\big((\alpha',\eta'),(\alpha,\eta)\big) = min\left\{1, \frac{p(\alpha',\eta')/q(\alpha',\eta'|\alpha,\eta)}{p(\alpha,\eta)/q(\alpha,\eta|\alpha',\eta')}\right\}$$

There is a typical advantage of MH algorithm that we need not consider the full conditionals because the normalizing factors cancel in the ratio of acceptance probability equation. For more details on the MH algorithm please refer to [3] and [9].

RESULTS

After running 25K iterations of the MH algorithm on the posterior density, we reject the initial 5000 samples to allow the Markov chain convergence. Next we plot the remaining samples to get a 3D plot of the posterior. Fig. 5 shows the samples (green dots) obtained from the MH algorithm, and a connecting surface is plotted.

To obtain the values of α and η from the posterior, we look at the samples obtained from the MH algorithm. Fig. 6 and Fig. 7 show the samples of α and η and their histograms.

As seen, the subsequent sample generation looks stationary [5], thus we can say that the Markov chains have converged. Also looking at the histograms in Fig. 7, they resemble the normal distribution, thus the means of the samples represent the meaningful inference for α and η parameters.

We obtain the following values of α and η for the posterior:

$$\alpha = 0.6327, \eta = 1.2225 \quad (5)$$

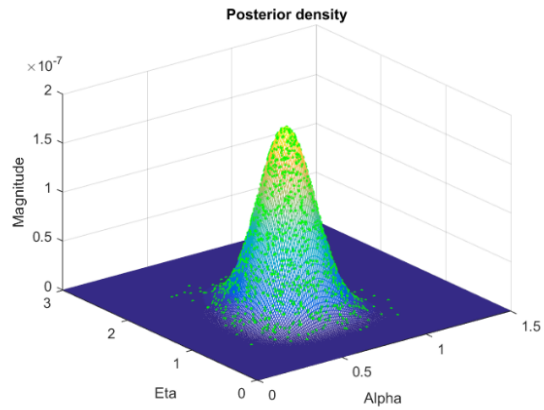


Figure 5: Posterior density for α, η.

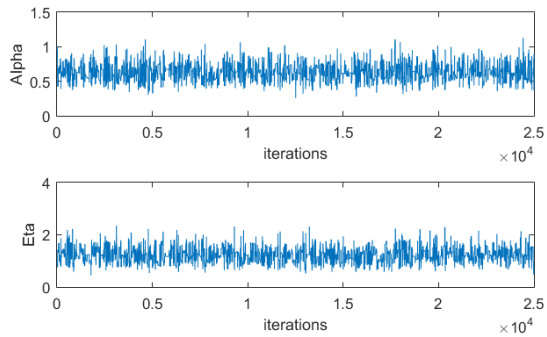


Figure 6: α,η samples from MH algorithm.

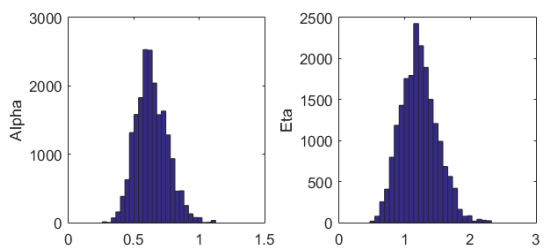


Figure 7: α,η samples' histogram from MH algorithm.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DISCUSSION

Fig. 8 shows the cumulative failure distribution function of the prior, data and posterior distributions for Weibull using the parameters in Eq. 2, Eq. 4 and Eq. 5. According to our discussion on choosing the hyperparameter β, we expressed low confidence in the value of alpha being 1. This can be seen in Fig. 8 where the posterior shape is more like the data distribution, i.e. the relative strength of the data distribution is much more than the prior distribution.

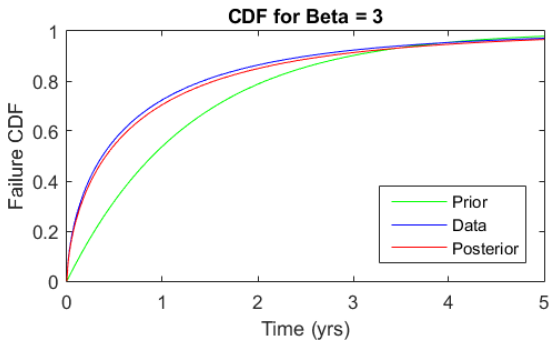


Figure 8: Cumulative failure distributions for β=3.

To illustrate the concept of relative strength of prior and data, we increase the confidence in prior by increasing the hyperparameter value. Fig. 9 shows the prior, data and posterior distribution for β equal to 15.

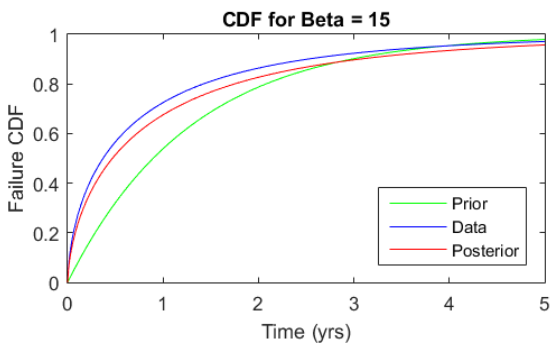


Figure 9: Cumulative failure distributions for β=15.

The prior strength is now increased that is apparent on the posterior, which is now closer to the prior as compared to Fig. 8. The posterior parameters for $\beta = 15$ are a higher value of $\alpha = 0.6404$ and lower value of $\eta = 1.1249$. For our analysis, we uphold the value of $\beta = 3$, because it represents our high confidence in the actual machine failure data, with a mild influence of the MC model results.

CONCLUSION

RHIC beam permit system has been extensively studied for its reliability characteristics. The MC model provides many insights to the reliability performance of the BPS. This includes the marginal probability values of various system level catastrophic events, marginal probabilities of failure modes of individual modules, importance of each component with respect to its failure rate and structural placement, paths of failure propagation and bottlenecks in the system. This helped understand very fine failure dynamics of the beam permit system. However it uses the military handbook which is quite conservative in its approach.

On the other hand, the historical failure data of BPS provides us with the actual failure aspects of the system. This helps to quantify the overall system failure distribution that emerged as a Weibull failure distribution with decreasing failure function. It represents the real survival behavior of the BPS. However due to high the reliability of BPS, we have a small data of only 16 failures points till date, which does not allow us to take a profound look into the system.

Thus it is necessary to emphasize the importance of both the information sources. Bayesian paradigm facilitates an excellent way to coalesce these two to furnish the most informed inference [10] about the BPS reliability, with flexibility to regulate the influence of either of the information sources according our confidence in them.

ACKNOWLEDGEMENT

This work would not have been possible without the guidance of S. T. Rachev, who helped built strong fundamentals in Bayesian statistical framework. We thank him for his continued support. We also thank T. G. Robertazzi for building basics in probability and statistics. Additionally we thank R. Schoenfeld and C. Theisen for help in mining the historical failure data.

REFERENCES

[1] P. Chitnis et al., MOPPC075, ICALEPCS 2013, San Francisco.
[2] P. Chitnis et al., MOPPC076, ICALEPCS 2013, San Francisco.
[3] S. T. Rachev et al., *Bayesian Methods in Finance*, 2008, Wiley.
[4] MIL-HDBK-217F, *Military Handbook-Reliability Prediction of Electronic Equipment*, Department of Defense, 1995.
[5] S. M. Ross, *Introduction to Probability Models*, 10th ed., 2010, Elsevier.
[6] L. M. Leemis, *Reliability, Probabilistic Models and Statistical Methods*, 2nd edition, 2009.
[7] MATLAB® Release 2015a, The MathWorks Inc., Natick, MA, 2015.
[8] K.P. Burnham et al., *Model Selection and Multimodel Inference: A Practical Information Theoretic Approach*, 2nd ed., 2002, Springer.
[9] B. Vidakovic, "MCMC methodology", http:// www2.isye.gatech.edu/~brani/isyebayes/bank/handout10.pdf
[10] K.A. Brown et al., WEPGF134, these proceedings, ICALEPCS'15, Melbourne, Australia(2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONTINUOUS DELIVERY AT SOLEIL

G. Abeillé, A. Buteau, X. Elattaoui, S. Lê, Synchrotron SOLEIL, Gif-sur-Yvette, France
G. Boissinot, Zenika, Paris, France

Abstract

IT Department of Synchrotron SOLEIL [1] is structured along of a team of software developers responsible for the development and maintenance of all software from hardware controls up to supervision applications. With a very heterogonous development environment such as, several software languages, strongly coupled components and an increasing number of releases of the entire software stacks, it has become mandatory to standardize the entire development process through a "Continuous Delivery approach"; making it easy to release and deploy on time at any time. We achieved our objectives by building up a Continuous Delivery solution around two aspects, Deployment Pipeline [2] and DevOps [3].

A deployment pipeline is achievable by extensively automating all stages of the delivery process (the continuous integration of software, the binaries build and the integration tests). Another key point of Continuous Delivery is also a close collaboration between software developers and system administrators, often known as the DevOps movement.

This paper details the feedbacks on how we have adopted this Continuous Delivery approach, modifying our daily development team life and give an overview of the future steps.

SOLEIL CONCERNS&ANALYSIS

Synchrotron SOLEIL is a service center that delivers photons to external scientific users. The 24/7 operation of the Synchrotron SOLEIL facility relies on developed software by the "Control and Acquisition Software" team. This team has to regularly deliver changes requested by the business; it has become a challenge to perform it in the right timeframe. That is why the Continuous Integration and Delivery concepts have been studied. The main objective of these concepts is to automate every single process from the code building up to its operation in production.

Continuous Integration

Continuous Integration is a practice for detecting software defects at the earliest stage possible within the full development cycle to limit them via the automation of every step. Continuous Integration relies on key elements such as:

- *Version Control*: everything must be checked in to a single version control repository: code, tests, build and deployment scripts…
- *Automated Build*: Every project must be built and tested from the command line and must be runnable into the Continuous Integration environment.
- *Team agreement*: The commitment of the whole development team is required. Everyone must frequently check in small changes; and agree that the highest priority task is to fix any change that breaks the application.

Continuous Delivery

Continuous Delivery goes a step further. It assures to deliver a fully working and tested software in small incremental chunks to the production platform. It enables frequent release and deployment. All phases of a software product lifecycle from its coding up to production maintenance have to be automated (Fig. 1).

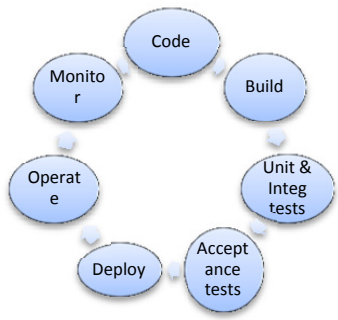


Figure 1: Continuous delivery.

DevOps

Soleil DevOps philosophy strengthens up a collaborative mode between the Development and Operations staff to achieve the same objectives: provide a full set of IT services meeting both business and IT requirements in terms of functionalities, responsiveness, and availability. DevOps is the mandatory journey to obtain a fully operational continuous delivery process.

Expected Benefits

The main expected benefits for SOLEIL of these methods are:

- Reduce all potential error-prone manual actions
- Reduce time to delivery; improve the reproducibility, repeatability and reliability of the delivery process.
- Federate the team around a common work process
- Focus on coding business requirements rather than on tooling
- Improve the synchronisation between change management and operation; reduce the number of incidents

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SOLEIL CONTEXT

Figures

The Soleil Control and Acquisition Software team is composed by:

- 4 full time equivalent Java developers
- 3 full time equivalent C++ developers
- 1 full time equivalent for system administration
- 1 full time equivalent for quality assurance

This team has to manage:

- 380 software modules in Java or C++.
- Production under Linux or Windows
- All source code and scripts are checked-in various version control repositories: 1 local *CVS* server and 3 remote *SVN* servers.
- The dependency management and build tool used for Java and C++ codes is *Maven 2* [4]. All code is continuously built and monitored within a "scheduler" web server called *Jenkins* [5]. It contains approximately 500 jobs.
- All the artifacts produced by Jenkins are stored in an artifact repository called *Nexus* [6]

The current CI platform architecture is the following (Fig.2):

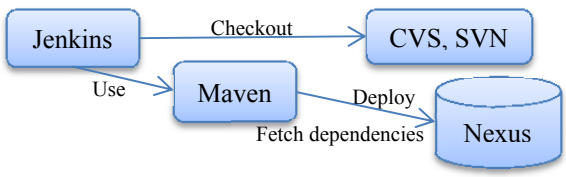


Figure 2: Current CI platform.

Each member of the team has to manage a very high number of components and their dependencies. Figure 3 is dependency tree example that illustrates the complexity of mastering a module's dependencies:

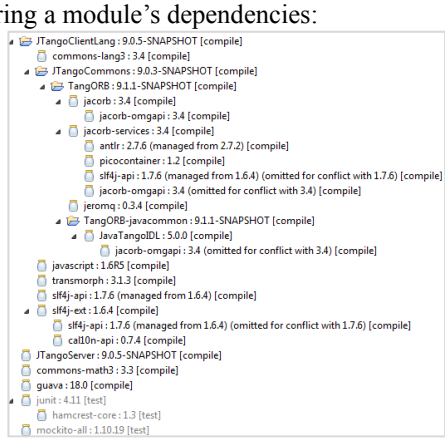


Figure 3: Dependency tree of a SOLEIL Java module.

Only a subset of our projects contains unit and integration tests.

"Zip" packages are assembled with Maven and delivered at every Machine shutdown (~ every 2 months) on 30 production platforms. The deployment process is fully automated with Linux shell scripts. The packages deployment interrupts each production platform for approximately 2 hours. Very basic acceptance tests are run manually directly after deployment during this time-slot.

The production platforms have been standardized almost everywhere:

- All the Linux servers have been just virtualised with the container technology OpenVZ [7]. There are 145 containers in operation.
- 200 industrial PC that controls hardware with windows.
- 130 "exotic" PCs that are imposed by suppliers of detectors, power supplies…

Real Life Situation

With such heterogeneity, it is a daily challenge to maintain such a Continuous Integration platform. As the main issue of the current solution is due on its global complexity, a limited number of admin are able to administer this platform.

The number of failed jobs in Jenkins is still high as some developers are considering Jenkins notification emails just as spam emails.

The current solution suffers also from technical issues, i.e. we are stuck with legacy versions of the tools because the upgrade is far from trivial.

Moreover, some parts of the C++ modules are not even considered in the continuous integration system as some Maven C++ plugins are not fully compliant (i.e. mixing compiler versions or 32/64 platforms).

Our Continuous Delivery process highlights the complexity of our software and infrastructure which has required an audit of the current process. Some aspects have been totally revamped as detailed hereafter.

OUR CONTINUOUS DELIVERY SOLUTION

Some key aspects of the delivery pipeline must be reviewed considering both methodological and technical aspects.

Team Collaboration

A fully operational deployment pipeline relies primarily on the commitment of the entire development team. This aspect has been quite underestimated and it is essential to emphasize it. A background activity is now being driven to convince the team that improving the quality of their deliverables should be the highest priority. A first step is to encourage everyone to discuss and share on technical aspects or on work methodologies either during organized meetings or stand-up-meetings. Furthermore, SOLEIL has the chance that the Control and Acquisition Software team is a mix of both developers and system administrators, but there is still room for improvement regarding the DevOps movement:

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- System administrators need to get closer to the business requirements for efficiently managing the infrastructure.
- Developers need to get closer to the operation so that it becomes a reflex to include key operational indicators and logging information for incident analysis when developing a software service.

The continuous delivery process also introduces a new central role for the quality assurance staff that is that a daily challenge considering the small size of the team and the business pressure for delivering. The quality assurance manager have to take in charge the building of the team cohesion, collaboration, communication to change for adopting the Continuous Delivery paradigm. He must have also a full DevOps profile to manage the whole deployment pipeline from assistance and training of developers to the administration and monitoring of the Continuous Delivery platform. And finally, he should constantly review and ensure that the delivery demands are respected.

Another key success factor is the involvement of the business since it is directly impacted by some aspects of the Continuous Delivery process (i.e. the deployment and the operation of our software deliverables). Exposing our process and our commitment through a service agreement enables the business to be part of the process of change management to validate, accept and plan every delivery.

Change Management

Today, many incidents are still raised due to direct modifications in the production environment. Forbid this bad habits and force a strict change management process are keys to limit incidents in production. Any change has to be tested before it goes live. Only an approved deliverable by both IT and business can be shipped to the production environment.

Simplification

For maintaining the whole Continuous Delivery platform at a reasonable total cost of ownership, the only solution is to reduce the number of parameters like:

- the number of software components;
- the number of target platforms;
- the number of Version Control System.

But the path towards simplification is tricky as it implies a deep reconsideration of our practices. Moreover, the cost of simplification can be huge.

Build lifecycle and Dependency Management

A software module metadata is split in various locations:

- Inside the build descriptor on the Version Control System
- Inside the project job configuration in Jenkins

To manage the large set of components, it is mandatory to provide a central service that manages the information and lifecycle of a project, for example:

- Version Control System information
- Authors: e-mail addresses
- Status: OPEN, CLOSED

To afford developers a simple way for managing their project dependencies, the only solution for SOLEIL is to enforce a “LATEST” strategy meaning that all retrieved dependencies of a component are automatically set to the latest version. It has been made possible by using some advance features of Maven 2. But the current Maven solution suffers from issues:

- The update to the latest versions is done by a polling mechanism and it contains many caches along the build process. The latest version is not always guaranteed.
- This latest strategy relies on a deprecated Maven key word “RELEASE”.

Consequently, we cannot upgrade to the latest build tools (Maven 3 [4] or Gradle 2 [8]) because of this “LATEST strategy”. Since there is no off-the-shelf solution, the only way is to design a totally custom one.
The lifecycle of our components is imposed by Maven with only 2 statuses (SNAPSHOT and RELEASE). The review of the solution has led us defining new statuses and promotion in between. The new defined statuses are:

- *BUILD*: project under build
- *INTEGRATION*: project ready for continuous integration
- *RELEASE*: project build, integrated and tagged. Ready for packaging.

On the other hand, The Maven solution for C++ does not fulfil all the requirements. So the best solution is to move to a native tool for C++ build like for example Scons [9] or CMake [10].

User Acceptance Test (UAT) Platform

Acceptance testing ensures that the user needs, requirements and business process are met.

At SOLEIL, the only actual testing is done manually directly into production just after deployment by the IT and the business teams (when related to Machine equipment). The deployment operation is quite long and risky, with an iterative patching and re patching process until disappearance of incidents, regardless of user satisfaction.

It is obvious that an UAT platform that mirrors the production should be set. Then the continuous integration server will automatically deploy our fresh packages with the same deployment process as the production’s one. On this platform, we should at least run manual tests before establishing more and more automated tests.

Considering that 20% of our software controls hardware, with some of them unique (vacuum pumps, power supplies…), we have to develop a simulation strategy.

Deployment

Currently, only a system administrator deploys zip packages for linux and windows. A better solution is to

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

use the OS native packaging system such as rpm for our RedHat distributions so that libraries conflicts can be detected at the deployment time instead of runtime.

Moreover, all of our binaries are deployed on a central mounting point with Linux shell scripts. A better solution is to deploy only the necessary binaries directly of the right target with tools that allow easy large-scale deployment whilst adding consistence and compliance checks. So we need to include tools like Puppet [11], Ansible [12] or CFEngine [13]…for our deployment phase.

Monitoring

Enabling efficient and easy ways to monitor the operation of our deliveries is also one of the key success factors.

For the monitoring of the infrastructure we use Nagios [14]. Our system administrator is currently thinly tuning it to provide key performance indicators, so that every alarm will be considered as an incident that must be taken into account immediately.

Our software applications do not provide at this stage enough audit logging or monitoring indicators. We must standardize our application logging system for archiving all of them. Developers should also integrate runtime application indicators that can then be included in a monitoring system and thus raise alarms.

Our New Architecture

To address the whole technical points previously addressed, a new solution has been designed. Here is the overview of the architecture (Fig. 4):

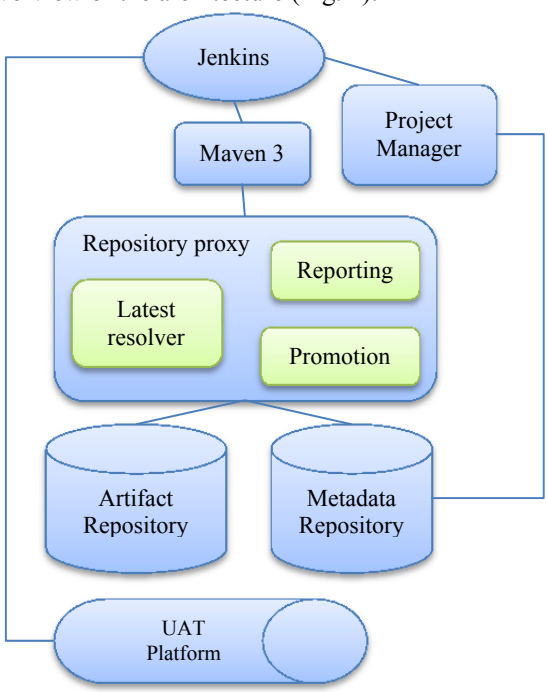


Figure 4: The new Continuous Delivery platform.

The "*Project Manager*" provides the service to create, update and close a project. This service will automatically configure the Continuous Integration server (Jenkins).

The "*Repository proxy*" is a service on top a collection of binary software artifacts and metadata. It provides some custom functions such as:

- *Latest Resolver*: it retrieves the latest version of a component.
- *Promotion*: it promotes a component ensuring the pre-defined lifecycle. For example, releasing a component will create a tag on the Version Control System and deploy the binary into the artifact repository.
- *Reporting*: provide audit reports like for example the list of promoted artifacts between two dates.

The whole solution is based on the Java technology. All services are provided as REST (REpresentational State Transfer) Web Services following the Microservices architecture [15]. The artifact repository in implemented with a MongoDB [16] document oriented NoSQL database. For the UAT automated tests, the technology choice is to be made.

Plan of Action

Putting in application the whole continuously delivery process is an enormous challenge given SOLEIL complexity and the limited resources allocated for this project. Hence, an iterative strategy has to be defined.

Firstly, the cases of Java and C++ have to be addressed separately. So a new continuous integration server will be set-up first for Java modules. The C++ case will be addressed later. Secondly, the continuous integration platform will be upgraded before moving on other aspects such as the UAT platform, the deployment and monitoring processes.

The new architecture is currently under development. The "project manager" service is fully operational and the "repository proxy" service is still under coding.

The SOLEIL infrastructure virtualization has opened up new possibilities. It is now faster and easier to set up and change the environment for development, continuous delivery, and acceptance platforms. So, the set-up and qualification of new solution platform is in progress.

The toggle to the new Continuous Integration solution for the Java part is straightforward for Java developers as the build tool is still the same. The only induced change should be in the management of the component lifecycle; the developers have to be trained on how to promote their components.

For the C++ part, the solution will depend on the workload. So the a-minima solution will keep Maven while the "deluxe" will use a native C++ build tool.

Moreover, at every new step or change, a continuous communication and training of the team must be driven to remind the IT and business teams why and how it will be applied. It is now obvious at SOLEIL that the methodologies and tool set up must follow the maturity of our team and not the other way round.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONCLUSION

Continuous Delivery is a key activity for conducting the quality of. IT services. But, the cost for putting the process smoothly in operation is always underestimated.

The Continuous Delivery approach is holistic and challenges us on every aspect of our daily practices. It enforces to manage the total cost of ownership of a service from its inception up to its operation. A deployment pipeline automation goal is to reduce the time and cost of a delivery, so each special case can add more complexity and cost to it. Therefore there is no other solution than to standardizing and simplifying all our practices. In this context, SOLEIL is currently initiating the implementation of some ITIL practices which is a framework of best practices for managing IT services [17].

After a 10 years' experience of operation, SOLEIL still need to emphasize a close coordination between developers and system administrators following DevOps philosophy.

Continuous Delivery relies on strong convictions. You have to be intimately convinced and motivated to be able to convince the whole company including technical staff; managers; and even business users.

In conclusion, SOLEIL can finally see the light at the end of the deployment pipeline but putting in application the Continuous Delivery paradigm is really a long-winded task worthwhile the effort to obtain the requested quality level of service.

REFERENCES

[1] http://www.synchrotron-soleil.fr
[2] http://martinfowler.com/bliki/DeploymentPipeline.html
[3] https://sdarchitect.wordpress.com/2012/07/24/understanding-devops-part-1-defining-devops
[4] https://maven.apache.org
[5] https://jenkins-ci.org
[6] http://www.sonatype.org/nexus
[7] https://openvz.org
[8] http://gradle.org
[9] http://www.scons.org
[10] https://cmake.org
[11] https://puppetlabs.com
[12] http://www.ansible.com
[13] http://cfengine.com
[14] https://www.nagios.org
[15] https://www.mongodb.org
[16] http://martinfowler.com/articles/microservices.html
[17] Improving SOLEIL Computing Operation with a Service-Oriented Approach, A.Buteau, ICALECPS 2015, Melbourne, MOPGF150

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

SHOT RATE IMPROVEMENT STRIVE FOR THE NATIONAL IGNITION FACILITY (NIF)

G. Brunton#, G. Bowers, A. Conder, JM. Di Nicola, P. Di Nicola, M. Fedorov, B. Fishler, R. Fleming, G. Lau, D. Kalantar, D. Mathisen, V. Miller-Kamm, V. Pacheu, M. Paul, R. Reed, J. Rouse, R. Sanchez, M. Shaw, E. Stout, Y. Tang, S. Weaver, R. Wilson
Lawrence Livermore National Laboratory, Livermore, CA 94550 USA

Abstract

The National Ignition Facility (NIF) is the world's largest and most energetic laser experimental facility with 192 beams capable of delivering 1.8 megajoules of 500-terawatt ultraviolet laser energy. The energy, temperatures and pressures capable of being generated allow scientists the ability to generate conditions similar to the center of the sun and explore physics of planetary interiors, supernovae, black holes and thermonuclear burn. NIF has transitioned to a 24x7 operational facility and in the past year significant focus has been placed on increasing the volume of experimental shots capable of being conducted so as to satisfy the demand from the wide range of user groups. The goal for the current fiscal year is a shot rate of 300 (> 50% increase over the previous year), increasing to a sustainable rate of 400 the year after. The primary focus areas to achieve these increases are; making more shot time available, improvements in experiment scheduling, and reducing the duration of a shot cycle. This talk will discuss the control system improvements implemented and planned to reduce the shot cycle duration and the systematic approaches taken to identify and prioritize them.

INTRODUCTION

The National Ignition Facility (NIF) laser system [1] provides a scientific center for the study of inertial confinement fusion (ICF) and matter at extreme energy densities and pressures [2]. Each NIF experiment, or shot cycle, is managed by the Integrated Computer Control System (ICCS) [3], which uses a scalable software architecture running code on more than 2000 front end processors, embedded controllers and supervisory servers. The NIF control system operates laser and industrial controls hardware containing 66,000 control points to ensure that all of NIF's laser beams arrive at the target within 30 picoseconds of each other and are aligned to a pointing accuracy of less than 50 microns. Every NIF shot cycle [4] consists of approximately 1.6 million sequenced control point operations, such as beampath alignment, configuring diagnostics and arming triggers.

NIF has transitioned to a 24x7 operational facility and in the past year significant focus has been placed on increasing the volume of experimental shots capable of being conducted so as to satisfy the demand from its wide range of scientific user groups and maximizing the return of experimental data. The goal for the FY15 fiscal year was defined as a shot rate of 300 (> 50% increase over the previous year), increasing to a sustainable rate of 400 the year after. Figure 1 depicts the historical shot rate achieved on NIF and the goals going forward. Also indicated are the planned methods for achieving these goals:

- Formalizing a 24/5 shot week and increasing the number of weeks per year performing experiments (increased to 44 weeks)
- Improving the NIF shot scheduling paradigm by grouping similar shots to minimize the frequency of mid-week diagnostic reconfigurations
- Reducing the shot to shot durations. This includes both the shot cycle duration and the activities performed between each shot

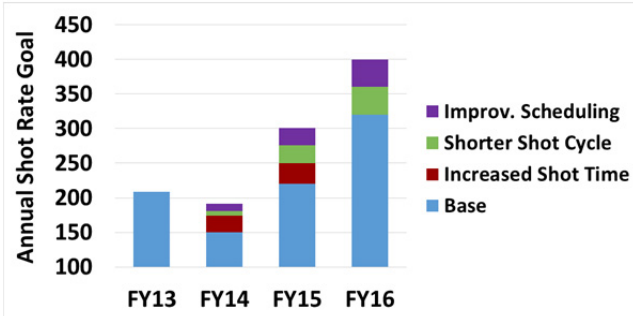


Figure 1: Historical and Planned NIF shot rates.

With both the increase in shot weeks and shot scheduling improvements commenced focus shifted to reducing the shot to shot durations. This work discusses the approach that was taken, a summary of the enhancements implemented and the results achieved.

ANALYSIS APPROACH

During each shot metrics are collected and archived that capture execution times of all phases and steps of the cycle. The mining of information from this vast data set was a critical component of the analysis and led to many of the identified improvements. To assist in rapidly visualizing these metrics the team leveraged the use of Splunk [5], a tool that we previously had extensively used

brunton2@llnl.gov

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

to monitoring the health of the control system [6]. Splunk quickly highlighted key metrics at a high level for each shot cycle by presenting the data using a Gantt-like view with detailed information associated with each phase.

NIF shot cycle durations vary considerably depending of the complexity of the experiment and thus the metrics were subcategorized as such. The most recent 3 months of metrics were analyzed and a baseline defined to measure future success. Table 1 highlights these baseline metrics

Table 1: Q4FY14 Average Shot Cycle Durations

Shot Cycle Category	Average Shot Cycle Duration (Hours)
Warm Simple	19.6
Warm Complex	22.7
Cold	24.0
Layered	45.2

A top-down analysis was performed to identify the major shot activity sequences and determine, given system requirements and constraints, whether any resequencing could be performed to reduce the overall critical path through improved parallelization. In doing so both the critical path and 'close-to' critical path sequences were analyzed to ensure the evaluation of savings was accurate.

A bottom-up analysis was then performed on significant critical path activities to identify whether any could be optimized. As with the top-down analysis, Splunk greatly assisted in providing a rapid and consistent method to mine, visualize and identify the key significant activities worthy of further detailed analysis.

Return on investments were calculated on each proposed enhancement to ensure sufficient savings were achievable and that each was assigned the appropriate prioritization. The following section summarizes the major control system enhancements selected and implemented.

ENHANCEMENTS SUMMARY

Laser Preparation Parallel Shot Cycles

Top-down analysis identified that although laser energetic qualification was performed once target and diagnostic alignment were completed it could in fact be performed in parallel. There was no system requirement for this constraint other that the presently imposed automated shot cycle sequencing. A shot cycle enhancement was planned and implemented to allow NIF operations to perform these functions concurrently in two parallel shot cycles and synchronize once complete. This allowed the laser setup activity to be performed off-critical-path which resulted in reducing the overall shot cycle by up to 1 hour. Figure 2 presents a simplified Gantt view of the major shot cycle activities and depicts how they are now executed across two parallel shot cycles

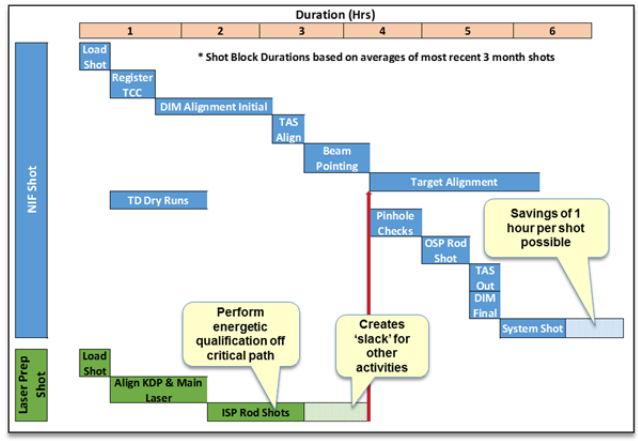


Figure 2: Parallel Shot Cycles Gantt View.

With the introduction of parallel laser preparation shot cycles two activities were identified to utilize the 'slack' time created. The first was the verification of amplitude modulation levels that is required following a facility wavelength reconfiguration (occurred 120 times during FY14). The second is the pulse shape calibration loop (Loop1) performed for high precision experiments (occurred 150 times during FY14). Both of these activities previously required discrete shot cycles to complete (2-3 hrs each).

Leveraging the new laser preparation shot framework, software modifications were implemented to optionally execute both these high frequency activities as part of the primary experiment and obviate the need for discrete shot cycles.

Target Alignment Assistance Tool

Target alignment sequences are typically one of the longest duration critical path activities and additionally had the greatest variances. Unlike NIF laser alignment, the target alignment process remains a largely manual process with some operator assists. The variations in NIF target types and alignment fiducials have historically made automating this process difficult. Much of the variance was identified to be caused by user input errors and experience levels in operational staff.

To aid with solving this, a new Target Alignment Assistance Tool (TAAT) was implemented. The tool uses Excel based script templates to semi-automate the manual alignment approach. The tool interfaces directly into the control system for measurement data provides the ability to embed alignment formulae, defined by subject matter experts, thereby removing opportunities for user input error. This data driven approach provides flexibility to adapt to novel target types and new alignment templates can rapidly be deployed without the need for any software modifications. Table 2 below details the reductions in user input achieved through use of the tool. In addition to a reduction of alignment time variance the use of the TAAT tool is saving approximately 30 minutes per alignment sequence over the previous manual approach.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Table 2: Comparison of User Input Savings Using TAAT

	# Measurements	# Moves	# Data Entries	# Move Choices
With Manual Alignment	1413	130	83	26
With TAAT	763	34	0	0
Savings (%)	46%	74%	100%	100%

New Positioner Rules Of Engagement

Another source of alignment duration variance was found to be operator availability and coordination. Until recently the majority of the positioner movement rules of engagement required 2 operators (move requestor and move approver/observer). Lack of availability of one of the two required operators, due to various disruptions and distractions, often impacted the positioner alignment durations. To reduce the impact the rules of engagement were revised. The new rules resulted in the ability for a single operator to perform 90% of the shot cycle positioner moves without a secondary approver/observer whilst still ensuring machine safety was not compromised. The control software enhancements provide verification that the new rules are enforced. Only the final fine alignment steps, when positioners are within very close proximity, now require a 2 operator rule. The revised rules allow single operator positioner movements under the following circumstances:

- Concurrent positioner movements (insertions and retractions) for positioners outside the safe handoff boundary (1.4m from chamber center)
- Single positioner moves when no other positioners are unlocked and no other positioners are inside the critical zone (50cm from chamber center)
- Concurrent positioner retractions (z-only moves inside safe handoff zone) if none are deemed to be entangleable

Critical Path Analysis Tool

There was a significant initial effort required in manual data analysis to identify the critical path for a single or group of shot. It was challenging to identify how much slack existed in a non-critical path segment so as to accurately evaluate expected benefits from reducing the critical path. As we fully expected to continue to optimize the sequencing of the shot cycle we invested in the development of a Critical Path Analysis Tool (CPAT) to ease the burden of this analysis and to aid in measuring benefits from enhancements and to identify further gains. The tool provides far more detailed metrics than the manual analysis. Figure 3 displays a screenshot of the CPAT tool showing the visualization of a critical path segment of the shot cycle. The tool has already been used to identify sub-optimal sequencing of the pre-amplifier (rod) shot activities. With resequencing, 5 minutes have been saved of each of the 1300 rod shots fired annually which equates to approximately 11 additional shot cycles.

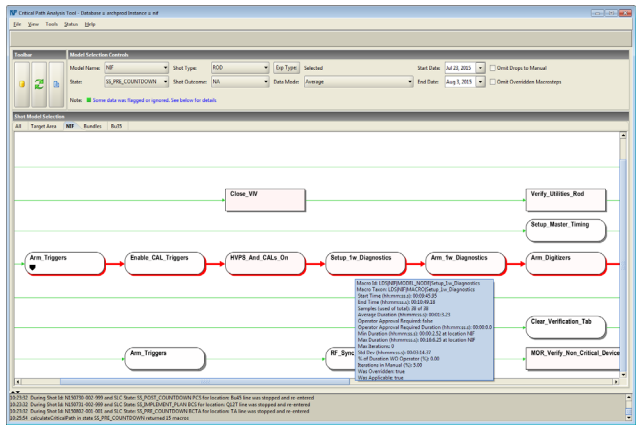


Figure 3: Critical Path Analysis Tool (CPAT) example.

Final Optics Damage Inspection Optimization

A long critical path activity that occurs between shot cycles is the inspection of the NIF final optics. The activity uses a single camera located at target chamber center and it is positioned to inspect each optic set using a hexapod manipulator. With only a single camera, that needs to inspect up to 192 beamlines, the process can take over 3 hours to complete. By analyzing source code debug logging during a scan it was identified that by disabling focus motor braking during the scan of a single beamline and performing moves in parallel with reconfiguration of the backlighting laser that significant savings were feasible. The software modifications yielded savings of 50 minutes per scan. With inspections required 2-4 times per week the savings totalled 110 operational hours per year.

Diagnostic Foreline Vacuum Automation

Diagnostic reconfigurations between each shot can be a costly activity to vent and pump down following the exchange. To reduce the impact, vacuum automation was developed for the valve and hi-vac pump management system drastically reducing the amount of human interaction from industrial controls operations staff. Automation was added for 3 diagnostic positioners, the Cryo Target Positioner and 5 of the fixed diagnostic packages. The automation allows operations staff to initiate vent and pump cycles with a few button clicks and the automation manages the valve and pump control. The automation has significantly reduced operator interactions thus avoiding availability delays and potential for human error. Savings obtained range from 15-30 minutes per diagnostic positioner/package.

SHOT RATE IMPROVEMENT STATUS

With the majority of the shot rate improvements having now been deployed benefits obtained from them were measured. NIF set itself a challenging goal of 300 target shots during FY15 which represented a shot rate greater than 50% more than FY14. Figure 4 details the monthly shot volume and the average weekly shot rates achieved this year. As shown, NIF completed its goal 1 month

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ahead of schedule with an anticipated total annual shot count of approximately 350.

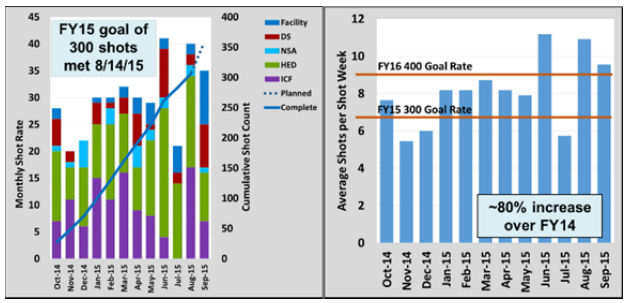


Figure 4: FY15 Shot rate metrics.

The two weekly shot rate threshold lines show the required rates for 300 and 400 annual shots respectively. During the final months of the year NIF exceeded the 400 shot rate threshold, with the exception of July due to a facility maintenance period, therefore we believe we are in an excellent position to meet the FY16 goal.

As a result of all the shot rate improvement initiatives, significant reductions have also been achieved in the shot to shot durations for all experiment categories as indicated in the table 3.

Table 3: Shot to Shot Average Duration Improvements

	Baseline Average Shot Cycle Duration (Hours)	Post-Optimizations Average Shot Cycle Duration (Hours)
Warm Simple	19.6	9.9
Warm Complex	22.7	11.7
Cold	24.0	12.3
Layered	45.2	23.0

FUTURE WORK

Although the majority of the shot rate improvement activities were completed in the past year NIF will be deploying three further major capabilities in the next two years to further benefit the shot rate.

'Gatling' shot cycle support to facilitate back to back system shots with only a single costly laser setup. Target exchanges are performed between each system shot and each uses a subset of the NIF beamlines. The first experiments are planned to be conducted this fall on NIF with a total of 8 shots expected to be performed in a 24 hour period which will represent a 3 hour saving for each additional shot performed in the series.

The Advanced Tracking Laser Alignment System (ATLAS) will use a laser tracking package for diagnostic positioner alignments and replace the existing alignment method using opposed port imaging systems. This method decouples the diagnostic alignment from the need to use the Target Alignment System (TAS) and thus removing the activity from the critical path.

Two additional Target and Diagnostic Manipulators (TANDM) are also being deployed over the next two years. These multifunction positioners will allow continuous layering on the Cryo target positioner without impacting the shot schedule. The TANDM positioners require the ATLAS system as no opposing port alignment system (OPAS) is being implemented for alignment.

CONCLUSION

This paper presents the systems based approach taken to identify and implement the major enhancements to increase the NIF shot rate in FY15 and the results achieved. Historical metrics proved invaluable in the process to accurately assess the cost of shot activities and rapidly isolate where improvements would yield the biggest gains.

Both a top-down and bottom-up approach to analysis produced results but the former (i.e. big picture) typically yielded the biggest returns. Capturing accurate return on investments was very important as it aided in selling the need for change and to assign the appropriate prioritization.

System reliability is also very important to analyze in a highly parallelized control system as a single underperforming component can be very costly in a system with overall speed governed by the slowest cog.

System optimizations are often best left until system is complete however it is very important to consider throughout the system design phases to ensure constraints are not being unnecessarily imposed that would limit optimization potentials in the future.

ACKNOWLEDGEMENT

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344

REFERENCES

[1] E. I. Moses, "Overview of the National Ignition Facility", Fusion Science and Technology 54 (2008).

[2] J. D. Lindl, et al, "The US ICF Ignition Program and the Inertial Fusion Energy Program", 30th EPS Conference on Controlled Fusion and Plasma Physics, St. Petersburg, Russia, 2003.

[3] P.J. Van Arsdall, et al "National Ignition Facility Integrated Computer Control System", Third Annual International Conference on Solid State Lasers for Application (SSLA) to Inertial Confinement Fusion (ICF), Monterey, CA, 1998.

[4] L. Lagin, et al, "Shot Automation for the National Ignition Facility", ICALEPCS, Geneva, Switzerland, 2005.

[5] http://www.splunk.com, Splunk® IT Data Engine.

[6] J. Fisher, et al, "Monitoring of the National Ignition Facility Integrated Computer Control System", ICALEPCS, San Francisco, CA, 2013.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INTRODUCING THE SCRUM FRAMEWORK AS PART OF THE PRODUCT DEVELOPMENT STRATEGY FOR THE ALBA CONTROL SYSTEM

G. Cuni, F. Becheri, D. Fernandez-Carreiras, Z. Reszela, S. Rubio-Manrique, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain

Abstract

The Controls Section at the Synchrotron ALBA[1], produces and supports the software to operate the accelerators, the beamlines and the peripheral laboratories. It covers a wide range of disciplines like vacuum, motion, data acquisition and analysis, graphical interfaces, or archiving. Since the installation and commissioning phases, we have been producing the software solutions mostly in single-developer projects based on the personal criteria. This organization scheme allowed each control engineer to gain the expertise in particular areas by being the unit contact responsible to develop and deliver products. In order to enrich the designs and improve the quality of solutions we have grouped the engineers in teams. The hierarchy of the product backlogs represents the desired features and the known defects in a transparent way. Instead of planning the whole project upfront, we try to design the products incrementally and develop them in short iterations mitigating the risk of not satisfying the evolving user requirements. This paper describes the introduction of the Scrum framework as the product development strategy in a service oriented organization like the Computing Division at Alba.

INTRODUCTION

Alba is a 3rd generation Synchrotron Light facility located in Cerdanyola del Vallès, (Barcelona), commissioned in 2012 with seven operational soft and hard X-ray beamlines, devoted to experiments in biosciences, condensed matter (magnetic and electronic properties, nanoscience) and materials science among others. Nowadays, two new beamlines are in construction (infrared microspectroscopy and low-energy ultra-high-resolution angular photoemission for complex materials).

In addition to the Controls Section, the Computing and Controls Division has other 3 sections devoted to management systems software, electronics design and support or Information Technology systems and network administration. The mission of the division is to offer services and give support to the beamlines, laboratories and accelerators, as well as to the whole installation to produce high quality experiments. The range of services comprises hardware and software solutions for control systems, personal safety, equipment protection, data acquisition, data analysis, high performance computing, document management, and networks and communications among others.

ORGANIZATION OF THE CONTROL SOFTWARE DEVELOPMENT DURING THE INSTALLATION AND EARLY OPERATION

The Controls Section was setup in the early 2005. During the phases of design, construction and installation the size of the team was progressively increasing until the current 16 members. The section was responsible for the design, installation and commissioning of projects in a wide range of disciplines, for both accelerators and beamlines. The fact of being a single support group for the whole facility showed important competitive advantages during the installation, where peak loads, could be better managed balancing manpower among the different customers. The software design, development, and the protection systems were scheduled in parallel for the accelerators and the beamlines, although the installation and commissioning of the accelerators engendered peak loads, which often took precedence in order to match the milestones and manage the critical paths for the deliveries. The project and the group relied on common tools, technologies, international collaborations and internal-developed transversal libraries and frameworks, crucial for keeping the service support manageable and the service level agreements.

The different products and subsystems were progressively delivered, and run in production. At this stage, the section started to share efforts between the on-going projects and the service support with a number of bugs and defects to fix at the beginning, and later an increasing number of service requests and requests for change. The service desk relies on a ticketing system where every issue is registered with the relevant information to be classified by priority, unit (e.g. an accelerator's group, a specific beamline, etc.), and service (e.g. a subsystem like vacuum, motion, graphical interfaces, etc.). In this scenario, although all products were based on common tools, they were mostly developed by individuals, who ended up taking the responsibility for the related tickets (bugs or new feature requests), and therefore the particular knowledge of the product could not be spread.

Once the Control System has been delivered, during the operation, new features, bug fixes, and advanced functionalities need to be properly scheduled for deployment and commissioning in very restrictive time slots (mostly on shutdowns or "maintenance days"). Furthermore, during operation, the support and maintenance tasks for the running products gain priority

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

against new developments due to its critical effect on the stability, the performance, or new capabilities of the installation.

The combination of service management best practices and project management methodologies for the development of controls software has proven to be successful [2] allowing us to give support and rapidly respond to incidents as well as periodically provide updated versions of the software packages of the Alba Control System [3].

SPREADING THE KNOWLEDGE

Taking responsibility for a project is often a great motivation for a software developer. Making one single person responsible for the development, installation and support of a particular product is in many cases efficient. The developer feels immediately committed to the project and motivated for delivering a high quality product in time and within specifications. However, sometimes a project may be unmanageable by a single person, or in case of a deployed product in operation, the necessary knowledge needs to be spread and transferred to a bigger group to accomplish the required service level (i.e. 24/7). Having regular meetings among all the members of the group help to share information, functionalities, feature requests etc., so the team gets updated on the evolution of the products. However, although weekly meetings keep the group informed and the dedicated collaborative brainstorming sessions are also setup occasionally to inject fresh ideas into the project dynamics, the ownership of the products remains in the developer hands.

During the operation and support of the service, having the specific know-how of complex products centralized in one person has important risks. An urgent issue may be raised and require dedication of a person who had already committed 100% of his work time to another issue, a scheduled maintenance task or a delivery on a fixed deadline. Being the only developer able to do some tasks creates also internal stress when equally urgent high priority issues appear. Besides, the developers can get stuck inside their own knowledge, unable to reach beyond their projects' scopes, not being able to be open to other projects or perform other collaborative activities. Moreover, if a developer leaves the team, there is an impact on the pace of product delivery. The orphaned projects need to be distributed among the rest of the team, who although know the tools, need to learn about that particular project, and the newcomer will only be able to take over some projects after long learning period. The combination of the single-developer projects with the service support of the existing products given by individuals leads to the following difficulties:

- Single developer bottlenecks
- Single person of contact for services has to take decisions on priority when new issues arise
- The team knows the highlights of projects but not the insights
- Developers are not aware and therefore do not profit from other similar solutions already provided by the group
- Newcomers (50% of the team was renewed in the last three years) need to learn common tools, frameworks, and libraries. This requires a huge effort from senior members to setup trainings and give support to the newcomers

Once the Alba's operation started, for the reasons above stated, we decided to explore other alternatives for managing projects and services in parallel. Agile methodologies focus on the collaborative work and empower team culture especially needed when interests of the group take precedence over personal projects. Sometimes, projects require an extra effort due to deadlines or events that cannot be postponed, and a team has to work together for a while, leaving the individual projects in a frozen state.

THE SCRUM FRAMEWORK

The Scrum [4] process was conceived in the early 90's, based on a research stressing the importance of teams and indicated the excellent performance achieved when teams, as small and self-organized, are fed with objectives instead of tasks: *"the best teams are those that are given direction within which they have room to advice their own tactics on how to best head toward their joint objective. Teams require autonomy to achieve excellence"*.

Scrum is a framework for developing and sustaining complex software products based on an empirical process approach where more is unknown than known and predictions have little value given a high rate of change and uncertainty. In this environment, knowledge comes from experience and making decisions based on what is known. It is not a process or a technique for building products, but a framework that makes clear the relative efficacy of the product management and development practices enabling improvement.

The Scrum Team

There are three core roles that compose the Scrum Team: a **Product Owner**, the **Development Team**, and a **Scrum Master**. The Scrum Team is self-organized in order to accomplish the objectives, and has all the skills required to execute the work to be done. The Product Owner is responsible for maximizing the value of the product and the work done by the Development Team ensuring that **Product Backlog** items have clear definitions, they are ordered to best achieve goals, and confirming that the Development Team members understand them to the level they need. These people are the professionals that create the **Product Increment** functionalities based on the Product Backlog items, and the accountability belongs to the Development Team as a whole. The Scrum Master is the facilitator whose responsibility relays on ensuring that the Scrum Team as a group follows the rules to maximize the value to be created, and promoting the self-organization.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Scrum Events

Scrum suggests a set of events with a predefined maximum time length to fulfil all the coordination and management activities. The **Sprint** is the core event, it consists of the **Sprint Planning**, **Daily Scrums**, development work, the **Sprint Review** and the **Sprint Retrospective**. Sprints duration is often limited between a week and a month. Delivering products iteratively and incrementally maximizes opportunities for feedback and reduces the risk of changing requirements or increasing complexity. The Sprint Planning is a time-boxed event where next possible deliverable increments are selected, discussing how will be the work done. Daily Scrums are events of fifteen minutes fixed length, where the Development Team shares progress since last Daily Scrum, organizes the work paying attention on any blocking situation that may prevent reach the planned increments. At the end of the Sprint, the Sprint Review checks the work that has been done, and what has been not finished based on the original planning. The Sprint Retrospective analyses how the team has performed in terms of communication, coordination, and means to achieve the work, with the objective of identifying what can be improved for the next Sprint.

Scrum Artifacts

The Product Backlog is the main artifact of Scrum, it is a unique ordered list of items that describe what is still needed to be done for a particular product and it is responsibility of the Product Owner to expose which are the current known requirements. The Sprint Backlog contains the items that have been selected, and the tasks identified in the Sprint Planning which are needed to do the work. At the end of the Sprint, the Product Increment is a concept that combines all Product Backlog items that have been completed.

EMBRACING THE SCRUM FRAMEWORK

Evaluation Phase

By the end of 2013, the Controls Section had three individual projects that required an internal boost. There was a big opportunity to evaluate if Scrum would help in the organization, execution, and coordination endeavors. During the first half of 2014 this exercise was started by defining the roles and following the Scrum rules and events, planning sprints with well-defined user stories and focusing on small product increments. It was soon appreciated the benefits of the enforced communication and implicit collaborative activities in that controlled environment, and at the end of this first implementation, there were very satisfactory results. There are plenty of documented advantages of incremental agile software development, and the list below highlights the outcomes from that research experiment:

- Designing solutions by teams instead of individuals lead to more robust products and of better quality since each member contributes from his experience and point of view
- Incremental and iterative approach helped in validating important assumptions fast what mitigated the cost of change
- Constant focus on delivering potentially shippable product increments enabled achieving earlier return of investment
- Self-organized development team did not require constant supervision and worked in a stress-less environment

These preliminary tests had proven that applying Scrum for complex software developments helps in achieving better results, and based on the size of the projects these are some recommendations:

- The best team size is from 4 to 6 members
- Two weeks sprint length has proven to be a good choice
- One controls engineer should not be part of more than one development team at a time
- Due to the support nature of the section, dedication of 60% of the time for the Sprint Backlog, keeping the other 40% for incidents, and high-priority individual service requests (which cannot be foreseen when planning) seems reasonable

Creation of the Scrum Teams

After this first contact with Scrum, a plan to enhance the Controls Section's development strategy was started. It was focused in incrementally changing the group organization for new developments and critical problems where collaborative work was clear to be an advantage.

The Scrum Master and Product Owner roles have been assigned at the beginning, sharing a common vision on how each project would enter in each team backlog. Our customers were informed about the change in how work will be planned.

During mid-2014 the first group built was the Beamline Control Systems Scrum Team. The communication channels between beamline scientists and controls-contacts didn't require any change, and it was enough to take care identifying those requests about complex problems for which design and implementation activities would be worth sharing. Those requests were then merged in a general, serialized and prioritized top-level product backlog that provided a clear view of what needed to be addressed during each sprint iteration for all beamline developments.

After few sprints, a second group mostly dedicated to internal software frameworks did the same transition, starting by defining specific product backlogs for all the individual projects that had been managed by each team member, and created a unified product backlog view that combined all the requirements that the team would have to work together when implementing them. While learning from co-workers, the developers enjoyed to contribute with their own points of view. At the end of 2014, we created a third team, and in the course of the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

first quarter of 2015 we have started building the fourth and last squad.

Tools for the Scrum Artifacts

In the adoption of Scrum, it was enough to enrich the same platform that was already used for project tracking: Redmine which has plugins for managing product backlogs. Having a tool that provides at any point in time the possibility to monitor the Sprint progress is essential. The team can report progress by simply interacting with a shared board, inputting the remaining work which are very valuable inputs for the Daily Scrum, and can be simply extracted from auto-generated charts at any time.

Figure 1: Team Backlog, TaskBoard and BurnDown chart.

Sprint Reviews and Retrospectives

Sprint reviews and retrospectives offer great opportunities that let information flow, and over the time, some topics evolve while others remain. Additionally, those meetings gave a chance to really keep track of the project evolution as a whole by reviewing the work from the given value perspective, and considering how it was reached. In nearly all retrospectives, one concept that repeats quite often is to bear in mind that during the sprint, one should spend some time in backlog refinement even that this task does not add any value to the current sprint product increment, it is fundamental for the next planning. Clear "definition of done" explanation in backlog items help when preparing the tasks, when developing to fulfil those requirements, because it reduces uncertainty forcing that everyone understands it. Our experience showed that it is really important to keep in mind some buffer of time reserved for support-oriented duties when calculating the sprint capabilities..

CONCLUSION

The introduction of the Scrum Framework as part of the product development strategy has proven to be very successful. It has been extremely important to start with a pilot program which facilitated the selection of the right tools, and gave hints on how the system could be tailored for our needs. Focusing on communication and the value given to our customers, as well as ensuring that there is a clear definition of what has to be done, helps during every stage of the development because all the team members are enforced to understand and agree on next steps. It is essential to have available a view of how the team is performing and what is still pending to be done within the Sprint. Also, having the Sprint Backlog defined upfront, indicates that the Product Owner admits that items not planned will be left for later phases, hence the team can concentrate the efforts in a limited set of projects. Actions like code reviews, pair programming and retrospectives about the process itself are activities that empower the team building process.

It is worth to mention that we still keep individual projects outside the Scrum Teams, mainly research activities, developments of proof of concepts and prototypes that are conceived as personal objectives and create personal and professional growth.

ACKNOWLEDGMENT

The authors would like to thank all Alba's Controls Section members for their openness in this adoption, and the collaborative mind-set when brainstorming in order to find the best matching for the Scrum theoretical framework and our specific facility environment. We would also like to thank the whole MIS section for their partnership in updating Redmine and installing the plugins needed for this purpose, as well as our customers for their cooperation and comprehension during this transition.

REFERENCES

[1] Alba website: http://www.albasynchrotron.es
[2] D. Fernández-Carreiras et al. "Using Prince2 and Itil practices for Computing Project and Service Management in a Scientific Installation", ICALEPCS2013, San Francisco, United States, TUMIB01
[3] D. Fernández-Carreiras et al. "Status of the ALBA Control System", ICALEPCS2009, Kobe, Japan, TUP090
[4] Scrum Guides website: http://www.scrumguides.org/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

USE OF AUTOMATION IN COMMISSIONING PROCESS OF THE UNDULATORS OF THE EUROPEAN X-RAY FREE ELECTRON LASER

Suren Karabekyan, Joachim Pflueger, European XFEL, Hamburg, Germany
Hongxiang Lin, Yongtao Liu, USTC/NSRL, Hefei, Anhui, People's Republic of China
Xuetao Wang, Hisense Co. Ltd., Qingdao, People's Republic of China

Abstract

For operation of the three undulator systems of the European XFEL, a total of 91 undulators are needed and have been produced. For production, magnetic measurements, tuning and commissioning of these devices only two years were foreseen by the project schedule. For these purposes, automated and optimized procedures were needed to accomplish a number of workflows, time-consuming adjustments and commissioning tasks. We created several automation programs which allowed us to reduce the time spent on the commissioning of the control system by an order of magnitude.

OBJECTIVES

All undulators are planned to be commissioned and put into operation in two years, starting from 2013 until the beginning of 2015. Given the fact that European XFEL has three magnetic measurement labs available, 3 weeks were foreseen for commissioning of the control system and the magnetic measurements and tuning of each undulator.

For each undulator, the commissioning procedure includes the following main steps:

1. Installation and adjustment of the linear encoders to achieve an accuracy of ±1 µm.
2. Adjustment of the tilt angles of the undulator girders to better than ±150 µrad.
3. Alignment of the undulator relative to the magnetic measurement bench.
4. Magnetic measurements and field fine tuning.
5. Final measurements and documentation of magnetic properties of undulator.
6. Evaluation and implementation of the correction coefficients for rotary encoders of undulator-motors.
7. Calibration of the temperature sensors.

Items 3 to 5 are part of semi-automatized magnetic measurements and tuning procedures applied to each undulator. To proceed with these tasks, approximately 2.5 weeks are necessary. The items 1, 2, 6, 7, which took without automation 2 - 2.5 weeks, must be automated to be accomplished in less than 3 days.

BASIC APPROACH

The undulator control system is based on industrial components produced by Beckhoff Automation GmbH. Using the TwinCAT software, a programmable logic controller (PLC) can be run either on an industrial PC which is used to control an undulator, or it can be run on a separate Windows PC carrying out the following steps:

- Initialization: Connection to the undulator PLC or to external controllers or devices, loading and initializing of configuration data.
- Data acquisition, depending on the configuration data.
- Data processing and analysis, graphical representation of the analysis results and calculation of required values.
- Formatting the output as required, such as creating ASCII tables, PDF protocols or configuration files for the undulator PLC

ADJUSTMENT OF THE LINEAR ENCODERS

The undulator gap is controlled by four servo motors. Each of them has its own, absolute rotary encoder. This allows an indirect measurement of the gap. A second, more precise system, is directly measuring the gap by means of two absolute linear encoders HEIDENHAIN LC 183 installed on both ends of the undulator girders (see Fig. 1). Both systems can be used alternatively to provide feedback to the motion control system.

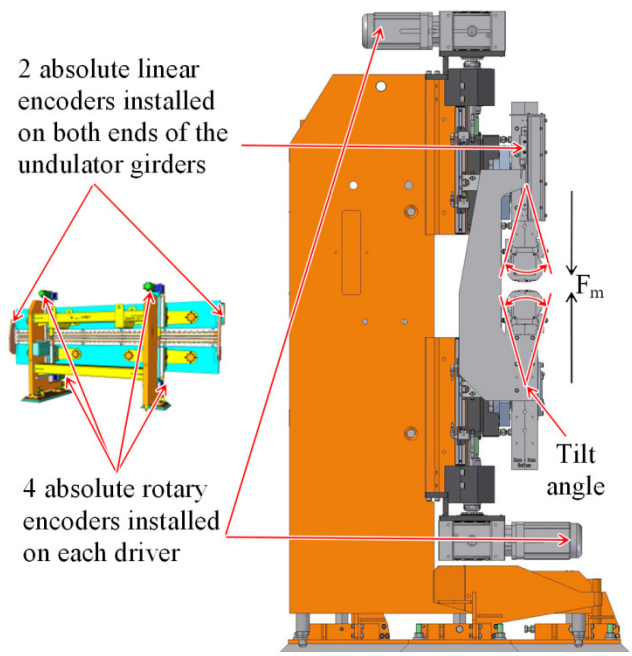


Figure 1: Undulator control and adjustment components.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The installation of a profiled linear guideway, which holds an encoder head, has to be validated in order to ensure the requirements of the gap measurement accuracy [1, 2] in the full operational range. For this purpose, two HEIDENHAIN MT 101K reference gauges, are mounted next to the linear encoders and are used to verify the adjustment quality. In order to automate this task, a C++ program was developed. It communicates with the undulator control system and adjusts the undulator gaps from a configuration file using the feedback from rotary encoders. As soon as the desired gap is reached, it reads the position of the linear encoders and the reference gauges. The gaps from the reference gauges are obtained via an HEIDENHAIN ND 287 readout device equipped with an Ethernet interface. Gap values are read from the configuration file and adjusted one by one. Hysteresis effects are identified by increasing and decreasing the gap.

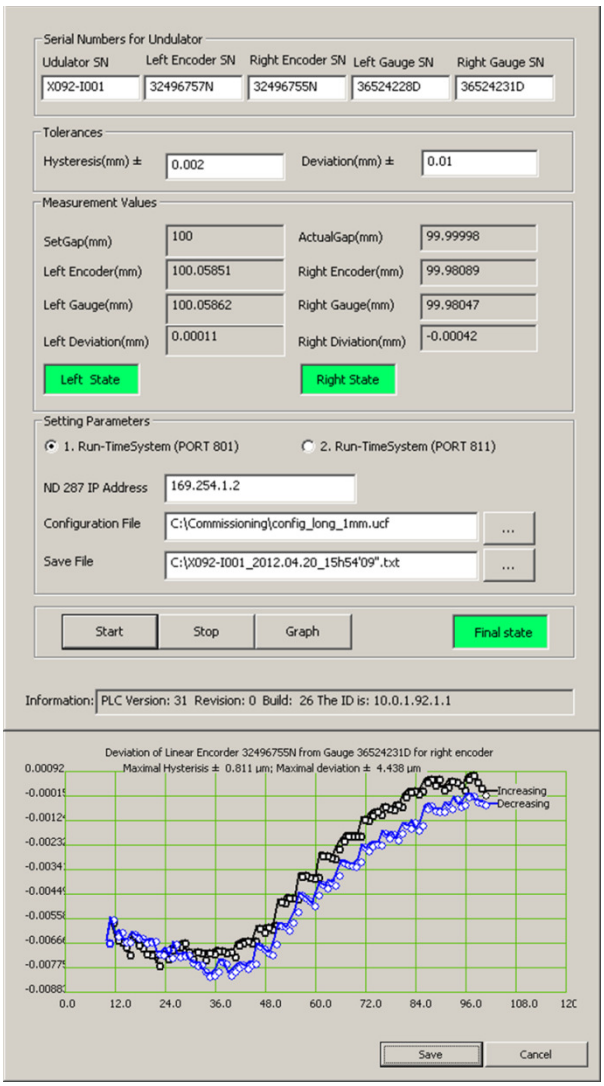


Figure 2: Main user interface and measurement results of linear encoder adjustment program.

A small hysteresis of about 1-1.5 µm in total between increase and decrease of the gap is seen in Fig. 2. After the measurements, the program calculates the maximal hysteresis as well as the deviation between encoders and gauges. The prompt results are indicated by signal lamps. The measurement results are shown in graphical form to facilitate the search for the problems. If the results are out of specs, the adjustment procedure must be repeated. Otherwise, the measurement results can be saved in the form of PDF protocol.

ADJUSTMENT OF THE TILT ANGLE OF THE UNDULATOR GIRDERS

The magnet girders of an undulator have a rotational degree of freedom. Due to magnetic forces between the upper and lower magnet structures, the frame of an undulator bends and the girder tilt angle changes as a function of the gap. To minimize the influence of the horizontal component of the magnetic field, the girder tilt should be adjusted symmetrically around the vertical position over the full operational range of the gap (see Fig. 1).

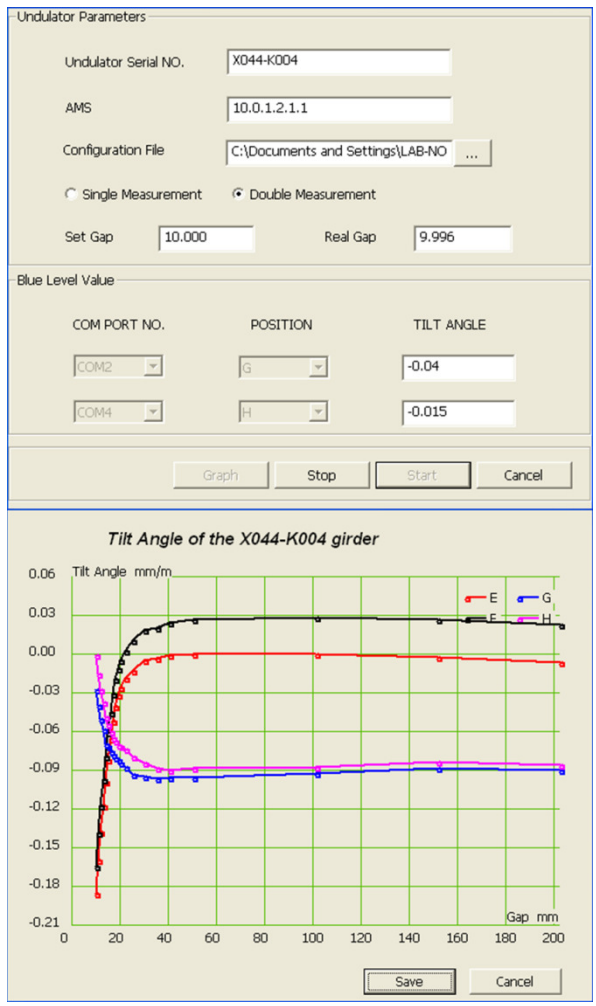


Figure 3: User interface and measurement results of girder tilt adjustment program.

A residual girder tilt of about ±200 µrad is permitted. Tilt measurements are made on one girder at a time. Two

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

WYLER BlueLEVEL devices are used on two reference surfaces. For precise adjustment, this procedure needs to be repeated several times, and this turned out to be time consuming.

As in the previous case, an automated measurement program tool based on C++ was created. The program runs on a separate PC. This computer, using one of its interfaces, is connected to BlueLEVEL instruments via a COM port to read the tilt values. Via Ethernet interface, it is connected to the Beckhoff industrial PC. Communication is established using the TwinCAT ADS library. The program sequentially changes the gap value and reads the tilt angle of the girder. The results of the measurements are presented in graphical form and can be saved as a PDF document.

EVALUATION OF THE FEEDFORWARD CORRECTION COEFFICIENTS

When gaps are small and the magnetic field is strong, the magnetic force between the girders of an XFEL undulator reaches up to 170 kN and causes a significant elastic deformation of the undulator support frame.

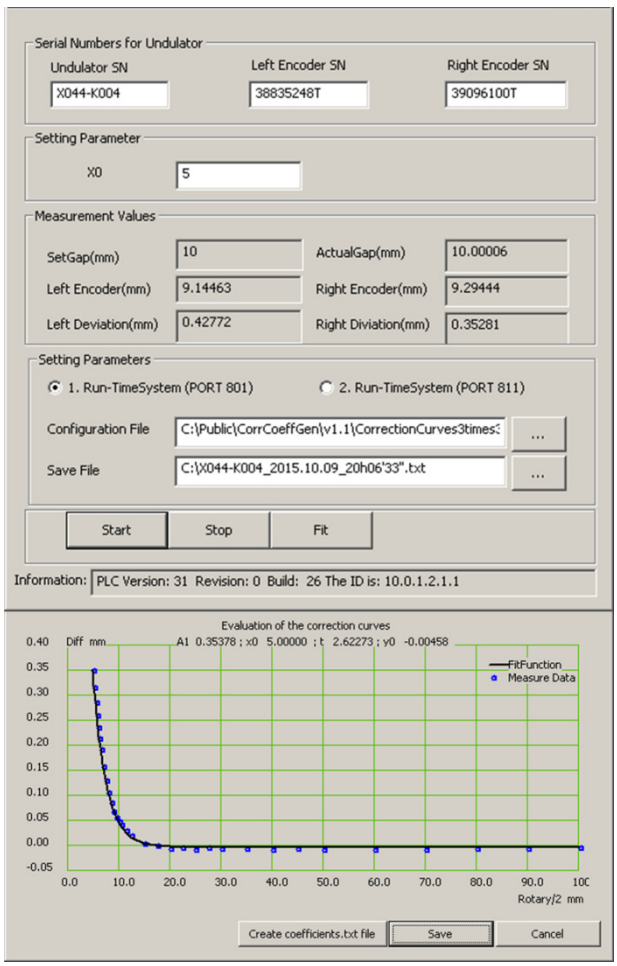


Figure 4: User interface and measurement results with fitting exponential decay function and evaluated coefficients.

Deviation between the linear and the rotary encoder readings results may reach 500 - 900 μm. The linear encoder measures the true gap, while the rotary encoder reading needs to be corrected for the deformation effects.

First, the linear encoders must be adjusted and validated against the reference gauges, readings of which are considered "true" gap values. Second, the deviation is measured in the operational gap range. Third, parameters of an appropriate fit function are calculated. Fourth, the fit function with its parameters is transferred to and used by the undulator PLC for feedforward correction. This is a standard feature in the Beckhoff Numerical Control package.

Fitting Algorithm

The Gauss–Newton least-squares algorithm was used to fit the nonlinear function $f(\boldsymbol{c}, x)$ with parameters $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ to the measured data $(x_i, y_i)(i = 1, 2, \ldots, m)$. This algorithm minimizes the sum of squares R of residual errors $r_i = f(\boldsymbol{c}, x_i) - y_i$. $R = \sum_{i=0}^{n}[f(\boldsymbol{c}, x_i) - y_i]^2$. Minimum is achieved when

$$\frac{\partial R}{\partial \delta c_j} = 2\sum_i r_i \frac{\partial r_i}{\partial c_j} = 0 \; (j = 1, \ldots, n)$$

The derivatives $\frac{\partial r_i}{\partial c_i}$ are functions of the independent variable and the parameters; therefore, these gradient equations do not have a closed form solution. An initial guess value $\boldsymbol{c}_j(0)$ must be substituted for the parameters. Afterwards, the parameters are processed iteratively, that is, the values are obtained by successive approximations: $c_j \approx c_j(k+1) = c_j(k) + \delta c_j$, where k is iteration number and $\delta \boldsymbol{c}_j$ is a vector of increments. Using first-order Taylor approximation of $f(\boldsymbol{c}, x)$ at $\boldsymbol{c}_j(0)$ results in

$$f(\vec{c}, x_i) = f_0(\boldsymbol{c}, x_i) + \sum_j \left(\frac{\partial f_0(\boldsymbol{c}, x_i)}{\partial c_j}\right) \delta c_j$$
$$= f_0(\boldsymbol{c}, x_i) + \sum_j J_{ij}\, \delta c_j$$

The residuals then are given by $r_i = y_i - f_k(c, x_i) - \sum_j J_{ij}\, \delta c_j = \Delta y_i - \sum_j J_{ij}\, \delta c_j$

To minimize R, the gradient equation is set to zero and solved for δc_j [3]. A fitting function of the form $f(\boldsymbol{c}, x) = c_1 + c_2 * e^{-\frac{(x - x_0)}{c_3}}$ was chosen and is used in the program

Program Performance

The program works in a fashion similar to the previous ones. It sequentially changes the undulator gap values, acquires the values of rotary and linear encoders, calculates the deviation values, fits the exponential decay function, and, finally, creates a file with the correction coefficients, which will be used later by the undulator PLC for feedforward correction of rotary encoders. Fig. 4 shows the control box and the graphic output created by the program.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CALIBRATION OF TEMPERATURE SENSORS FOR UNDULATOR CELL

NdFeB permanent magnet material is used for the XFEL undulators. The material has a reversible temperature coefficient η typically about -1.1x10^{-3}/°C. In order to compensate for the magnetic field change caused by temperature change, a correction of undulator gap is applied by the local control system. The gap correction Δg is calculated by the following equation: $\Delta g = \frac{\lambda_U \eta}{b+2cg/\lambda_U}\Delta T$, where η is the temperature coefficient of the material, $\Delta T = T_{Nom} - T_{Act}$, T_{Nom} is the nominal temperature at which the K-parameter of an undulator has been measured, T_{Act} is the actual temperature of the magnet structures, λ_U is the undulator period length, g is the undulator gap, b and c are empirical constants, describing the gap dependence of the magnetic peak field. The threshold value for performing a change of the undulator gap is estimated to be between 0.05 and 0.2 °C. The accuracy of the temperature measurement system is envisioned to be 0.03 °C or better [4]. To provide accurate temperature measurements, three PT100-3 sensors are mounted on the magnetic structures. A fourth one is used for the vacuum chamber temperature control system. All four sensors need to be calibrated before usage.

Calibration Method

All four temperature sensors together with a certified reference temperature sensor are thermally connected to an aluminum block and inserted into a thermally stabilized water bath in a thermostat. To proceed with calibration measurements that would allow finding of the offset and slope at 0°C for calibrating sensors, the temperature thus could be well controlled.

In order to automate the calibration procedure, a C++ program runs on a separate computer. This computer is connected to the Beckhoff industrial PC via Ethernet and reads the data from the four undulator sensors. It is also connected via COM ports to the HAAKE water bath and to a certified ALMEMO temperature measurement device with the reference temperature sensor.

The program has two operating modes - calibration and monitor. In the calibration mode, the program sequentially changes the temperature of the water bath from the minimum to the maximum values in selectable steps. Once the set temperature is achieved and stabilized, it acquires the data from calibration and reference sensors and calculates the coefficients to be applied. Each sensor is equipped with a memory block, which allows saving the calibration coefficients. After evaluation of the offset and slope, the program can write these values directly into the memory block of the calibrated sensor. The results of the measurement could also be saved as a text file.

In the monitor mode, the program compares the data of 4 calibrated sensors with the data of 4 certified reference sensors which are mounted close to undulator and vacuum chamber sensors. In case it is necessary, new offsets to the sensors are applied. The program also saves data into a text file. Fig. 5 shows the control box of the program.

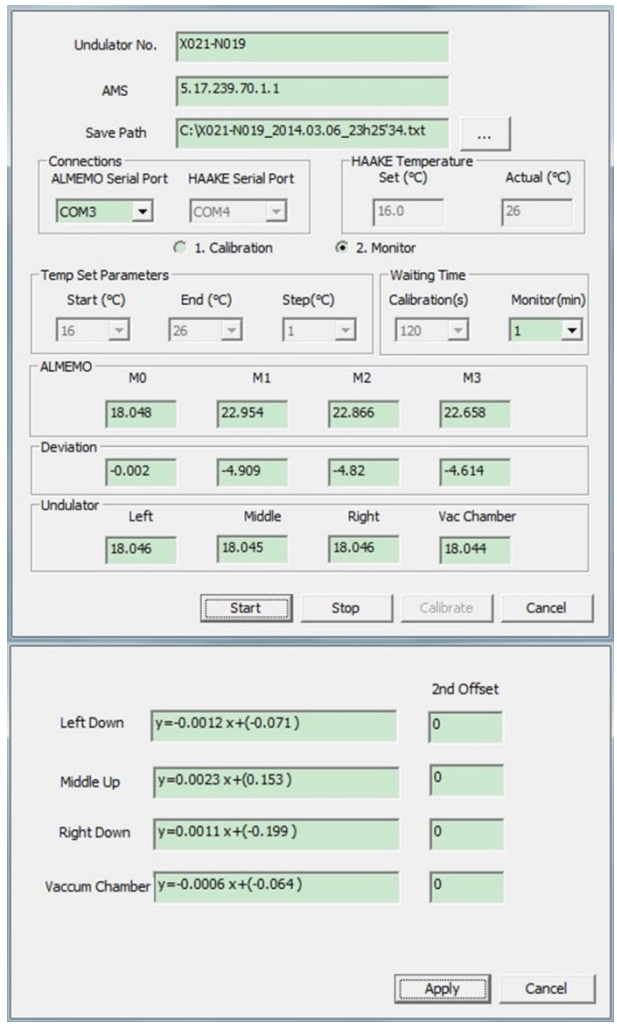


Figure 5: Main interface of the calibration program in monitor mode with evaluated coefficients.

REFERENCES

[1] S. Karabekyan et al., "The Local Control System of an Undulator Cell for the European XFEL", ICALEPCS2011, Grenoble, France, October 2011, p 450 (2011), http://www.JACoW.org

[2] S. Karabekyan et al., "The Undulator Control System for the European XFEL", IPAC'12, New Orleans, USA, May 2012, THPPR002, p 3966 (2012); http://www.JACoW.org

[3] Wikipedia website: https://en.wikipedia.org/wiki/Gauss-Newton_algorithm

[4] A. Hedqvist et al., "XFEL Activities at MSL: Undulator Temperature Compensation and Quadrupole Fiducialization", FEL2010, Malmö, Sweden, August 2010, p 675 (2010) http://www.JACoW.org

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INTERFACE MANAGEMENT FOR THE SKA TELESCOPE MANAGER

P.S. Swart*, G.M. le Roux, SKA South Africa NRF, South Africa
A. Marassi, R. Smareglia, INAF-OAT, Italy
S. Vircic, NRC-Herzberg, Canada
S.R. Chaudhuri, TRDDC, India

Abstract

The Square Kilometre Array (SKA) project is currently in the Pre-construction Phase. During this phase, the telescope subsystems are being designed. The Telescope Manager (TM) is a supervisory control and monitoring subsystem in each of the two radio telescopes of the SKA (SKA1-Low and SKA1-Mid). The TM interfaces with a number of diverse telescope subsystems. Interaction between TM and these subsystems is a major source of requirements for the TM. Careful management of TM external interfaces is therefore important. This discussion is a case study of TM interface management. Firstly, how system architectural design aspects like separation of concerns in the control hierarchy reduce telescope complexity with regards to interfaces is discussed. Secondly, the standardisation approach for monitoring and control interfaces to facilitate early elicitation of interface requirements for the TM, and to manage the diversity of interfacing subsystems is discussed. Thirdly, the relations between interface definition and requirements analysis activities, using SysML representations as an example is discussed.

BACKGROUND

The SKA Phase 1 (SKA-1) Observatory will consist of the general headquarters and two radio telescopes: SKA1-Mid in South Africa and SKA1-Low in Australia. Each telescope consists of a number of sub-systems, which are in the SKA project referred to as Elements, of which a TM is one.

There are therefore two TM items, namely SKA1-Mid TM and SKA1-Low TM. For brevity, in this paper the focus is on SKA1-Mid TM (therefore omitting specifics of SKA1-Low TM) because the two telescopes are similar from the TM perspective in its function and interfaces. For less formal reading, "TM" is used when referring to common aspects of these two items, whereas "SKA1-Mid TM" refers to the TM of the SKA1-Mid telescope.

TM OVERVIEW

The TM is a distributed, computer-based supervisory Monitor and Control (M&C) system. TM is responsible for managing [1]:

- astronomical observations,
- telescope hardware and software sub-systems in order to perform those astronomical observations,
- data to support users in achieving operational, maintenance and engineering goals.

For observation management the TM: assists creation, assessment and approval of science proposals; generates schedulable observation units from science proposals; schedules observations; and coordinates execution of the observations to achieve the data products required by the end user.

Telescope management entails configuration and high level control of the telescope Elements and collection of monitoring data which allows the user to monitor the status of Elements, infrastructural support systems and environmental conditions that impact telescope operations.

The Telescope status and configuration data are managed to form a system model that describes the status of the telescope in real-time and historically. To achieve this, the TM collects and stores telescope configuration, telescope dynamic status and environmental data. Stored data are presented to users as current and historic telescope state to assist users during system integration, commissioning, operations and maintenance.

SKA1-MID TM EXTERNAL INTERFACES

This paragraph gives an overview of the external interfaces of the TM, using SKA1-Mid TM as an example (refer [1] par. 1.3). Figure 1 shows SKA1-Mid TM and the systems that it interfaces with.

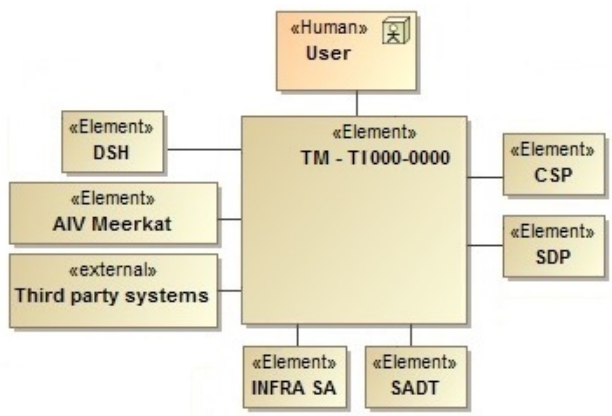


Figure 1: TM relations with SKA environment.

Interfaces with Telescope Elements

The TM has two basic types of interfaces with Elements: data exchange, and physical. The data exchange interface type is categorised into: M&C, general, networking and synchronisation and

*paul@ska.ac.za

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

timing. Categories of the physical interface type are: mechanical, power and cooling.

The TM has M&C data exchange interfaces with Elements that are to be monitored and controlled: Central Signal Processor (CSP), Science Data Processor (SDP), Dish (133 Dishes), 64 dish Karoo Array Telescope (MeerKAT) Local Monitor and Control (LMC) and Signal and Data Transport (SADT). For M&C of the MeerKAT telescope (to be integrated into SKA1-Mid telescope), LMC sub-systems are developed, which will interface with TM.

TM, via general data exchange interfaces, provides to CSP & SDP parameters required for delay tracking, beamforming, calibration, radio frequency interference mitigation, and auxiliary data that the SDP augments science data with.

Via synchronisation & timing and networking data exchange interfaces, the SADT provides services to TM.

Interfaces with Observatory Systems

TM provides the telescope's user interfaces (UIs) to scientists, observation planners, engineers, commissioners, maintainers, and operators. The TM has an application programming interface for users to programmatically create scheduling blocks. TM collects failure related data from Elements, and sends these to the Integrated Logistic Support System via a data exchange interface. TM has a data exchange interface with South African Infrastructure (INFRA SA) to obtain key indicators of INFRA status that affect telescope operations, and local weather data.

The TM uses infrastructural services (such as cooling and electrical power) provided by INFRA SA via physical (mechanical, thermal and electrical) interfaces with INFRA SA.

Interfaces with External Systems

The TM provides information from external systems (outside of the boundaries of the SKA observatory) to other SKA Elements and users, as required. These external systems are either custom experiment hardware or providers of services like: astronomical catalogues, flight information, earth orientation parameters, ionospheric prediction, satellite information, Virtual Observatory Events and weather information.

TM Interface Challenges

The TM interfaces with a diverse set of systems and implements a number of external interfaces. Interaction with such a large number of systems can be a major source of requirements for TM if interfaces are diverse and there is strong coupling between TM and interfacing systems. Careful interface management of TM external interfaces is required to address this diversity.

The SKA project is a collaboration of Element Consortia; each Element Consortium consists of a number of geographically distributed institutions, which poses the challenge of communication and coordination.

TM design work started before a comprehensive, central telescope architecture was available, causing some initial uncertainty of TM scope and boundaries.

The following paragraphs contain some strategies employed and experiences gained during TM interface management.

SKA OSERVATORY ARCHITECTURE AND TM INTERFACES

The SKA Organisation introduced separation of concerns by allocating local M&C functions to each SKA Element (referred to as Element LMC). Building on this, and to reduce complexity, we defined a control hierarchy, with less frequent, judgement-based human intervention at the top, the TM with telescope and sub-array level coordination in the middle, and frequent real-time autonomous control by Element LMC at the bottom [3].

On the monitoring side, the Element LMC performs metering and low level, real-time monitoring, whereas TM collects sensor data and performs monitoring in the telescope or sub-array context, using sensor data aggregations provided by Element LMCs.

Layering of M&C responsibilities causes separation of concerns [5] and enabled M&C interface standardisation.

M&C INTERFACE STANDARDISATION

Development work at Element level started before a central, comprehensive telescope architecture was established. The absence of a functional model, that allocates specific functions to Elements, meant uncertain scope and boundaries w.r.t. telescope M&C. The TM Consortium launched an M&C standardisation effort to set initial scope and boundaries for TM and its interfacing parties, and to generate an early, basic set of interface requirements for TM, to be refined later. This M&C standardisation reduced interface diversity from an M&C perspective and also produced guidelines to improve technical communication between Element consortia.

Approach

Activities in each phase of the Telescope development life cycle were identified and analysed. This analysis was done from the point of view of understanding whether TM needed to play a role in any of these activities.

The exercise helped us to identify the key TM functions, TM role in SKA operations and dependencies on other SKA systems. The list of TM responsibilities includes facilitating integration of Elements, telescope management, maintenance and usage.

Each TM function that involves interaction with other Elements of SKA was analysed to understand the information dependency between them. This helped to arrive at the appropriate information content and protocol that the interface between TM and the Elements needed to support. Few examples of such information are the self-description of the Elements provided to TM to facilitate their integration in SKA, information about their operational interface such as commands, events, alarms

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

and so on that would be used by TM to control them. It also described behavioural aspects like control models.

We saw the need to standardize information content and created the LMC interface guidelines document. We used available standards from the public domain such as X.731 towards this standardization activity.

The LMC interface guidelines document was circulated to all the Element consortia for their feedback. The Element representatives provided their feedback on the content of this guideline which was then subsequently accepted as the standard to be used by all Element LMC's for interfacing with TM.

Selection of TANGO as M&C Framework

Standardization effort for interfaces between TM and Element LMCs includes the following aspects: general principles, required functionality, format and content of the messages and communication infrastructure. As soon as the general requirements for these interfaces were identified, the SKA team started investigations related to the communication protocols and frameworks to be used for implementation. TM Consortium provided leadership in this area and invited all other SKA Consortia to identify, investigate and suggest candidate technologies. TM Consortium developed a comprehensive list of requirements and criteria for selection of the communication framework for communication between TM and other SKA Elements. The list of candidate technologies and criteria for selection were reviewed by all SKA Consortia that implement M&C interface with TM. In March 2015, representatives of all SKA Consortia met in Trieste, Italy to discuss the proposed technologies and criteria for selection. The programme included presentations by the experts for the proposed technologies. At the end of the meeting TANGO Control System (CS) was identified as a technology of choice for implementation of the M&C interfaces between TM and other SKA Elements.

The decision to use TANGO CS for interface between the Elements does not mean that TANGO CS will be used for M&C function internal to the Elements; but the report from the meeting recommends to all SKA Consortia to use TANGO CS internally where possible and suitable. TM itself will use TANGO CS for internal M&C.

INTERFACE DEFINITION AND TM REQUIREMENT ANALYSIS

From allocation of Telescope requirements to TM in [2], a functional structure for TM was derived. TM functions were specified by TM requirements in [1]. TM interactions with other Elements were covered in TM requirements by referring to interfaces from requirement text, for example, "The SKA1_MID TM shall send time stamped desired pointing coordinates to the Dish via its I.S1M.TM_DSH.001 interface". Up to now, SysML (refer [6]) representations of TM behaviour were limited to use case diagrams (Fig. 2) and swimlane activity diagrams (Fig. 5).

The SysML specification [6] provides a precise model based representation for interfaces and interface instance integration [5]. We are currently working towards modelling information exchange between the TM and other Elements as SysML object flows. Once functions that are allocated to TM and other Elements have been modelled as SysML activities, and the activities have been allocated to TM and Elements via SysML blocks, the object flow between the TM and Elements can be shown. Thus the systems behaviour (functions) and system structure (interfaces) can be brought together by tracing each object flow to the allocated interface port and connector uniquely identified by its interface number (see Fig. 3 & Fig. 4).

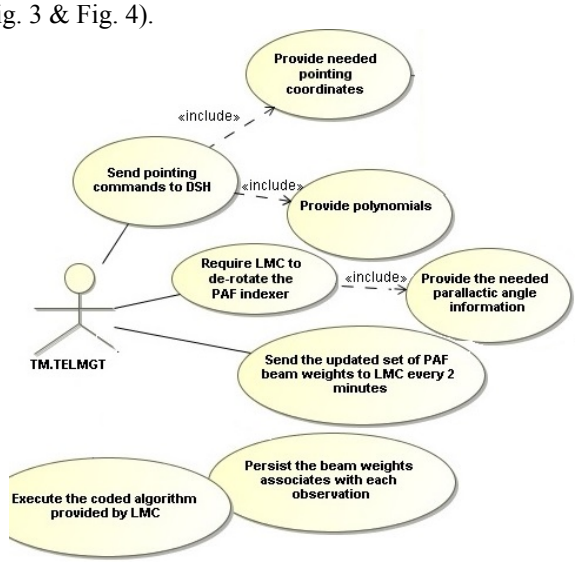


Figure 2: Dish pointing control use case diagram.

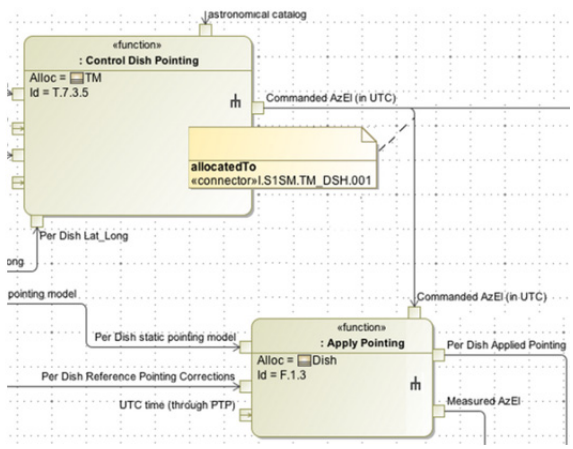


Figure 3: Behavioural view – interface and object flow.

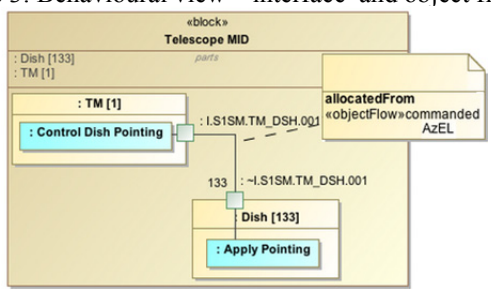


Figure 4: Structural view – interface & object flow.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

USER INTERFACE DEVELOPMENT

A requirement is a documented physical and functional need that a particular design, product or process must be able to perform. Interfaces provide the specifications of relevant properties of a system or component that can be connected to other systems or components together with their actual interaction. When users interact with a computer system, they do so via a UI.

Starting from the requirements as defined, collected and organized in [1] & [2], and the SKA Concept of Operations and Use Cases documentation provided by the SKA Office in [4], a users and tasks analysis phase followed in order to develop a solid conceptual model of stakeholders and of the interfaces involved in the operation of the SKA Observatory from a TM point of view. Documentation of SKA pathfinders and precursors on operational concepts, actors and systems has been considered too.

A project glossary was used to unambiguously define and describe actors, roles and tasks in the application domain. SysML use case diagrams were used to represent system scenarios and interactions workflows, then detailed out with via textual descriptions and swimlane activity diagrams to fully describe dynamic behaviour (see Fig. 5 an example of TM interactions with Dish and the operator in a specific scenario.

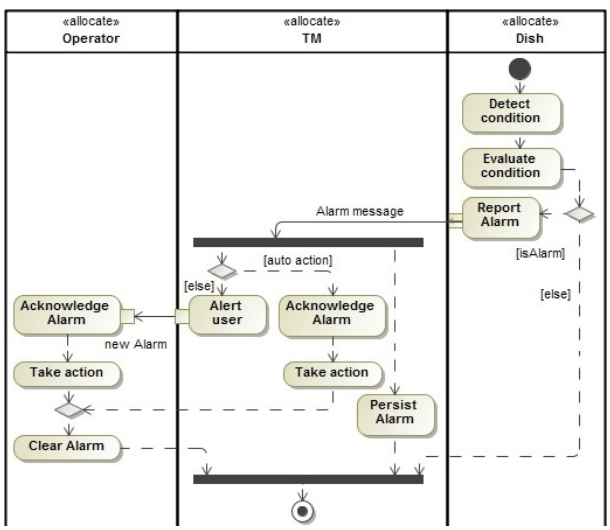


Figure 5: Alarm reporting SysML swimlane diagram.

A user-centred approach will be adopted in TM UI development. It will consist of several iterations, involving users' active contribution (interviews, brainstorming sessions, feedback) to fully understand user needs in performing the tasks that they have to perform while interacting with TM and the actual usage environment (organisational, social, and physical). User-centred design is hoped to optimize usability as Part 11 of the ISO 9241 standard [7] defines it as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."

Prototypes will be used to elicit detail requirements and highlight possible critical dependencies between different components, drive out technical gaps, get stakeholders and end users feedback, test common framework tools and libraries, interaction flows and common look and feel, evaluate usability and accessibility.

Validation interviews will be performed with a subset of real users or user proxies.

SUMMARY

The external interfaces of the TM Element of the SKA are diverse and many. Resulting complexity is handled by M&C separation of concerns and interface standardisation while SysML is used in models that integrate interface definition with requirements analysis, and supports UI analysis and design. M&C standardisation and UI design and development are two of the focus areas of the TM Consortium in interface management.

ACKNOWLEDGMENT

The authors wish to acknowledge all past and present members of the Telescope Manager Consortium.

REFERENCES

[1] P.S. Swart, G. le Roux, Y. Wadadekar, L. van den Heever, A. Bridger, J.C. Guzman, "SKA1 TM Requirement Specification", T0000-0000-RS-001, Revision F, July 2015.

[2] W. Turner, "SKA Phase 1 System (Level 1) Requirements Specification", SKA-TEL-SKO-0000008, Revision 4, September 2014.

[3] P.S. Swart, "SKA TM Design Report", T0000-0000-DR-001, Revision D, June 2015.

[4] R. McCool, "Technical Use Cases", SKA-TEL-SKO-0000016, Revision 1, August 2014; https://my.alfresco.com/share/s/nF-dp1kbQwC_GG388KuRJw

[5] E. Fosse, C.L. Delp, "Systems Engineering Interfaces: A Model Based Approach", in Proceedings of 2013 IEEE Aerospace Conference, IPAC'14, Big Sky, MT, USA (2013); http://dx.doi.org/10.1109/AERO.2013.6497322

[6] Object Management Group, "OMG Systems Modeling Language (OMG SysML TM), version 1.3," OMG, Tech. Rep. OMG document number formal/2012-06-01, June 2012; http://www.omg.org/spec/SysML/1.3/

[7] ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PYTHON SOFTWARE FOR MEASURING WAVELENGTH AT OPTICALLY PUMPED POLARIZED ION SOURCE (OPPIS)*

P. Kankiya†, J. Jamilkowski, Brookhaven National Laboratory, Upton, NY 11973 USA

Abstract

Often diagnostic tools are packaged with proprietary software and it is challenging to integrate with native environment. The HighFinesse Angstrom Wavemeter used at OPPIS experiment for laser wavelength measurement is controlled using commercial software not supported by RHIC style controls. This paper will describe the integration of such a complex system and use of python for cross platform data acquisition.

INTRODUCTION

The control system environment at BNL's Collider Accelerator Division is a rich repository of control system applications developed in-house and supports complicated supervisory controls and data acquisition tasks, although is limited to supporting these tasks in the Linux environment. Very rarely do we have a requirement to support systems that require software development in a non-UNIX based platforms. We have explored a few solutions in meeting this limitation [1]. In this paper a solution to integrate windows binaries using python is presented. By implementing this solution it will be possible to log data and generate alarms, use synoptic displays etc after gathering measurements from foreign binaries.

SYSTEM IN USE

The polarized beam for RHIC spin physics experimental program is produced in the Optically-Pumped Polarized H-Ion Source (OPPIS), the RHIC polarized H-ion source is upgraded to higher intensity (5-10 mA) and polarization by using a very high brightness fast atomic beam source for enhanced luminosity RHIC operation. Pulsed laser is used to optically pump the rubidium vapour. It is important to measure laser wavelength synchronous to AGS cycle to ensure polarization in desired direction [2]. A high precision device that can measure pulsed laser parameters with high accuracy at a fast rate is needed. To meet this requirement Armstrong meter model shown in Figure 1 was procured with a vision that it will be integrated into RHIC control applications.

Data Acquisition Procedure

Wavelength meter is setup in pulsed operation and externally triggered mode. These control set up tasks are performed by using proprietary software supplied with the hardware. Operator can provide preferred unit of measurement. Operation is started by adjusting the exposure manually or selecting automatic manner.

*Work supported by Brookhaven Science Associates, LLC under Contract No. DE-AC02-98CH10886 with the U.S. DOE

† pkankiya@bnl.gov

For the purpose of integrating measurement results from this device into custom applications, all program values and settings are accessible for user via calls to a dynamic linked library (DLL) called wlmData.dll routines. The wlmData.dll uses shared memory for inter-process communication. The software requires that the server (the Wavelength Meter application) and the clients (any access and control programs) access the same DLL file and not just identical copies which leads to limiting the use of the interface software to one instance at a given time. Elimination of restriction of use of DLL by one client at a time and integrating the whole setup with in-house controls is important to make use of existing infrastructure and correlate data measurements.

Figure 1: High Finesse Wavelength measuring device interfacing a windows computer and a light source.

SOLUTION

There are three common approaches to solving the aforementioned problem:

1. Write your own low level library to communicate with the device.
2. Write an extension in C code to bridge the gap between the libraries and use a cross platform compiler for generating supported executable format.
3. Wrap the library using your **foreign function interface** (FFI) support. Foreign refers to the fact that the functions come from another language and environment.

Because the wlmdata.dll is not an open source file, option one cannot be implemented. Option 2 is more time consuming and limitations has been addressed in another attempt described in paper [1]. Option 3 is considered to be a pragmatic solution. From multiple foreign function

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

interface (FFI) languages available python was considered for deploying interface software due to availability of knowledge in the field.

IMPLEMENTATION

A python module known as ctypes [3] allows easy integration of C libraries directly without writing wrapper code in C. With help of ctypes the concerned DLL is imported into a python program called WLM. A class named wavelength meter is defined. All functions of the DLL are available for execution once an object of this class is instantiated. The source code provided with the DLL package contains header files with error codes and function definitions. The header files are re-written to be python modules. The return types of each method must be explicitly specified and must match the original definition. By not complying to translating C header to ctypes exactly will lead to segmentation faults at runtime.

Data acquisition is made simple by executing routines such as callback (CallbackProc/Ex) and wait-event mechanisms (WaitForWLMEvent/Ex) which are prototyped one to one on the python side with help of ctypes. The mapped functions are called using standard C calling conventions as opposed to windows stdcall calling conventions [3]. These routines receive any measurement results and WLM state changing information.

The data collected from the python interface program can be then packaged and sent across Linux environment with help of RPC module and adoIf protocol implemented in python [4].

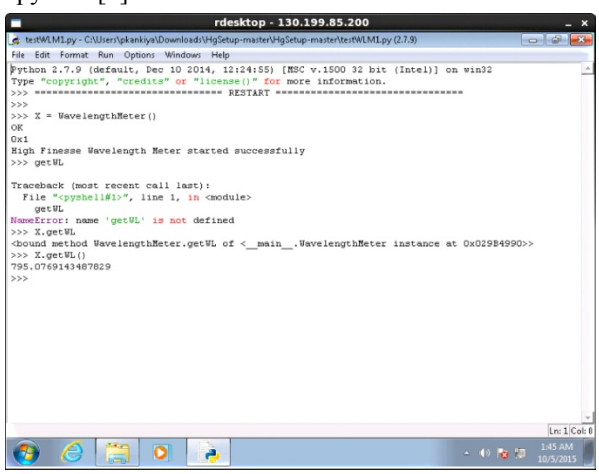


Figure 2: Simple object created from wavelength meter class and used to measure wavelength from get method.

DEVELOPMENT ENVIRONMENT

As described in a previous paper [1], to support C++ style ADO manager on windows we would have to port the existing code base on windows side with help of clear case remote client. Whereas, in case of what can be called a python manager a clearcase client is not required and any version control can be utilized due to non-dependency on legacy code. IDLE is used for editing and compilation of code. Figure 2 illustrates the use of python object to measure laser wavelength from an IDLE shell.

Python compilers are much lightweight applications as compared to CYGWIN like cross platform applications that have many OS dependencies and are complicated to install.

STATUS AND PLANS

A one to one map of wavelength meter package is developed in python with help of ctypes module. Measurement of data using the call-back and synchronous acquisition is tested. Work is under progress to port data to Linux side. This approach serves as a quickly deployable solution to integrating windows binaries to RHIC controls environment. For future projects it is advisable to find a solution such that developers do not have to fully repeat the C declarations. We plan to test this solution with operational data in the upcoming run of RHIC.

REFERENCES

[1] P. Kankiya, et al, “Integration of Windows binaries in the Unix-based RHIC control system environment.
[2] A. Zelenski, et al, “The RHIC polarised ion source upgrade with neutral ion source injector.
[3] https://docs.python.org/2/library/ctypes.html
[4] L.T. Hoff, J.F. Skelly, “Accelerator devices as persistent software objects”, Proc. ICALEPCS 93, Berlin, Germany, 1993.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONTROL SYSTEM FOR A DEDICATED ACCELERATOR FOR SACLA WIDE-BAND BEAM LINE

N. Hosoda, T. Fukui, T. Ohshima, T. Sakurai, H. Takebe#, RIKEN/SPring-8, Hyogo, Japan
M. Ishii, JASRI/SPring-8, Hyogo, Japan

Abstract

The X-ray free electron laser facility SACLA has provided opportunities for the exploration of new science. However, these opportunities are severely limited, because only one beam line (beam line 3) is currently available for user experiments. Therefore, there is an urgent need to increase the number of beam lines available. One approach is to install an additional accelerator. We decided to move the SCSS prototype accelerator to an upstream space of beam line 1 in the SACLA undulator hall and reused it as a dedicated accelerator for the beam line. We started an RF conditioning of the dedicated accelerator in October 2014 using MyCC, a MySQL-based temporary data acquisition system compatible with MADOCA. After the RF conditioning, we smoothly transitioned the system from MyCC to MADOCA. The beam commissioning of the dedicated accelerator started in September 2015, and we successfully observed the first extreme ultraviolet free electron laser in October 2015. The control system ensures the coordinated seamless operation of the SACLA accelerator and the dedicated accelerator.

INTRODUCTION

The X-ray free-electron laser (XFEL) facility SACLA (SPring-8 Angstrom Compact X-ray free electron LAser) aims to generate coherent X-ray beams with a wavelength of less than 0.1 nm [1]. It consists of a 400-m-long 8 GeV linear accelerator and a 120-m-long insertion device. In 2011, a commissioning of SACLA was started with beam line 3 (BL3). After the success of generating XFELs, the user experiments at BL3 started in 2012. The SACLA is providing opportunities for the exploration of new science.

Because an XFEL facility employs a linear accelerator, it provides an electron beam for only one beam line at a time. This means that its availability for user experiments is severely limited compared to that of a storage ring-type photon facility such as SPring-8, which has 57 beam lines. Therefore, there is an urgent need to allow a multi-beamline operation of SACLA.

One approach is to deliver an electron beam to multiple beamlines pulse-to-pulse using fast switching magnets [2]. Originally, a DC bending magnet at the end of the accelerator was used to switch an electron beam path between beam lines. We replaced it with a fast kicker magnet and DC twin septum magnets in January 2015. As a result, we successively injected electron beams to beam line 2 (BL2) and BL3 and achieved simultaneous lasing in both beam lines.

Another approach is to install an additional accelerator. Prior to the construction of SACLA, we built a SCSS (SPring-8 Compact SASE Source) prototype accelerator in 2005 to verify the feasibility of lasing using a thermionic gun [3]. We achieved the generation of SASE FEL with a wavelength of 50–60 nm in an extreme ultraviolet (EUV) region in 2007. After the initial purpose was accomplished, it was used for user experiments until 2013. We decided to move the SCSS prototype accelerator to an upstream space of beam line 1 (BL1) in the SACLA undulator hall, and reused it as a dedicated accelerator for BL1 [4].

The SACLA accelerator and the dedicated accelerator share one beam line, BL1. We had to coordinate both accelerator operations, while the independence was maintained as much as possible.

When we designed a control system for the dedicated accelerator, it was natural to use the MADOCA (Message And Database Oriented Control Architecture) control framework, which was already used at the SCSS prototype accelerator and the SACLA [5–7].

The requirements for the control system were as follows. The first priority was not to disturb SACLA user experiments. The RF conditioning was tightly scheduled to start in October 2014 and the beam commissioning was scheduled in September 2015. The human resources available to develop the system were limited. The period in which we could enter the undulator hall to test equipment was also limited. Moreover, the dedicated accelerator was operated at the SACLA control room.

Our strategies were as follows. We reused all control software developed for SACLA. At the start of the RF conditioning, we used a temporary data acquisition (DAQ) system compatible with MADOCA, because some RF components were not installed. After the RF conditioning and a completion of the construction, we switched the DAQ system from MyCC to MADOCA.

In this paper, the status of the dedicated accelerator is reported in terms of its control system.

THE DEDICATED ACCELERATOR FOR BL1

Figure 1 shows a schematic view of the dedicated accelerator for BL1 at the SACLA facility. Although we planned for five beam lines in the undulator hall, only three beam lines had been constructed until now. The bending angle of the transport lines from SACLA to BL1 is 3 degrees and BL1 is located 6 m apart from the SACLA accelerator in parallel. Therefore, a 110-m-long vacant space exists upstream of BL1. The dedicated accelerator was placed in that space.

Present address: Okinawa Institute of Science and Technology (OIST)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

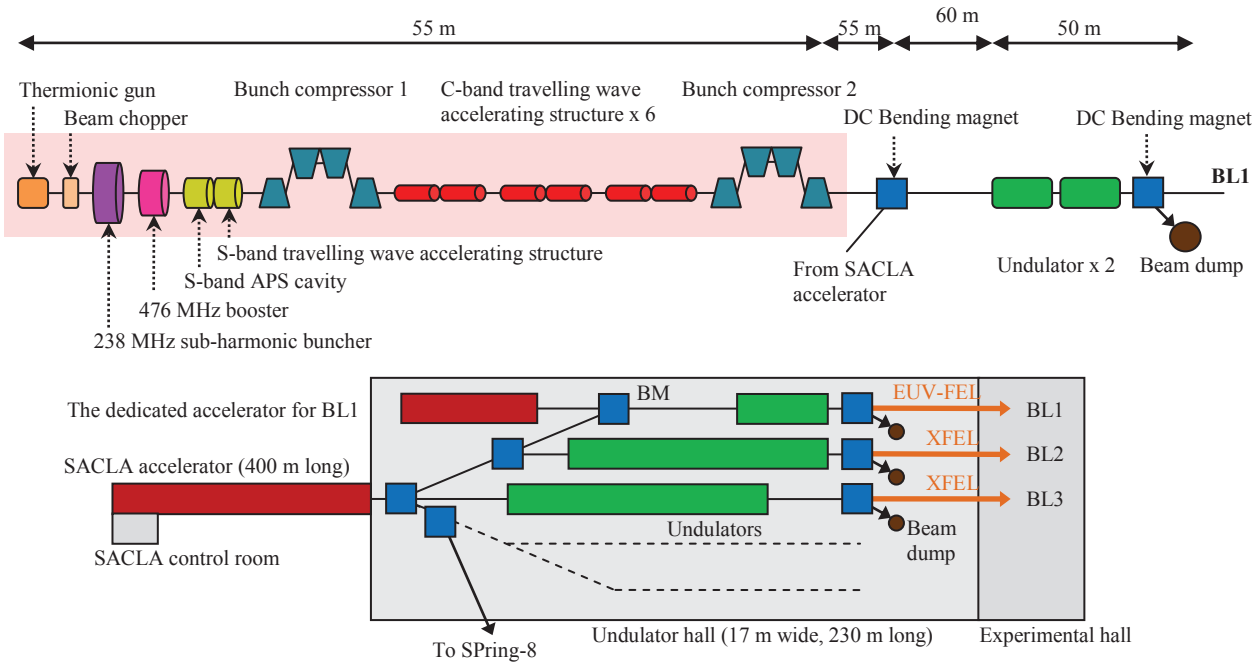


Figure 1: Schematic view of the dedicated accelerator for BL1 at the SACLA facility.

All RF components except C-band were reused from the SCSS prototype accelerator. The C-band accelerating structure was newly designed to increase the beam energy per unit length. We installed 3 C-band units at this stage. All low level RF (LLRF) modules, timing modules, and monitor modules, such as the beam position monitor and beam current monitor, were replaced by modules developed for SACLA to achieve higher stability and to standardize their control system. The design parameters of the dedicated accelerator are listed in Table 1. At the downstream of bunch compressor 2, a space of 9 C-band units is reserved for a future upgrade. If 9 C-band units are added, electron beam energy will reach 1.4 GeV and we can provide soft X-ray FEL to users.

Table 1: Design Parameters of the Dedicated Accelerator

Accelerator	Beam energy	420 MeV
	Bunch charge	0.3 nC
	Peak current	300 A
	Repetition rate	60 Hz
Undulator	Periodic length	18 mm
	K parameter	2.1 (max)
	Number of periods	460
FEL	Wavelength	42 nm (K = 2.1)
	Pulse energy	100 uJ

We had to concentrate the construction work in the undulator hall to summer shutdown periods, because we could not enter it during the user experiment periods of SACLA. In summer 2013, the injector part from an electron gun to bunch compressor 1 was installed. The remaining part was installed in summer 2014. However, works outside of the undulator hall (for example, the installation of 19-inch racks containing the VME systems, programmable logic controllers (PLCs), and power supplies) were performed during user experiments on the condition that the works did not disturb user experiments.

EQUIPMENT CONTROL HARDWARE

VME System

We used the CPU board XVB601 (Intel Core i7), manufactured by GE, for the LLRF component, and SVA041 (Intel Pentium M), manufactured by Sanritz, for the other components.

The high speed A/D board MVD-ADC01 (238-MHz sampling, 16-bit resolution, and 4-channel input), manufactured by MELOS, is used for RF phase/amplitude detection of the RF components and readout of the beam position monitor/beam current monitor. The high speed D/A board MVD-DAC02 (238-MHz sampling, 16-bit resolution, and 4-channel output), manufactured by MELOS, is used to control the phase/amplitude of the driving RF signal of a klystron using an IQ modulator. The high resolution A/D board MVD-ADC04 (1-MHz sampling, 24-bit resolution, and 4-channel input), manufactured by MELOS, is used for a CSR bunch length monitor.

An 8-channel trigger delay unit (TDU), manufactured by MELOS, generates very low jitter trigger signals from a 60 Hz master trigger signal distributed via the SACLA master trigger/reference RF clock distribution system. This means the dedicated accelerator and SACLA work synchronously. The TDU has a function to count the number of master trigger inputs, and the function is used

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

to add a tag number to each A/D data in an event-synchronized DAQ process [8]. In the VME system without a TDU, an interrupt register board (INTREG) Axvme4900, manufactured by ARKUS, counts the number of master trigger signals.

To control the power supplies of the 97 magnets in total of the dedicated accelerator, an optical remote I/O system OPT-VME/OPT-DIO, manufactured by Hitz, is used. The OPT-VME is a master board and can control 4 OPT-DIO slave boards. Most of the OPT-VME/OPT-DIO boards used at the prototype accelerator were reused after cleaning and inspection to ensure that there was no degradation of quality.

To access PLC, we use a VME-based FL-net interface board. FL-net is one of the Ethernet-based open standard protocols for a factory floor network authorized by the consortium in Japan.

We prepared 24 VME systems for the dedicated accelerator. Table 2 lists the number of VME boards used. In basic terms, each VME system is located in the south klystron gallery near the equipment. However, the VME systems for FL-net are placed in the SACLA control room because that is the only network which requires it.

Table 2: Number of VME Boards for the Dedicated Accelerator

	CPU	MVD-ADC01	MVD-DAC02	TDU	MVD-ADC04	INTREG	OPT-VME	FL-net
LLRF	6	17	6	6	0	0	0	0
HPRF	2	0	0	0	0	0	0	4
monitor	4	11	0	0	1	3	0	1
timing	2	0	0	1	0	0	0	1
magnet	6	0	0	0	0	0	15	0
vacuum	2	0	0	0	0	0	0	3
utility	2	0	0	0	0	0	0	4
Total	24	28	6	7	1	3	15	13

PLC

PLCs are used where slow and rigid equipment control is required; for example, in the high voltage power supply of a klystron (HPRF), the cooling water system, the vacuum system, and the environmental parameter monitor of a 19-inch rack system. As mentioned above, PLCs are remotely controlled via the FL-net VME interface board. In addition, we prepared programmable graphic panels near the PLCs for local control.

Interlock System

The interlock system of SACLA consists of a personal protection system for human radiation safety, a machine protection system against the failure of the vacuum components or the magnet power supply, and a beam operation interlock system for the management of the beam route and the amount of electron charge/energy.

The beam operation interlock system for the dedicated accelerator was installed separately from the system for SACLA. However, both systems share the excitation current data of the DC bending magnet, located at a junction point of the two accelerators to manage which accelerator has the permission to inject an electron beam to BL1.

The machine protection system was modified to manage which gate valves should be closed if a vacuum leak occurred in one accelerator, because the user experiments of the other accelerator must not be disturbed.

SOFTWARE

Equipment Control

The operating system of the VME CPU is Solaris 11 (64 bit) for the LLRF component and Solaris 10 (32 bit) for the other components. In the VME system, three basic MADOCA processes are running. These are: a message server process which receives control messages from operator consoles, an equipment manager process which accesses equipment, and a poller process which collects data at a constant frequency, such as at 0.5 Hz. In LLRFs and the monitor VME, an event-synchronized DAQ process which collects data for all beams shot at 30 pps together with an event tag number is also running. Furthermore, in the LLRF VME, a PID feedback process to stabilize the RF phase/amplitude and an abnormal RF waveform acquisition process are running [9]. Although we are required to run such high-load processes in the LLRF VME, the VME system is still working well.

Database

As indicated by the unabbreviated name of MADOCA, the database is indispensable. The Sybase RDBMS is the main database in SACLA [10]. We use it for poller data logging, signal management, equipment setting parameter management, and equipment alarm management. We decided to treat the dedicated accelerator data in the SACLA database.

In October 2014 while SACLA user experiments were ongoing, we needed to start an RF conditioning of the dedicated accelerator. Before the start, new equipment data and tables related to the dedicated accelerator had to be added to the database. However, we did not want to do this because the smallest possibility of a disruption to user experiments caused by a mistake in database operation always existed.

We had developed a simple DAQ system, MyCC, which is compatible with MADOCA [11]. The target of MyCC is DAQ of a small number of signals in a limited acquisition period. This allows for frequent modification of the equipment system. The features of MyCC are as follows. A MySQL database dedicated for MyCC is used instead of Sybase. The three basic MADOCA processes on the VME CPU are used in MyCC without any change. The MyCC signal registration list is created from the MADOCA signal registration list. This means that if we

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

finish the configurations of the MyCC signal registration list, we do not have to modify the MADOCA signal registration list. The MyCC database API has the exact same function calls as those of the MADOCA database API. Therefore, without any source code modification, an operation GUI application works for both MyCC and MADOCA, with the only relinking the API libraries. For these reasons, we adopted MyCC for the RF conditioning.

About 2,800 signals were registered in MyCC, and the RF conditioning was started on schedule. In January 2015 during a winter shutdown, the DAQ was smoothly transitioned from MyCC to MADOCA using a proof signal registration list. The RF conditioning lasted until July 2015 and achieved a sufficient acceleration gradient and an acceptable RF trip rate.

In addition to the Sybase server, a MySQL server for the event-synchronized DAQ and a Cassandra server for the abnormal RF waveform acquisition are also used in SACLA. In August 2015 during a summer shutdown, we registered equipment signals to these servers. Table 3 lists the number of equipment signals registered in the databases.

Table 3: Number of Equipment Signals Registered in the Databases

	SACLA		The dedicated accelerator	
	Sybase	MySQL	Sybase	MySQL
Analogue data	43,668	1,612	2,519	103
Digital data	16,130	902	1,103	62
Total	59,818	2,514	3,622	165

Graphical User Interface (GUI)

The SACLA is operated at the SACLA control room. We used nine PCs (IA architecture, SUSE Linux Enterprise, dual displays) for the operator consoles. Operation GUIs were developed using X-mate, a GUI development support tool based on X-Window.

Because the SACLA accelerator and the dedicated accelerator had to be operated seamlessly, we developed the integrated operation GUIs.

Figure 2 shows an operation GUI for the magnet power supplies. When a section of the overall view in the upper part is selected, a detailed view of the section is displayed in the lower part. To decrease the chance of mistakes in operation, the background color of the detail view is changed to clarify which accelerator is selected. The equipment parameters of BL1, such as the magnet power supply, timing delay, and beam monitors, differ between the SACLA accelerator and the dedicated accelerator. Therefore, the GUI was developed to select appropriate parameters using beam route information.

Figure 2: Operation GUI for magnet power supplies.

CONCLUSION

We constructed the control system of the dedicated accelerator for BL1 at SACLA. The MySQL-based temporary data acquisition system, MyCC, enabled the RF conditioning of the dedicated accelerator without affecting SACLA user experiments. After the transition from MyCC to MADOCA and the completion of the preparation for the all-control system, the commissioning of the dedicated accelerator started in September 2015. We successfully observed the first EUV-FEL at BL1 using the dedicated accelerator in October 2015. The control system ensures the coordinated seamless operation of the SACLA accelerator and the dedicated accelerator. We will start user experiments at BL1 before March 2016.

REFERENCES

[1] T. Ishikawa, et al., “A compact X-ray free-electron laser emitting in the sub-angstrom region”, Nature Photon. 6, 540–544, (2012).

[2] T. Hara, et al., “Pulse by pulse electron beam distribution for multi-beamline operation at SACLA”, FEL2014, Basel, Switzerland (2014).

[3] T. Shintake, et al., “A compact free-electron laser for generating coherent radiation in the extreme ultraviolet region”, Nature Photon 2, 555–559, (2008).

[4] Y. Otake, et al., “Relocation and improvement status of the SCSS test accelerator to provide dual FEL drivers at SACLA”, IPAC15, Richmond, USA, (2015).

[5] T. Fukui, et al., “Status of the SCSS control system – first phase of an 8 GeV FEL project in SPring-8”, ICALEPCS2005, Geneva, Switzerland, (2005).

[6] T. Fukui, et al., “Status of the X-ray FEL control system at SPring-8”, ICALEPCS2007, Tennessee, USA, (2007).

[7] R. Tanaka et al., “Inauguration of the XFEL facility, SACLA, in SPring-8”, ICALEPCS2011, Grenoble, France, (2011).

[8] M. Yamaga et al., “Event-synchronized data-acquisition system for SPring-8 XFEL”, ICALEPCS2009, Kobe, Japan, (2009).

[9] M. Ishii et al., “A Data Acquisition System for Abnormal RF Waveforms at SACLA”, ICALEPCS2015, Melbourne, Australia, (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

[10] T. Hirono et al., “Scaling up of the MADOCA database system for SACLA”, ICALEPCS2013, San Francisco, USA, (2013).
[11] T. Maruyama et al., “Development of myCC for a simple data acquisition system compatible with MADOCA”, 10th Annual Meeting of Particle Accelerator Society of Japan, Nagoya, Japan, (2013).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

STATUS OF THE PAL-XFEL CONTROL SYSTEM

Changbum Kim*, Geon-Yeong Mun, Soungyoul Beak, Youngjin Seo, Dong Cheol Shin,
Byoung Ryul Park, Woul-Woo Lee, Jihwa Kim, Heung-Sik Kang, and In Soo Ko,
Pohang Accelerator Laboratory, Pohang, Kyungbuk, Korea

Abstract

Pohang accelerator laboratory (PAL) started an x-ray free electron laser project (PAL-XFEL) in 2011 [1]. In the PAL-XFEL, an electron beam with 200 pC will be generated from a photocathode RF gun and will be accelerated to 10 GeV by using a linear accelerator. The electron beam will pass through undulator section to produce hard x-ray radiation. In 2015, we will finish the installation and will start a commissioning of the PAL-XFEL. In this paper, we introduce the PAL-XFEL and explain present status of it. Details of the control system will be described including a network system, a timing system, hardware control systems and a machine interlock system.

PAL-XFEL

Figure 1 shows a bird eye's view of the PAL facilities which is taken in the April 2015. A circular building in right hand side is the storage ring of the Pohang Light Source (PLS-II) and a new long building on the left hand side is the PAL-XFEL. The building construction was started in 2012 and it was finished at the end of 2014.

The length of the PAL-XFEL building is 1.11km which is consist of a 700 m long linear accelerator, a 250 m long undulator hall, and another 60 m long beam line hall. 10 GeV electron beam can be obtained from the 700 m long linear accelerator and 0.1 nm wavelength hard x-ray can be generated from the electron beam by using the self amplification of the spontaneous emission (SASE).

The total budget is 400 million US dollar and the project period is from 2011 to 2015. Installation of the components of the accelerator was started from the beginning of 2015.

Figure 1: Pohang Accelerator Laboratory.

*chbkim@postech.ac.kr

Figure 2: Pohang Accelerator Laboratory.

Figure 3: Pohang Accelerator Laboratory.

Figures 2 and 3 show the klystron and modulator gallery and inside of the linear accelerator tunnel, respectively. 51 klystrons and modulators and 174 accelerating columns will be installed up to end of 2015.

CONTROL SYSTEM

The PAL-XFEL control covers a large range of hardware and software. An Experimental Physics and Industrial Control System (EPICS) will be used for the control system of the PAL-XFEL. A powerful network system was installed and a large number of Input Output Controllers (IOCs), for example about 700 magnet power supplies, will be connected to a high level control system. An event timing system will be used to send trigger signals to all the equipment and RF distribution system will be ready to supply the 2856 MHz RF signal for S-band RF stations. In addition, a machine interlock system will be prepared to protect important equipment.

Figure 4 shows the network system layout and core switches with Fiber Distribution Frames (FDFs). Duplicated core switches were installed for safety and local switches were connected to them by using 1 Gigabit or 10 Gigabit bandwidth optical fiber.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Figure 4: Network core switches and FDFs.

An event timing system will be used to generate the trigger signal. Figures 5 and 6 show layout of the event timing system and the event generator, respectively. 476 MHz signal from RF distribution system and 360 Hz signal from AC power are provided to a VMTG to make 119 MHz which is synchronized with AC power. 119 MHz and 360 Hz signal go to the event generator to produce 60 Hz trigger signal.

Figure 7 shows layout of the RF distribution system. To generate RF power from 51 klystrons and modulators, 2856 MHz signal should be provided to them, however, such a high frequency can have serious loss in a long RF coaxial cable, so that 479 MHz signal will be generated from a Dielectric Resonator Oscillator (DRO) and sent along the linear accelerator by using a RF coaxial cable.

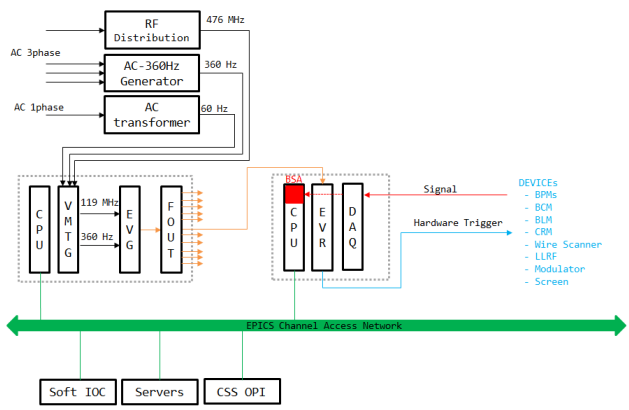


Figure 5: Event timing layout.

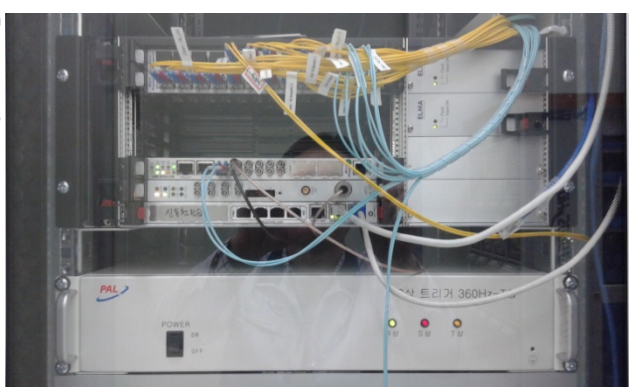


Figure 6: PAL-XFEL event generator.

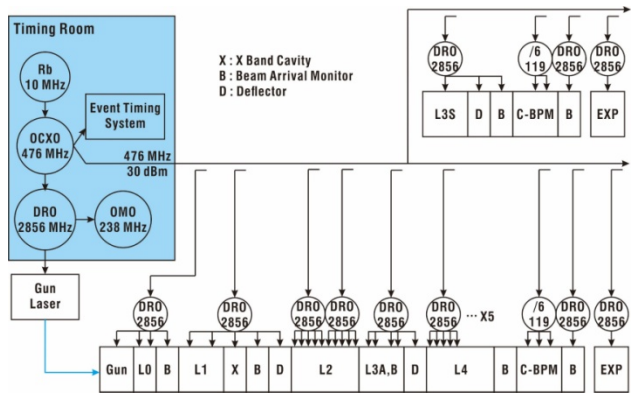


Figure 7: RF distribution system.

The RF coaxial cable will be installed inside the temperature controlled box on a tray along the linear accelerator.

Figure 8 shows modulator with its controller. Yokugawa Programmable Logic Controllers (PLCs) are used for modulator control and an EPICS module is installed to make an EPICS IOC.

Figure 9 and 10 show installed undulators inside the undulator hall and its control layout, respectively. A Microsoft Windows based, real time capability, embedded pc controller is connected by using an EtherCAT network to a server motor driver which moves motors of the undulator.

Figure 8: Modulator and its control system.

Figure 9: Installed undulator in undulator hall.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

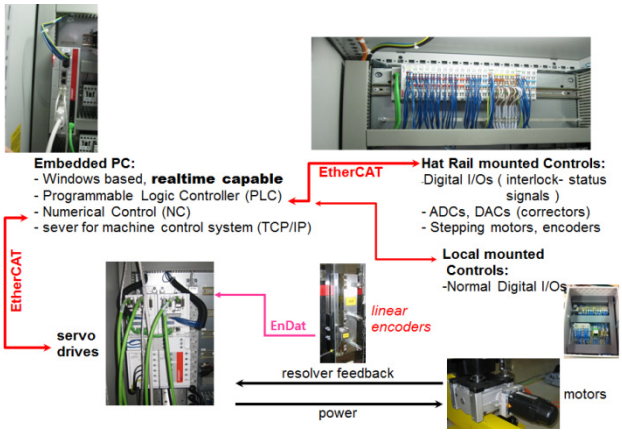


Figure 10: Undualtor control layout.

Figure 11 shows an uTCA based Beam-Position-Monitor (BPM) electronics. The BPM electronics consist of a chassis, a central processing unit, a channel manager, a network card, an event receiver, and analogue digital converters with rear transition modules. BPM electronics can measure the beam position with a time stamp which is generated by event timing system to make a beam synchronous acquisition.

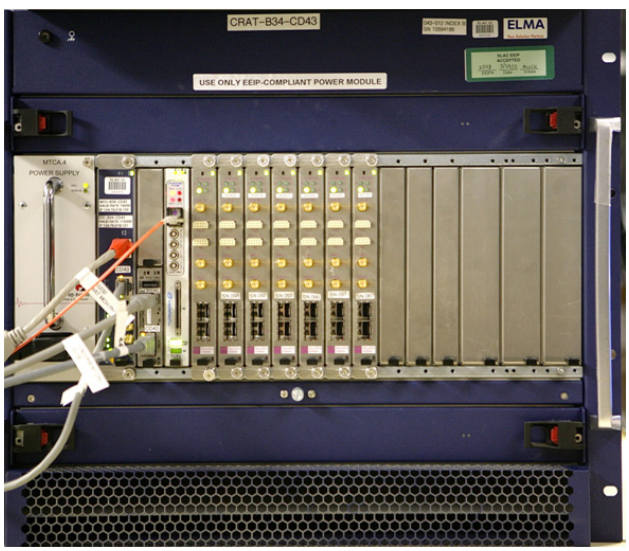


Figure 11: uTCA based BPM electronics.

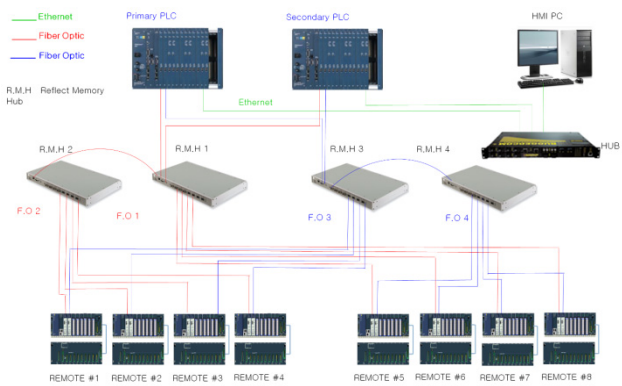


Figure 12: Machine interlock system layout

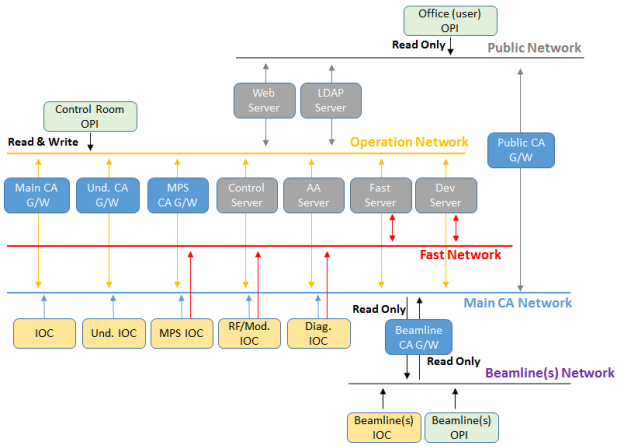


Figure 13: Control server connection.

Figure 12 shows layout of the machine interlock system. The machine interlock system will make interlock signal to protect critical component such as permanent magnets of an undulator which should be protected from a high energy electron beam. The machine interlock system will have duplicated main PLCs and local PLCs which are connected by optical fiber to reduce the communication time.

Figure 13 shows control servers and their network connections. There will be a public, an operation, a main channel access, a fast data, and a beamline network. Operation status can be monitored by using a web server and outside access will be controlled by a public gate way in the public network. Operation interfaces in the control room will be connected to control servers and gateways by using the control network. Local IOCs will be connected by main channel access network and protected by gateways from direct access of operators or outside users. The fast data network will be prepared for 60 Hz real time data transfer between IOCs and the fast data server.

SUMMARY

The PAL-XFEL will be installed until the end of 2015 and start commissioning in the beginning of 2016. After 1 year of commissioning it will be opened to public for user experiments. From the network system to the various kinds of servers, the control system of the PAL-XFEL is under construction and will be ready for the successful commissioning of the PAL-XFEL.

ACKNOWLEDGMENT

Authors would like to thanks to the collaborators such as the SLAC for the event timing system, the BPM electronics and the Cosylab for the high level control system.

REFERENCES

[1] H.-S. Kang et al., “Status of the PAL-XFEL Construction”, in proceedings of IPAC2015, Richmond, VA, USA (2015), p 2439.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

XFEL MACHINE PROTECTION SYSTEM (MPS) BASED ON UTCA

S. Karstensen, M. E. Castro Carballo, J. M. Jäger, M. Staack (DESY, Hamburg).

Abstract

For the operation of a machine like the linear accelerator XFEL at DESY Hamburg, a safety system keeping the beam from damaging components is obligatory. This machine protection system (MPS) must detect failures of the RF system, magnets, and other critical components in various sections of the XFEL as well as monitor beam and dark current losses, and react in an appropriate way by limiting average beam power, dumping parts of the macro-pulse, or, in the worst case, shutting down the whole accelerator. It has to consider the influence of various machine modes selected by the timing system. The MPS provides the operators with clear indications of error sources, and offers for every input channel the possibility to set dedicated machine modes to facilitate the operation of the machine. In addition, redundant installation of critical MPS components will help to avoid unnecessary downtime.

This paper summarizes the requirements on the machine protection system and includes plans for its architecture and for needed hardware components. It will show up the clear way of configuring this system - not programming.

Also a look into the financial aspects (manpower / maintenance / integration) is presented.

INTRODUCTION

The European X-Ray Free Electron Laser (XFEL) linear accelerator will bring an electron beam to the energy of up to 20 GeV and use it to generate extremely brilliant pulses of spatially coherent x-rays in an array of undulators using the Self-Amplified Spontaneous Emission (SASE) process. With a design average beam power of 600 kW and beam spot sizes down to few micrometers, the machine will hold a serious damage potential unless countermeasures are taken. To ensure safe operation of the accelerator, dangerous situations have to be detected by closely monitoring beam losses and the status of critical components, and to react appropriately. This is the task of the fast machine protection system (MPS) described in this paper. Several design features of the system have been influenced by experience from existing facilities, particularly the Free Electron Laser in Hamburg (FLASH).

A high flexibility of the MPS is essential to guarantee minimum downtime of the accelerator. In contrast to a storage ring where a beam dump typically implies a time-consuming refill of the machine, a linac offers the possibility to limit the length of the bunch train individually for each macro-pulse. Hence the reaction to failures of subsystems or even parts of the MPS can be much more specific—a dynamic limitation of the total beam power or a selective veto on beam transport into a particular branch of the beamline are possible. Experience with FLASH has also shown that the operation of the machine profits from an MPS whose behaviour can be changed or extended in a simple way. Apart from all of this, overall limitations, like power density, have to be respected.

XFEL ARCHITECTURE

The present chapter provides a brief overview of the XFEL facility (Figure 1). The major tunnel sections accommodate the following systems:

- injector
- linear accelerator (linac)
- beam distribution system
- undulators
- photon beam lines
- experimental stations

These components are distributed along an essentially linear geometry, 3.4 km long, starting on the Deutsches Elektronen-Synchrotron (DESY) laboratory campus in the north west part of the city of Hamburg, and ending in the neighboring federal state of Schleswig-Holstein, south of the city of Schenefeld, where the experimental hall will be located.

In the injector, electron bunches are extracted from a photocathode by a laser beam, focused and accelerated in a radio frequency (RF) gun and a superconducting acceleration module, and directed towards the linac with an exit energy of 120 MeV. After further acceleration, the bunches are longitudinally compressed in two bunch compressors, BC1 and BC2, at energies of 500 MeV and 2 GeV. In the subsequent main linac, the beam is brought to energies of up to 20 GeV (17.5 GeV is the energy foreseen for normal operation of the XFEL) before passing the collimation section. Afterwards, a fast kicker can send single bunches into a beam dump and the beam will be limited. The remaining bunch train can be sent into two undulator lines by the beam distribution kicker with a rise time of less than 20 µs. Each of the undulator lines ends in an electron beam dump, and each of the three main beam dumps is designed to withstand only half of the nominal beam power, i.e. 300 kW.

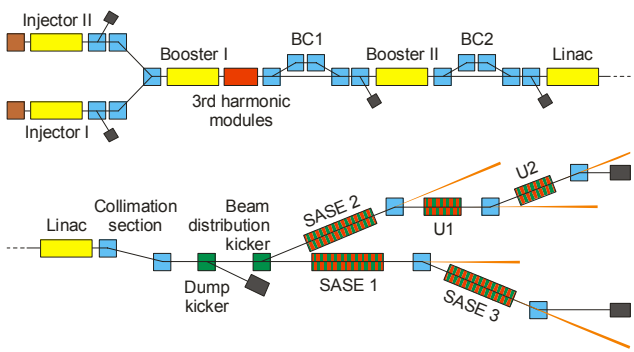


Figure 1: Schematic of the XFEL beam line.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REQUIREMENTS FOR THE MACHINE PROTECTION SYSTEM

The main requirements for the MPS may be summarizing in three points, in the order of their importance:

1. protect accelerator components and devices in the vicinity of the accelerator from direct and indirect damage by the beam
2. facilitate easy handling of the machine—impair machine operation only if necessary
3. limit activation of accelerator components to preserve their maintainability

While it is obvious that the protection from damage is of paramount priority, any machine can only serve its purpose if it can be operated. Beam time at the XFEL will be in high demand, and the goal should be to limit downtimes to their necessary extent. In this respect, it is of no importance whether downtimes are caused by hardware failures or operating errors; the MPS should be both highly reliable and "user-friendly".

Failsafe Behavior

Since most of the MPS electronics will be located in the accelerator tunnel, an elevated radiation background must be expected. An analysis on the behavior of FPGAs has been carried out at DESY [1] and shows that neutron-induced single event upsets (SEUs) are the major source of malfunctions. Therefore, the design of the electronics should ensure as far as possible that SEUs do not lead to unsafe or uncontrolled behavior of the system. Problems caused by power cuts or simple cable breaks must also be considered.

Reaction Times

In the XFEL, the distance from the injector lasers to the last undulators is approximately 3 km. Thus, at the speed of light, a signal needs about 10 µs to travel from one end to the other. Since the bunch frequency of the XFEL is 5 MHz, a maximum of 50 bunches could be moving within the accelerator at any given time.

Table 1: Minimum Number of Lost Bunches at Various Locations, According to Reaction Times of MPS, Assuming a Signal Velocity of 2/3 c

Beam loss location	Distance from injector	Distance from dump kicker	min. num. of lost bunches
Injector	0 m	–1970 m	0
bunch compr. 1	160 m	–1810 m	7
bunch compr. 2	360 m	–1610 m	15
linac center	1040 m	–930 m	44
linac end	1650 m	–320 m	69
beam distribution kicker	2010 m	40 m	2
last undulator	3010m	1040m	44

From this follows that, if beam losses are detected near the end of the machine, at least 100 bunches carrying a total energy of about 2.2 kJ would arrive at the position of the loss before the injector laser could be switched off.

To reduce this reaction time, the MPS clearly needs a second location for disposing the beam. The dump kicker, part of the beam switchyard at about 2.1 km along the machine, is the natural choice for this interaction. Table 1 lists the minimum possible reaction times and the minimum number of lost bunches at several possible locations of MPS alarms.

MPS ARCHITECTURE

The large scale of the XFEL imposes a severe technical issue: latency of electronics and signal transport speed of 3/4c in copper cables and 2/3c in optical fibers lead to a signal delay and in consequence additional lost bunches. To provide a short reaction time, the MPS implements a distributed Master/Slave architecture keeping short distances between components. Also every Master/Slave does a pre-processing of all incoming alarm signals that we have a here a real distributed system with distributed processing units.

The optical fibers are planned in a way that a set of fibers connects each RF section (4 cryogenic modules, 1 klystron) to the injector building XSE. Various points in the undulator sections are connected to the hall XS1, fiber sets from the experimental stations are collected in the hall XHEXP1. There are fiber connections between XSE, XS1 and XHEXP1. Each of the RF sections will be equipped with 2 MPS modules (Figure 2). Loops in the bunch compressors and SASE sections will contain 3 to 5 MPS modules.

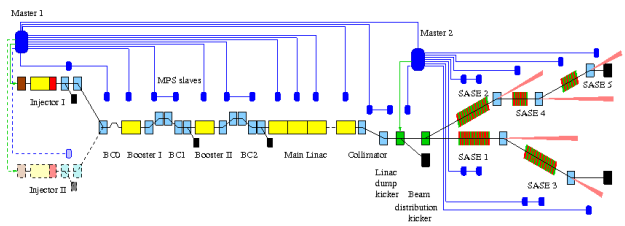


Figure 2: MPS distributed in XFEL.

The backbone of the XFEL Machine Protection System consists of roughly 130 MPS slave modules distributed along the machine. Each of these modules has digital inputs for "beam OK" signals from critical subsystems, accelerator components, or beam loss detection hardware. It also has fast serial input and output ports to connect to other MPS modules; to avoid problems with electromagnetic interference, these serial connections use fibre optic cables.

All modules are developed to provide high flexibility. They could be used as an intelligent data distribution knot, as standalone knot or just as a simple data collector, with single or redundant communication links.

Two master modules are located near the injector and near the linac dump kicker (Figure 2). The FPGA logic of these "MPS masters" defines the behavior of the system.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

They have direct connections to the injector lasers and to the dump kicker, allowing them to stop the production of new bunches and to dump bunches that are already in the machine. Both Masters are working very close together and can be seen as one Master. The latency between the two masters is negligible, as both masters have different tasks and the information interchange is only informal.

Referring to the XFEL Timing System [2], the complete MPS system is working asynchronously.

The serial connections between MPS masters and slaves form loops carrying a steady data stream. Within reasonable limits, the number of loops connected to each master can be chosen freely, while the number of slave modules in each loop is limited mainly by the desired latency of less than 1 µs excluding cable delays.

Simulations have shown that with current FPGA technology, the input latency per board is in the range of 42 ns (RS 422 input → FPGA).

Due to the used structure, the system is also easily scalable without major cabling work as of new slaves could be added to existing knots.

INTERFACES

The MPS slaves receive digital status signals from external systems via RTM / RS422 signal lines and information of the Timing System via the µTCA backplane. A cable break or short circuit within the RS422 lines will be detected and reported as an alarm without distinguishing between real alarm or cable break.

The main MPS master is connected to the Timing System to provide the information about possible Beam Modes (Figure 3) and Section patterns (Figure 4). Beam Modes representing the maximum allowed bunches and power within the linac sections (1 Bunch, single, medium, full). Section Patterns are showing the health status of sections. Also, since MPS slaves and masters constantly communicate with each other, this communication is watched by special algorithms to guaranty a failsafe operation.

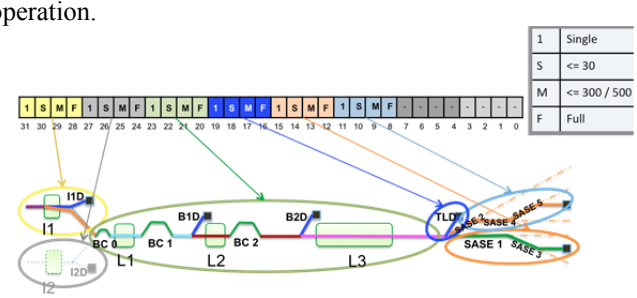


Figure 3: XFEL Beam Modes.

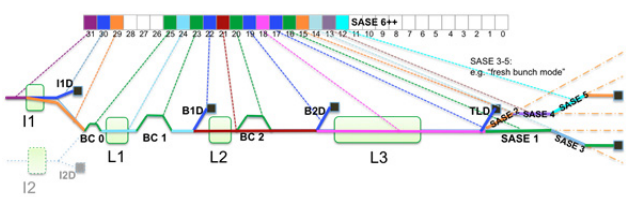


Figure 4: XFEL Section Patterns.

All incoming alarm signals are handled equally. There is no difference between slow (e.g. relay) or fast (e.g. electronic) signals.

µTCA TECHNOLOGY

MPS as most of the control components of XFEL will use the µTCA technology for the following reasons:

- Modular + modern architecture
- Reusability + PCIe + Ethernet
- High availability
- Redundant power and fan optional
- Well defined management protocoll
- High performance:
  - Very low analog distortions
  - 4 lanes PCIe: 400 MB/s ... 3.2 GB/s

XFEL fast electronics will be based on MTCA.4: with more than 200 Crates [3]

HARDWARE

The Master and Slave stations are equipped with the DESY DAMC2 (FPGA, PCIe, SFP) board and the appropriate Rear Transition Module (RTM) (galvanic separated RS422 IOs).

This strategy offers a large flexibility in case of technical problems as the DAMC2 is used everywhere inside the XFEL and MPS will take profit out of external developments, soft- and hardware wise. To receive digital alarm signals at DAMC2, a Rear Transition Module (RTM) has been developed for MPS. A schematic diagram is shown in Figure 5.

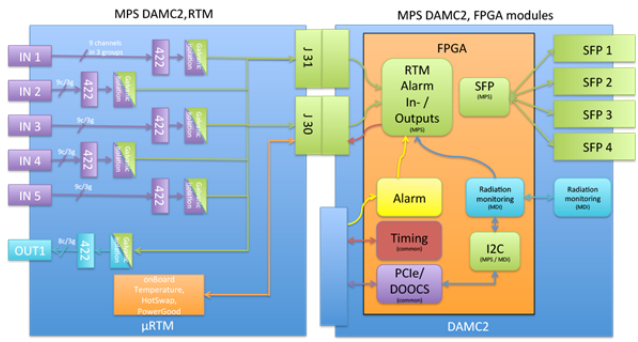


Figure 5: DAMC2 and RTM structure.

DAMC2 The DAMC2 board of Figure 5 has been developed by DESY and is equipped with a VIRTEX 5 FPGA. The documentation is available at DESY FEA group.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Figure 6: DAMC2 board.

MPS RTM The MPS RTM, shown in Figure 7, is the interface between RS422 in- and outputs and the DAMC2 board. It is developed at DESY and fulfills all requirements to be used with the DAMC2. The latency of an input signal from RTM to the FPGA of the DAMC2 has been measured with 42 ns per channel [4]. See also Figure 8 for more latency info. This has been measured in August 2013 and will be approved.

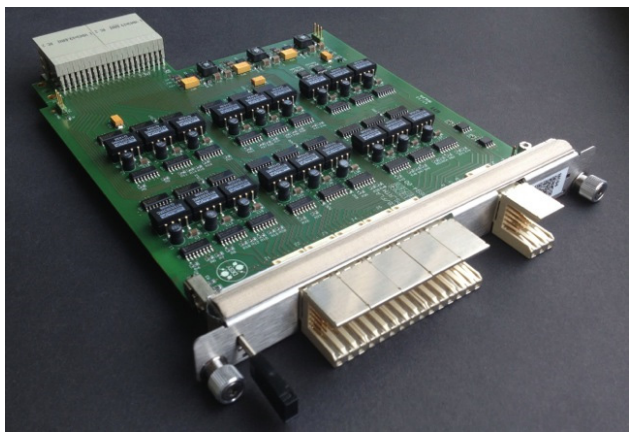


Figure 7: MPS RTM module.

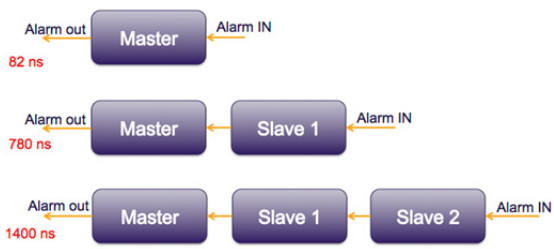


Figure 8: Latency measurement of several combinations of MPS modules.

CONTROL SYSTEM

The XFEL MPS is embedded inside the DOOCS control system [5]. DOOCS provides all necessary instances like GUIs, data logging and archiving, server functionality and other overlapping functionalities. DOOCS is used as interface for MPS configuration. It is not integrated into the safety-relevant process of the MPS. The MPS is running 100% self-reliance, if the connection to DOOCS is broken.

FINANCIAL ASPECTS

The importance to realize and design large facilities within a clear financial budget is evident. The amount of manpower and maintenance costs has to be taken into account from the beginning of every new project. For MPS it was calculated with 3 persons for the development phase and 2 persons for installation and running period. To achieve this goal, the MPS group has chosen existing hard- and software from the existing infrastructure. The only development of strictly necessary hardware inside this project is the µTCA RTM. The firmware in all MPS DAMC2 modules is the same, the additional maintenance is reduced to a minimum.

As basic idea of the whole concept, the International Linear Collider (ILC) is the force behind the design with an order of magnitude more units.

GLOSSARY

Macro-pulse
: A shot in the linac of a train of electron bunches with duration of up to ~600 µs plus the filling and decay time of the cavities. Usually repeated every 10 Hz

Beam Mode
: It defines the maximum number of bunches permitted per macro-pulse in a section of the accelerator. The MPS grants the mode by reading the machine settings and interlocks.

Section Pattern
: Every bit in the pattern describes the status of a subsection. If the bit is set to '1', beam is permitted within this subsection

REFERENCES

[1] D. K. Rybka, et al., Irradiation investigations for TESLA and X-FEL experiments at DESY. Report TESLA 2004-12, DESY, 2004.
[2] A. Aghababyan, C. Bohm, P. Gessler, A. Hidvégi, H. Kay, G. Petrosyan, L. Petrosyan, V. Petrosyan, K. Rehlich, XFEL Timing System Specifications Version 1.0, 3. 12. 2012
[3] Kay Rehlich, ARD Workshop, 23.08.2012
[4] S. Karstensen, Function and Latency test at MPS-RTM, September 2012
[5] http://tesla.desy.de/doocs/doocs.html

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

UPGRADE OF THE BEAM MONITOR SYSTEM FOR HADRON EXPERIMENTAL FACILITY AT J-PARC

Y. Morino*, K. Agari, Y. Sato, A. Toyoda,
KEK, High Energy Accelerator Research Organization, Ibaraki 305-0801, Japan

Abstract

Hadron experimental facility at Japan Proton Accelerator Research Complex (J-PARC) has been improved toward safety operation of high intensity beams. The monitor system of the slow-extraction beam line also has been upgraded to detect abnormal beam extraction. The rate monitor which measured the number of second particles from the production target with the rate of 1 kHz was prepared to detect short pulse extraction. The beam profile monitor was upgraded to read out the data at several times during a beam spill to detect a beam with the narrow width even for a short time. The beam loss monitor was upgraded to monitor beam loss with the rate of ~10 kHz to detect unexpected large loss immediately.

INTRODUCTION

Hadron experimental facility at Japan Proton Accelerator Research Complex (J-PARC) is designed to provide high intensity beams for particle and nuclear physics [1]. A 30-GeV proton beam show-extracted from main ring is injected to a production target (T1) at the hadron experimental facility. The acceleration cycle for the slow-extraction is 6 sec and the spill length is 1.5~2.0 sec.

Figure 1 shows an overview of the hadron experimental facility. In the beam switching yard, there are 33 beam line magnets in 250 m tunnel. In the HD-hall, the proton beams are injected to the production target (T1) and produce secondary particles. The proton beams are transported to the beam dump and absorbed safely. Several kinds of beam monitors are installed in the slow-extraction beam line. The beam monitors involving this paper are shown at Fig. 1.

On May 2013, a proton beam was instantaneously extracted to hadron facility in 5 msec. The short pulse beam molt the production target. After the accident, the beam operation was stopped at the hadron experimental facility. For the recovery of the hadron experimental facility, we upgraded the slow-extraction beam line in many aspects to sustain the abnormal beam extraction. The beam monitor system of the slow-extraction beam line was also upgraded to detect the abnormal beam extraction. The monitor system was developed based on EPICS [2]. The interlock system of the beam line was also improved based on the upgraded monitor system to protect the machines of the accelerator and the beam line.

The rate monitor was upgraded to measure the number of second particles from the production target with the rate of 1 kHz for the detection of short pulse extraction. The readout system of the beam profile monitor was improved to detect a beam with the narrow width even for a short time. The main purpose of the rate monitor and the beam profile monitor is to detect the abnormal beam extraction which may damage the production target. The readout system of the beam loss monitor was improved to detect unexpected large loss immediately. The operation of the slow-extraction beam line was restarted on Apr 2015 successfully.

RATE MONITOR

The rate monitor measures the number of second particles from the production target by using plastic scintillation counters. The rate monitor is installed outside the shielding enclosure of the beam line. The rate monitor was upgraded to read out the particle count with the rate of 1 kHz. Since the number of the second particles is sensitive to the beam intensity, the rate monitor is used to monitor the time structure of the beam intensity. The signals from the scintillators are converted to short pulses with the width of ~60 nsec by a NIM discriminator. The number of the second particles is measured by counting the pulses with a VME module with a FPGA board (GNV-251). The EPICS Input/Output Controller (IOC) runs on Linux on a VME CPU modules (VME-7807).

Figure 2 shows the online monitor display of the rate monitor. The number of the particles per 1 msec is plotted as a function of time. The upper panel shows the example plot in the case of the normal beam extraction. The lower panel shows the predicted plot in the case of the short pulse extraction. In the case of the short pulse extraction, the measured count will increase instantaneously, while the integrated number of the measured count will decrease due to the dead time of the read out system.

The important roles of the rate monitor are the detection of the short pulse extraction and the generation of signals for the interlock system. The upper threshold is set for the number of the particles per 1 msec (instantaneous count). In addition, the upper and lower thresholds are set for the number of the particles per 3 sec (integrated count). In the GNV-251 module, the integrated count is compared with the thresholds for every spill and the instantaneous count is compared with the threshold for every 1 msec. When the particle count crosses the thresholds, the GNV-251 module generates the interlock signal immediately and then the interlock system stops the next beam extraction [3, 4]. The control system of the rate monitor was also developed based on EPICS. The timing of the read out and the threshold values are controlled via EPICS records.

* ymorino@post.kek.jp

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

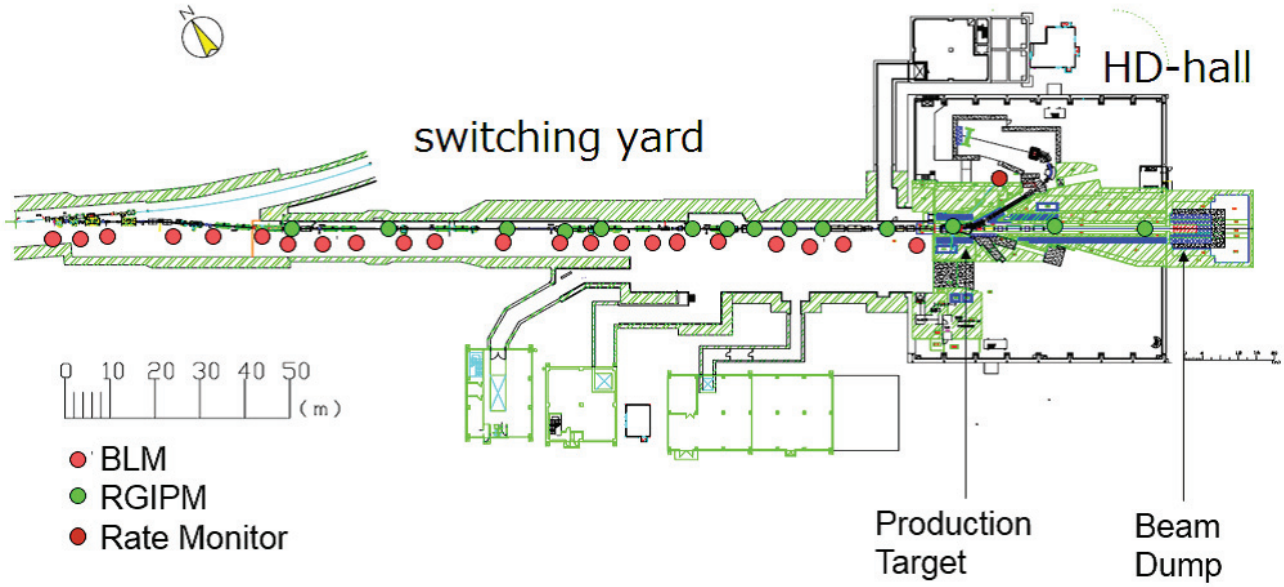


Figure 1: An overview of the Hadron Experimental Facility. Beam monitors involving this paper are shown.

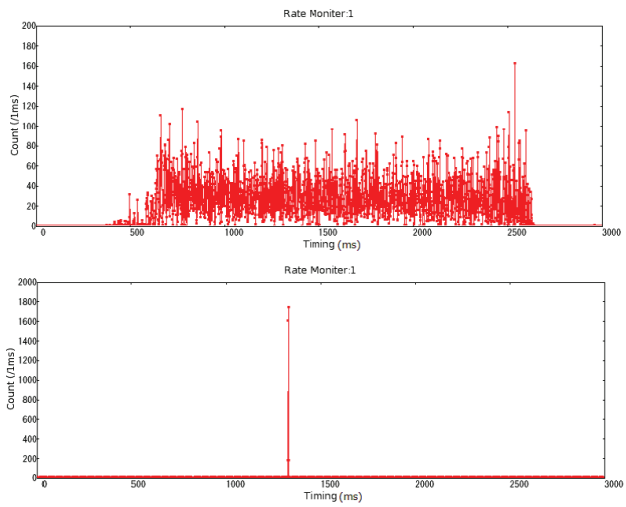


Figure 2: The online monitor display of the rate monitor. The number of the particles per 1 msec is plotted as a function of time. The upper panel shows the example plot in the case of the normal beam extraction. The lower panel shows the predicted plot in the case of the short pulse extraction.

Figure 3 shows the summary results of the rate monitor at the 33kW beam power operation. Left panel shows the result of the maximum value of the instantaneous count per a beam spill. Right panel shows the result of the integrated count. The upper and lower thresholds are also shown at Fig. 3. The integrated count was rather stable. However, the average of the integrated count was different before and after the machine maintenance. It made a double peak structure at Fig. 3. The fluctuation of each peak of the integrated count is less than 1%. The upper threshold value for the instantaneous count is 375. The upper and lower threshold values are 64000 and 37500, respectively. The performance of the interlock system related with the rate monitor was also confirmed.

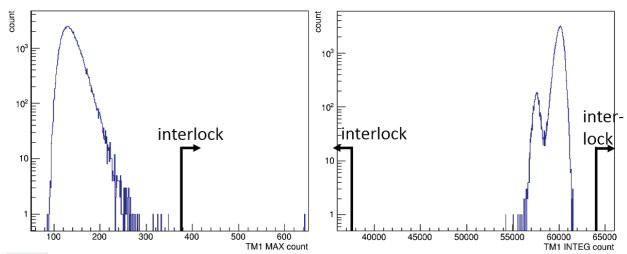


Figure 3: The results of the rate monitor at the 33kW beam power operation. Left panel shows the result of the maximum value of the instantaneous count per a beam spill. Right panel shows the result of the integrated count. The upper and lower thresholds are also shown.

BEAM PROFILE MONITOR

The beam profile in the slow-extraction beam line is measured with Residual Gas Ionization Profile Monitor (RGIPM) [5, 6]. 14 RGIPMs are located on the slow-extraction beam line as shown at Fig. 1. The signals of the RGIPMs are integrated up to about 2 sec by using a VME slow integrator (GNV-370N). The integrated signals are recorded with a VME A/D board (Advme-2607) after every beam extraction. The EPICS IOC runs on Linux on a VME CPU modules (VME-7807).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The RGIPM located at the front of the production target (T1 RGIPM) was upgraded to read out eight times per a beam spill. The read out timing of the T1 RGIPM is controlled by the GNV-251 module via EPICS records. Figure 4 shows the online monitor display of the T1 RGIPM in the case of the normal beam extraction. The upper left panel shows the pedestal data measured in the interval of every beam extraction. The other panels show the measured beam profiles after the subtraction of the pedestal. The red lines and the blue lines represent beam profiles for horizontal and vertical direction, respectively. The number at each panel represents the order of the read out. The position and width of the beam profile do not change dramatically depending on the order of the read out in the case of the normal beam extraction.

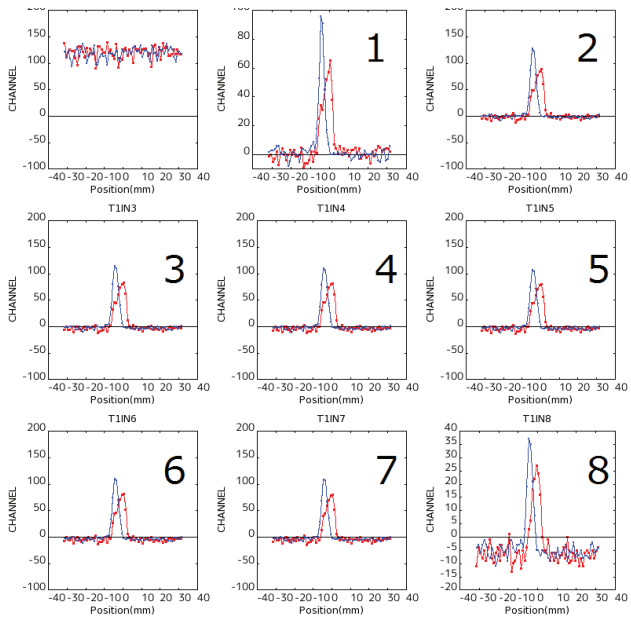


Figure 4: The online monitor display of the T1 RGIPM. The upper left panel shows the pedestal data measured in the interval of every beam extraction. The other panels show the measured beam profile after the subtraction of the pedestal. The red lines and the blue line represent beam profiles for horizontal and vertical direction, respectively. The number at each panel represents the order of the read out.

The improvement of the read out system of the T1 RGIPM aims at the detection of the abnormal beam extraction a with narrow width even for a short time. The lower threshold is set for the beam width at the T1 RGIPM in every time window. The threshold values are also controlled via EPICS records. When the beam width becomes below the threshold in any time window, the signals for the interlock system are generated to stop the next beam extraction.

Figure 5 shows the summary results of the beam widths at time window 4 at the 33kW beam power operation. Left panel shows the result of the beam width for horizontal direction and right panel shows the result of the beam width for vertical direction, respectively. The lower thresholds are also shown at Fig. 5. The typical beam widths at the T1 RGIPM were 2.6 mm and 1.8 mm for horizontal and vertical direction, respectively. The beam widths were rater stable. The fluctuation of the beam width was within 0.2 mm. The lower thresholds were ~1.9 mm and ~1.2 mm for horizontal and vertical direction, respectively. The performance of the interlock system related with the beam profile monitor was also confirmed.

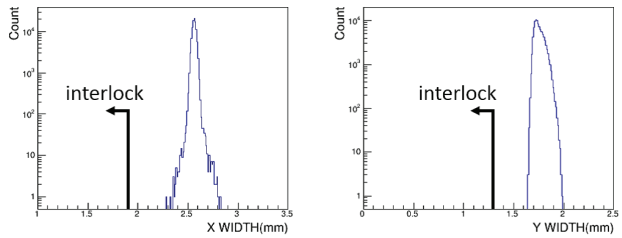


Figure 5: The summary results of the beam widths at time window 4 at the 33kW beam power operation. Left panel shows the result of the beam width for horizontal direction and right panel shows the result of the beam width for vertical direction, respectively. The lower thresholds are also shown.

BEAM LOSS MONITOR

Beam loss at the slow-extraction beam line is measured by using air ionization chamber (BLM). 22 pairs of BLMs are located beside the beam line as shown at Fig. 1 to monitor unexpected large beam loss.

The signals of BLMs are integrated up to about 2 sec by the GNV-370N module in the same way as the RGIPMs. FA-M3 Programmable Logic Controller (PLC) was introduced to read out the integrated signals of the BLMs to detect the accidental large loss immediately. Figure 6 shows a picture of the PLCs of the BLM read out system. A PLC sequencer CPU (F3SP71-4S), PLC A/D modules (F3HA12-1R) and a PLC output module (F3YC08N) are used to read out the integrated signals and to generate signals for the interlock system. The EPICS IOC runs on Linux on a PLC CPU module (F3RP61-2L) [7].

The PLC A/D modules digitize the integrated signals with the rate of 200 kHz. The ADC values of the integrated signals are compared with the thresholds every time the PLC sequencer CPU scans. The scanning rate of the PLC is ~10 kHz. When the ADC values excess the thresholds, the F3YC08N module generates the interlock signals immediately. The ADC values are recorded as EPICS records for every 200 msec during the beam extraction for online-monitoring. Figure 7 shows the online monitor display of the BLM. Each panel shows the measured beam loss at each time window after the subtraction of the pedestal. The number at each panel represents the order of the read out. The measured beam loss is plotted as a function of the located position of the corresponding BLM. Blue bar represents one

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

of the pair BLM and red bar represents the other of the pair, respectively. The upper thresholds are set for the measured beam loss. The interlock system of the BLMs also worked well.

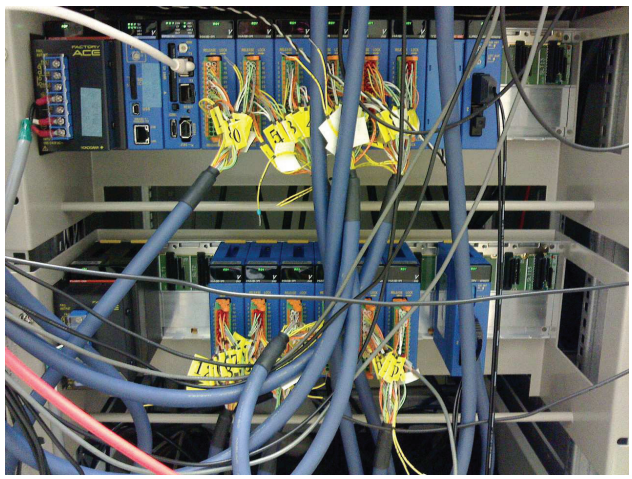


Figure 6: A picture of the PLCs of the BLM read out system.

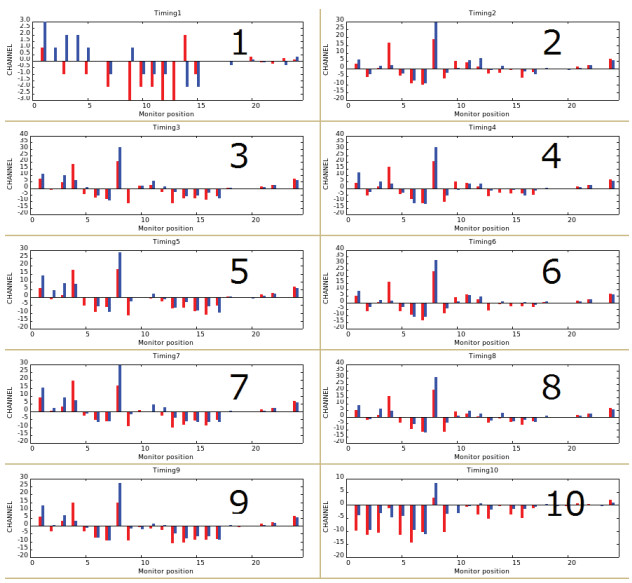


Figure 7: The online monitor display of the BLMs. The number at each panel represents the order of the read out. Blue bar represents one of the pair BLM and red bar represents the other of the pair, respectively. The measured beam loss are plotted as a function of the located position of the corresponding BLM.

SUMMARY AND OUTLOOK

The rate monitor, the RGIPMs and the BLMs have been improved to detect the abnormal beam extraction and investigate the status of equipment of the beam line. These monitors also generate the signals for the interlock systems to protect machines at the beam line when the abnormal beam injection is detected. The slow-extraction beam line was recovered on Apr 2015. The operation of the beam line was successfully done for ~2 months with the upgrade monitor and interlock system.

A new interlock system (Sx Abort) will be introduced from Oct 2015 so that the abnormal beam could be aborted even during the beam extraction. The Sx Abort system will be powerful method to protect the machines at the beam line. The 33 kW beam power has been achieved on June 2015, which corresponds ~1.5 times higher than one on May 2013. The beam power is planed to be increased further gradually. The beam monitor system and the interlock system of the beam line have been improved toward safety operation of the higher power beam.

REFERENCES

[1] J-PARC website: `http://j-parc.jp`

[2] EPICS website: `http://aps.anal.gov/epics/`

[3] N. Yamamoto et al., "Current status of the control system for J-PARC accelerator complex", Proceedings of ICAPEPCS 2007, Knoxville, Tennessee, USA, pp. 62-64, 2007.

[4] Y. Sato et al., "Control system for J-PARC hadron experimental facility", Proceedings of ICAPEPCS 2009, Kobe,Japan, pp. 331-333, 2009.

[5] T. Kwakubo et al., "Fast data acquisition system of a nondestructive profile monitor for a synchrotron beam by using microchannel plate with multianodes", Nucl. Instr. Meth. A302, pp397-495 (1991).

[6] Y. Sato et al., "Residual gas ionization profile monitors in J-PARC slow-extraction beam line", Proceedings of IBIC 2012, Tsukuba, Japan, pp. 267-270, 2009.

[7] Y. Odagiri et al., "Application of EPICS on F3RP61 to accelerator control", Proceedings of ICAPEPCS 2009, Kobe,Japan, pp. 916-918, 2009.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

NONLINEAR SYSTEM IDENTIFICATION OF SUPERCONDUCTING MAGNETS OF RHIC AT BNL*

P. Chitnis#, Stony Brook University, NY, USA
K. A. Brown, Brookhaven National Laboratory, NY, USA

Abstract

The Quench Detection System (QDS) of RHIC detects the Superconducting (SC) magnet quenches by voltage sensing. The real-time voltage across the SC magnet is compared with a predicted voltage from a behavioral model, a deviation from which triggers the quench event and energy extraction. Due to the limitations of the magnet model, many false quench events are generated that affect the RHIC availability. This work is targeted towards remodeling the magnets through nonlinear system identification for the improvement in QDS reliability. The nonlinear electrical behavior of the SC magnets is investigated by statistical data analysis of magnet current and voltage signals. Many data cleaning techniques are employed to reduce the noise in the observed data. Piecewise regression has been used to examine the saturation effects in magnet inductance. The goodness-of-fit of the model is assessed by field testing and comprehensive residual analysis. Finally a new model is suggested for the magnets to be implemented for more accurate results.

INTRODUCTION

The RHIC SuperConducting (SC) magnets store an energy of 70MJ in the form of magnet currents during a full energy run. The SC magnets are susceptible to quenches that lead to the development of tiny resistive zones. An operating current near 5000A (for a dipole magnet) can dissipate this enormous energy at this tiny resistive point causing catastrophic damage.

To safeguard against such failure, Quench Detection System (QDS) is employed. It monitors the SC magnets to detect the developing quenches and sends the magnet power dump signal and beam abort signal to the beam permit system [1]. Voltage sensing is employed for recognizing the developing quenches. The QDS consists of DSPs which store the electrical behavioral model of the SC magnets. The actual magnet output is compared to the model output, and a deviation is sensed as a developing quench, which generates a quench trigger.

The SC magnets exhibit a highly nonlinear behavior due to saturation and hysteresis of steel yoke [2]. Due to mathematically intractable nature of this behavior, the magnet model parameters are manually calibrated, which inhibit the accurate tuning of the model. Also it consumes valuable time when RHIC is running at 4K temperature. Inaccuracies introduce deviation in the model output, which leads to false failures, and resulting in unnecessary machine downtime. Thus to improve QDS reliability, it is necessary that the model truly imitates the SC magnet behavior. The aim of this work is to facilitate automatic generation of accurate magnet models through nonlinear system identification that will improve the reliability and availability of QDS.

Original Magnet Model

The SC magnet circuit's electrical behavior is modeled as a pure inductor with a series resistor. The pure inductor represents the SC magnet and the resistance represents the current leads to the magnet. The model is

$$V_c = L\frac{dI}{dt} + RI$$

Here I is the current through the magnet, L is the nonlinear magnet inductance, R is the lead resistance and V_c is the calculated voltage from the model. This voltage is compared to the observed voltage V_o across the magnet in real time. When a quench develops, additional resistance will appear causing V_o to deviate from V_c. A difference more than 25 mV is triggered as a quench event.

The parameter L is highly nonlinear in nature. It exhibits saturation i.e. its value decreases with increasing current. Also the *L vs. I* curve changes with the change in current ramp waveform. The model stores lookup table for *L vs. I* values, which have to be updated manually every time a new current waveform is introduced. An *L vs. I* lookup table is shown in Fig. 1. This gives us a rough idea of the nonlinear behavior of *L*.

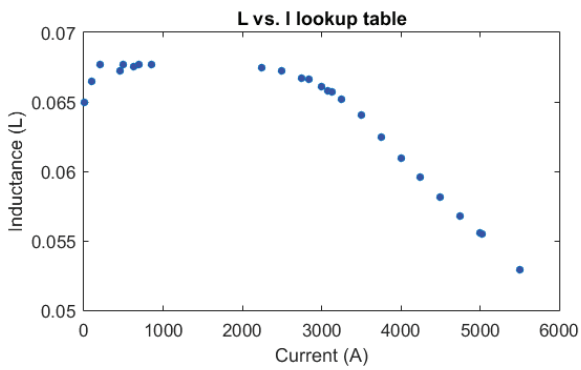


Figure 1: L vs. I lookup table values

*Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy.
#prachi.chitnis@stonybrook.edu

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SYSTEM IDENTIFICATION

Our aim is to find an accurate inductance variation with the current change to construct accurate *L vs. I* tables to be used in the DSP model. We analyze the magnet current and voltage data for a magnet tap named B1DSD9_5VT. For this particular voltage tap connected across dipoles, the *R* value is zero, so we are left with the following model

$$V_c = L\frac{dI}{dt} \quad (1)$$

One of the techniques that can be applied to solve for *L* is the linear regression, where the explanatory variable is the first derivative of the current and the response variable is voltage. We mine the voltage and current data from RHIC database using MATLAB® [3], which is found to be quite noisy. Particularly for the derivative of current, noise is highly amplified. Thus we attempt to clean the data first. We analyze the frequency spectra of the current and voltage data, and their variation over time through spectrograms shown in Fig. 2.

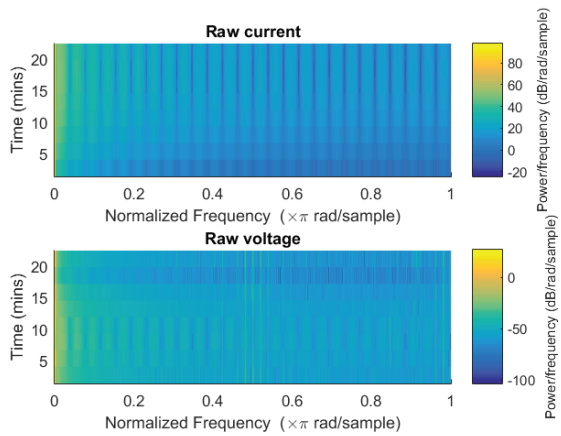


Figure 2: Spectrograms of voltage and current signals

Next we choose to filter the voltage and current data per the frequency characteristics in Fig. 2. The noise in the first derivative of current is now eliminated. Figure 3 shows the filtered (green) and raw (black) signals for the magnet current, first derivative of the current and voltage signals. The periodic noise and spikes in the data are now removed.

Now the data is ready for analysis after the preliminary processing. Consider the *dI/dt* waveform in Fig. 3. There are certain portions of the current where the first derivative of the current is either zero or has very low value. For a good fit of the linear regression model, the explanatory variable should have a substantial magnitude. Thus to apply the regression algorithm using Eq. 1, we eliminate small or zero values of *dI/dt* and corresponding values of *V* from the data. We have written a data classification algorithm to segment and identify the regions in current waveform where the first derivative is non-zero, second derivative is non-zero etc. The segmented portions of the magnet current are shown in Fig. 4.

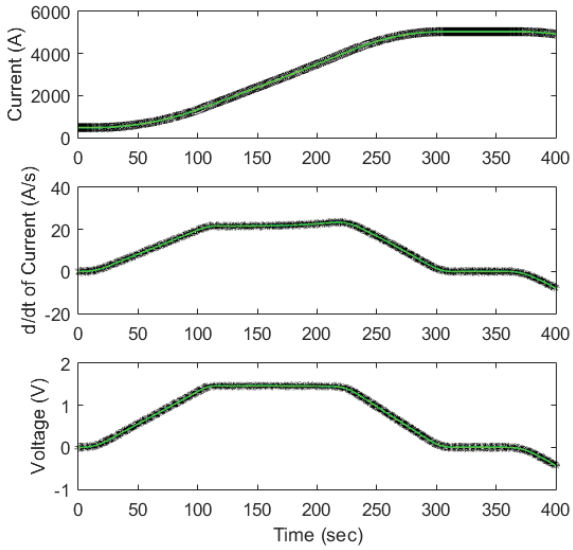


Figure 3: Raw and filtered I, dI/dt and V

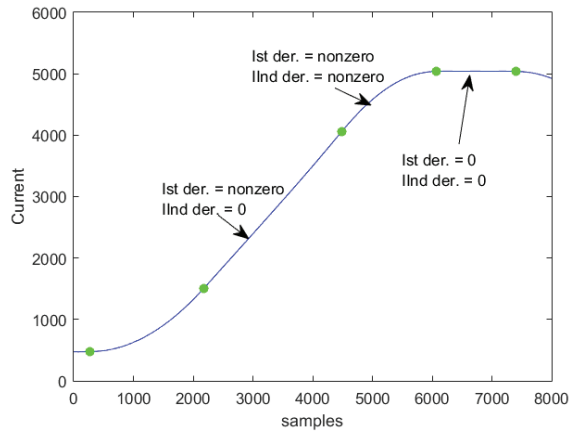


Figure 4: Current segmentation

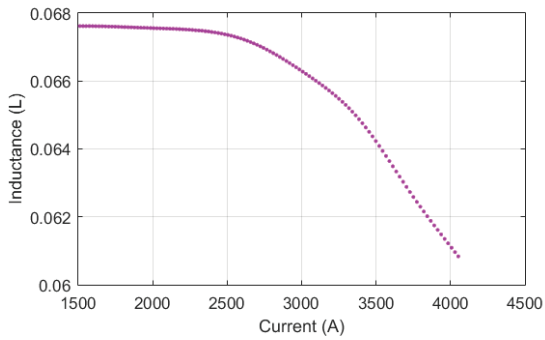


Figure 5: L vs. I curve from piecewise regression

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

We use Eq. 1 for regression in region where the first derivative exists. Now we address the nonlinear variation of L with increasing current. One method to deal with this is to use piecewise regression, where the current is divided into very small data sets, and regression is applied to each set for finding L value. The L is assumed constant for this small dataset. Now we plot these values of L with I in Fig. 5 that shows the saturation characteristics of L similar to Fig. 1, but much cleaner.

FIELD TESTING

Before testing the model in the field, we look at the residuals that we obtain from our analysis. The residuals between the filtered voltage and the calculated voltage are shown in Fig. 6. As seen the maximum value of the residuals is about 8mV, which is well below the 25mV limit.

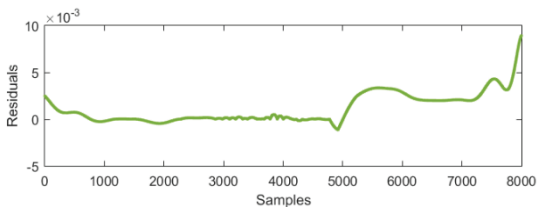


Figure 6: Residuals between the filtered and fitted voltage

The piecewise regression generates a smooth curve of about 200 data points for L *vs.* I curve (Fig. 5). However the DSP model can only store 35 values of L vs. I data. It interpolates the L values for in-between I values. We generated L *vs.* I tables for DSP using this data, and a screen shot from the field testing on the same voltage tap is shown in Fig. 7. The difference signal (in blue) is the moving average of 100 values of the actual voltage difference (in grey), which is used for triggering quench event in case it goes beyond 25mV. The maximum value of the trigger signal is found to be 6mV which is quite below the 25mV level.

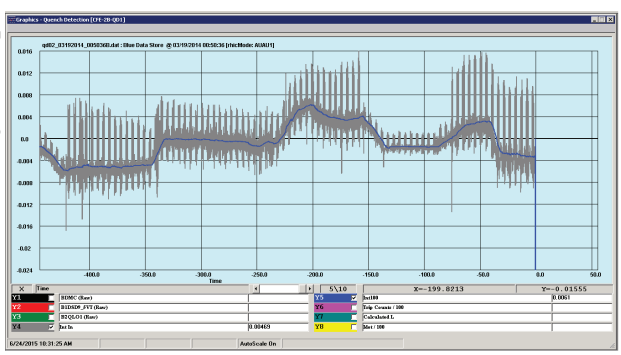


Figure 7: Field testing, quench trigger signal

Residual Diagnostics

The linear regression model validity can be established by residual diagnostics. We use the guidelines discussed in [4] for checking our model.

The true residuals of our model are obtained by the difference between the observed magnet voltage and the predicted voltage from the newly generated DSP table, as this table will actually be used in the field. Similar to the DSP, we find the predicted voltage from Eq. 1, by interpolating the L values for intermediate I values in the table.

We use the following rules for our residual diagnostics as discussed in [4]. It is to be noted that the shape of residuals in Fig. 6 will be visible in these residuals as well. We address this residual pattern later in this paper. First, the explanatory variable should be linearly related to the response variable. This can be observed by plotting the residuals against the explanatory variable. We already linearized our model by piecewise regression so that the residuals will be linear with respect to dI/dt. As seen in Fig 8, the trend is almost linear, with little variability.

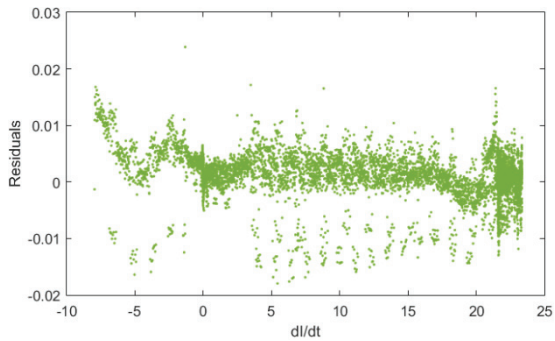


Figure 8: Residuals vs. explanatory variable

Secondly, the residuals should have nearly normal distribution. This can be seen by plotting the histogram and quantile-quantile plot (Fig. 9). From the figure it is seen that the residuals are a little light-tailed but are not skewed, and are nearly normal.

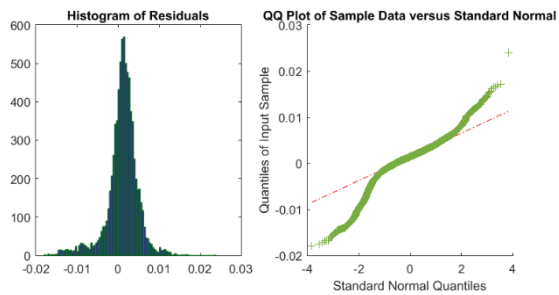


Figure 9: Histogram and QQ plot

Thirdly, the residuals should have constant variability. This can be checked by plotting the residuals against the response variable where the residuals should be randomly

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

scattered around the zero value. This rule is satisfied as seen in the Fig. 10.

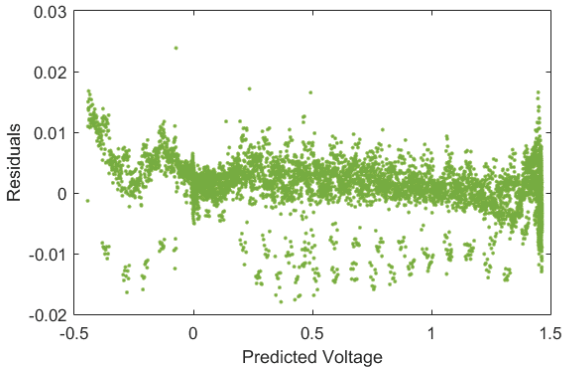


Figure 10: Residuals vs. response variable

Lastly the residual values should be independent of each other. This can be checked by the scatter plot of residuals (Fig. 11). In all the residual plots we see a periodic structure, which is due to the Booster main magnet pulses disturbing the power line [5, 6]. Other than this, the pattern looks random.

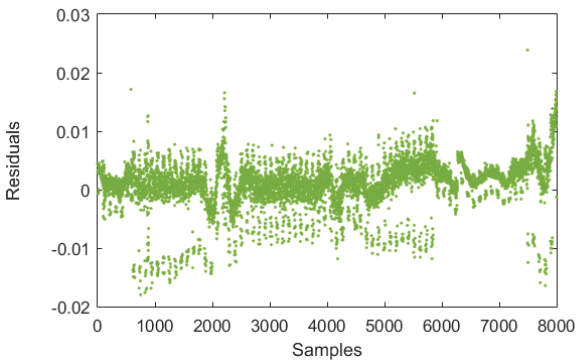


Figure 11: Scatter plot of residuals

DISCUSSION

To further improve the model, the remaining variability in the residuals has to be analyzed. As seen in Fig. 7, we see a pattern in the residuals. This variation depicts a dependence on the second derivative of current, which is shown in Fig 12.

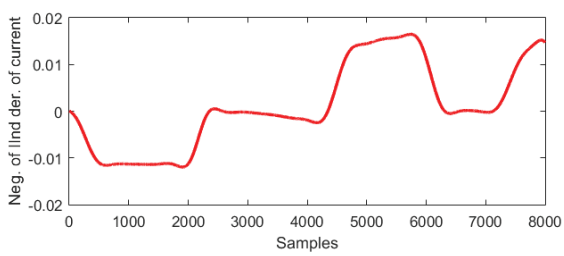


Figure 12: Negative of IInd derivative of current

To accommodate this variance, we can modify the magnet model as the following equation.

$$V_c = L\frac{dI}{dt} + X\frac{d^2I}{dt^2}$$

Here X is a coefficient of second derivative of current, which can be a constant throughout the data, and might not need piecewise regression. This X parameter can then be modeled as an eddy current component and/or parasitic capacitance in the circuit. More accurate analysis can be done by performing the frequency response analysis.

CONCLUSION

We utilize the statistical analysis concepts to reveal the underlying saturation characteristics of the magnet, and validate our model using field testing and residual analysis. This will facilitate the automatic generation of the DSP tables without any manual intervention. This helps to save the valuable resources when the RHIC is running at 4K temperature. Going further, we have also developed an analytical memory model that quantifies the saturation and hysteresis behavior of RHIC superconducting magnets [7]. This will help forecast the *L vs. I* curve for particular current waveform, accounting for nonlinearities due to saturation and hysteresis.

ACKNOWLEDGEMENT

We would like to thank D. Bruno for his constant support, help with testing the inductance tables and providing the magnet data. We also thank G. Ganetis for his guidance in explaining RHIC magnet characteristics.

REFERENCES

[1] C.R. Conkling, “RHIC Beam Permit and Quench Detection Communication System”, Proceedings of the Particle Accelerator Conference, 1997

[2] *Relativistic Heavy Ion Collider configuration manual*, Accelerator Division, Collider Accelerator Department, Brookhaven National Laboratory, NY, 2006

[3] MATLAB® Release 2015a, The MathWorks Inc., Natick, MA, 2015

[4] M Çetinkaya-Rundel, *Data Analysis and Statistical Inference*, Coursera MOOC videos, Duke University, 2014

[5] E. Bajon et al., “Booster main magnet power supply, present operation and potential future upgrades”, PAC 2011, Albuquerque, NM, USA

[6] P. Cameron et al., “Progress in tune, coupling, and chromaticity measurement and feedback during RHIC run 7”, PAC 2007, Albuquerque, NM, USA

[7] P. Chitnis, K.A. Brown, “Analytical modeling of inductance for saturation and hysteresis of superconducting magnets”, *submitted for journal publication*

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ALMA RELEASE MANAGEMENT: A PRACTICAL APPROACH

R. Soto, T. Shen, N. Saez, Joint ALMA Observatory, Santiago, Chile
J. Ibsen, European Southern Observatory, Santiago, Chile

Abstract

The ALMA software is a large collection of modules, which implements the functionality needed for the observatory day-to-day operations. The main ALMA software components include: array/antenna control/correlator, submission/processing of science proposals, telescope calibration and data archiving. The implementation of new features and improvements for every software subsystem must be coordinated by considering developers schedule, observatory milestones and testing resources available to verify new software. This paper describes the software delivery process adopted by ALMA since the construction phase and its evolution until these days. It also presents the acceptance procedure implemented by the observatory for validating the software used for science operations. Main roles of the software delivery and acceptance processes are mentioned on this paper by including their responsibility at the different development and testing phases. Finally, some ideas are presented about how the model should change in the near future by considering the operational reality of ALMA Observatory.

OVERVIEW

The software delivery process of ALMA Observatory has passed for several transformations aligned to observatory's project lifecycle. Thus, it changed from a static model (with few and big releases) to a dynamic schema with emphasis at the testing phases and reducing the integration time required for a new release. This paper will present the evolution of the ALMA Software Delivery Process and Release Management by describing the advantages/disadvantages of the models adopted. Final section will give an overview how the model will change in the near future.

SOFTWARE RELEASES IN THE PAST

The ALMA Observatory started its commissioning phase at the Chilean site during the end of 2009. It considered some antennas installed at the high site (5000m) and the deployment of the first quadrant for the baseline correlator. Additionally, the assembly, integration and verification activities (AIV) related to the new array elements delivered by manufacture vendors [1], continued more intensely at the Operation Support Facilities (3000m). This period was very intensive for the computing group, since simultaneous activities had to be supported in parallel. Commissioning process by using direct observing systems (control and correlator software, front-end archive, etc.) was required at the Observatory and, at the other hand, the preparation, integration and testing of pre and post observing software (Phase 1 and Phase 2 proposal submission, time allocation committee support, data processing software, etc.) was demanded as an Early Science task, planned to be started at the end of 2010. In terms of software releases, a cycle of 6 months for the delivery of new features was established, which included capabilities for the observing systems and the proposal handling process as well. These cycles considered several phases (with a timeline predefined) before declare the software as accepted for AIV activities or Early Science observations as described at [2]: *Code Freeze Period, Initial Integrated Testing, Initial Site Testing, Computing Release, Routine Use, Acceptance Testing and Integrated Testing*. The model worked relatively well during early construction stages when the activities were concentrated at the ALMA Test Facilities [3]. Software was commissioned by using prototypes antennas and there was time available with the operational hardware for testing purposes. However, this approach was deficient when commissioning and AIV activities started at the operational site. There were less access to the hardware for testing and more pressure for having new software capabilities working in order to continue progressing into array commissioning.

Preliminary testing was initially executed in a simulated environment, which was not able to reproduce the exact behavior of the operational hardware. So, even a full battery of tests were executed, which considered new functionality and regression tests, there were not enough to deliver a mature software for the site testing. Many bugs were detected using the operational hardware, which increases the cost of correct them [4]. On the other hand, the big amount of features delivered per release also produced additional problems to distill the software. Thus, the stabilization of a new release took about two months after the integration and testing team delivered it. After that, science group should proceed with the routine use and acceptance testing, but the long integration period introduced big delays in the whole commissioning process. Given the current scenery, the integrated computing team looks for changes in the software delivery process adapting the model to the current reality.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE CURRENT MODEL

Incremental Release Process

The model currently adopted differs from the previous one in terms of periodicity of the incremental releases delivered for science commissioning. We moved from 6-month period to bi-monthly schedule, which consider testing and integration as part of the cycle. Thus, a more frequent software delivery also implied that less features and improvements were included per release facilitating the integration, testing and debugging of the detected problems. Of course, these features and improvements are scheduled according to the observatory's milestone such as software needed for proposal submission, projects rating and starting of the observing cycle. This model also included the definition of different phases with formal handover between each one. Three phases were established:

1. **Phase A - Developer Integration & Testing**: Aimed to demonstrate that functionality is implemented as requested but based in unit tests. All tests must pass to move toward verification phase.
2. **Phase B - Verification**: Aimed to tests new functionality using both (largely) simulation and (when required) production environments. This include regression tests suite [5] for detecting bug and provide fixes to them. All tests must pass to proceed with the validation phase; otherwise features are "de-scoped".
3. **Phase C - Validation:** Aimed to validate additional capabilities and scientific data content. Performance and robustness aspects are also analyzed as part of validation tests (new releases should always behave better or at least no worse than the previous ones). Software acceptance requires both correct verification and validation test results.

The introduction of these formal phases also produced an optimization of the development and testing resources at the computing and science areas. A calendar with dates for every phase was prepared and circulated to developers, computing and science testers [6]. Also independent phases were parallelized (as showed in figure 1) which optimize the available resources. Release contents are fully tracked by using Atlassian tool called JIRA [7]. Every feature/improvement is registered in separate tickets, which also contain the testing results of every phase.

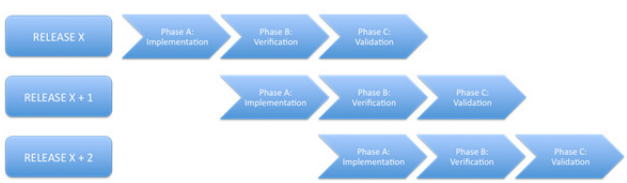


Figure 1: Incremental releases lifecycle.

Formal responsibilities were defined at the computing and science teams related to the planning and delivery of the software. Thus, the *release* and *acceptance* manager roles were introduced. The first one is responsible for the planning and delivery of the incremental releases and the completion of the verification phase. Acceptance manager represents the clients or users point of view. He/she is responsible for reviewing the planning of the releases and make sure they are according to the Observatory's milestones and science needs. Acceptance manager has also the responsibility for the execution of the validation phase and prepare the acceptance plan for the software which has successfully passed verification and validation phases.

Acceptance Process

Once several incremental releases have been successfully verified and validated, they must be accepted in order to be used for official science activities (no commissioning ones) such as: Early Science observations, hardware commissioning, etc. The acceptance manager carries out the acceptance process and it is usually held very close to an observatory milestone. It consider the following steps:

a) **Test Report Review (TRR)**: This meeting has a goal the revision of the verification/validation reports for all incremental releases included at the acceptances. Important issues are also identified and the schedule to solve them is defined. It can be more than a TRR before the acceptance if there are many issues still pending to be solved. Once everything is OK from computing and science point of view, then the candidate branch is created and we move to the acceptance testing period.

b) **Acceptance Testing Period**: During this period the final tests are performed over the candidate branch. The idea is not repeat the verification or validation phases but do a light regression tests of all applications to be deployed on production servers. These tests are performed in an isolated environment, which consider a recent image of the production DB in order to do them as much realistic as possible.

c) **Acceptance Review**: The acceptance meeting considers the revision of the acceptance tests report and also the revision of the pending items. During this meeting the final decision is taken about to deploy the new software in production servers or postpone it until all the critical items had been solved.

d) **Software Deployment in Production Environment**: This step considers the coordination with the different ALMA centers around the world for the deployment of the new software. It also included the changes into the database model needed for the new version of the applications. Usually it involves some downtime at the applications affected in order to deploy new versions. All the work is coordinated by the deployment manager who has to submit a report at the end with the status of the new software.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The Acceptance Manager is also responsible for preparing the acceptance calendar, which is aligned to the observatory's milestone and consider enough time to proceed with all the steps defined above. The acceptance calendar also contains the information about the incremental releases included into the acceptance. The calendar is distributed between scientists and developers and it represents the official information for any software planning activities.

The Software Change Control Board

Once a software release has been accepted, there is a formal process for introducing changes into the accepted branch. These requests can be associated to a software patch (related to a bug fix), to a change in the functionality requirements (called release change request), to a modification of the database schema (usually requested by software applications) or to a set of minor improvements for a specific application (called a service release). All software requests must be informed to the Release Manager by submitting a JIRA ticket [7].

A *Software Change Control Board* (SCCB) is a committee compound of various project stakeholders that typically fulfills the requirements for such a process. The group is formed for representative from computing, science and engineering sides. The SCCB (or a subset of them constituted mandatorily by the Release Manager and Acceptance Manager) will meet once a week to discuss and decide on outstanding software requests. Additional meetings may be called as required. Emergency requests need to be addressed immediately by a procedure properly established.

The decisions will normally be made by consensus. The nominated Acceptance Manager signs off all decisions. If no mutual agreement can be reached, the Release and Acceptance Managers together the head of the computing section responsible for the implementation must at least agree upon any decision. Otherwise the issue needs to be escalated to ALMA Observatory Management.

EVOLUTION TOWARD AN AGILE APPROACH

The ALMA observatory has concluded its construction phase and it is moving to full operations model. The transition implies more significance at the system robustness and stability in order to implement continuous observation model and a reduction at the time dedicated for commissioning and verification. For this reason, the software delivery process should be adjusted to the current observatory state. Thus, given the hardware restrictions, simulation capabilities will have a relevant role in the verification phase. The number of new features/improvements per release will be reduced but the emphasis at the software robustness becomes essential. System stability turns into a critical point in order to maintain observatory working most of the time, therefore the downtime due to new software releases must be strictly controlled.

The New Approach

Based in the situation mentioned above, an agile approach for the software delivery process should be adopted by the observatory. The proposal is being developed and it expected to be implemented in the coming years. Basically, this approach is based in the existence of a stable branch, which is patched for verification, and validation of new capabilities. Developers should commit functionality in separate branches and verification team should patch stable branch for verification purposes. If verification passed, Science testers should validate same functionality. After successfully validation, the patch can be integrated at the stable branch and considered ready to be used for observatory's activities. This model differs of the previous one, since integration is controlled by verification team instead of developers. Stability should be also granted since less functionality is included per iteration. Features, which do not pass verification or validation phases are rejected and scheduled for another iteration. Observatory's technical times are also optimized since only features, which have passed simulation tests, are considered to be verified with operational hardware. Despite this approach has several advantages in terms of software stability and accelerate the process for having new capabilities in production environment (by eliminating or reducing software acceptance process), it also require some fundamental changes in the paradigm currently adopted. It considers more discipline at the development, verification and validation teams in order to accomplish with the tight schedule, optimizing resources available and delivering new capabilities according to the observatory needs. The figure 2 illustrates the core of the new process where the science branch (accepted) is created after the verification and validation phases have been completed successfully.

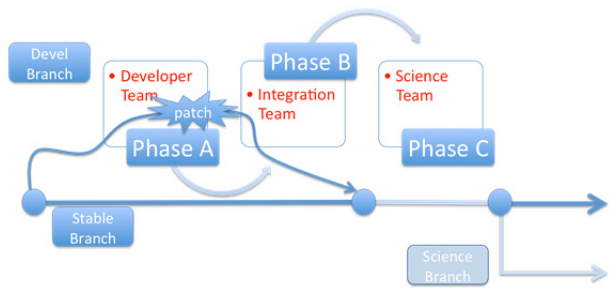


Figure 2: Proposal for agile approach.

CONCLUSIONS

This paper presented the evolution of the release management process in agreement with the life cycle of ALMA Observatory. There was a transition from a traditional and static development model, suitable for early construction phases, toward a dynamical one, which

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

considered commissioning restrictions. This new model takes into account the delivery of lite releases in terms of features but more stable as a whole. This also increased the frequency of the development cycles according to the observatory's milestones and decreased the integration/testing time required before the science commissioning phases. Formal phases were introduced as part of the process and responsible for every stage were properly identified and designated. This facilitated the process control, allowing a deterministic schedule for the entire cycle. There was also more emphasis for controlling changes over commissioned releases used for official science activities. The creation of a control board for approving/rejecting changes, evidenced the importance of maintain operational software stable as much as possible. The results showed at the end demonstrated this was the correct path since ALMA commissioning phase has been successfully performed from the software point of view.

However, there is still another important milestone to be completed by the Observatory in the coming years: the full operations model that will demand a new adaptation of the software delivery process in order to fulfill the operational requirements. Thus, an agile approach was proposed that considers the robustness and stability of the system as a mandatory goal over the introduction of new capabilities. It is expected that several improvements at the system simulation and continuous integration environment must be developed as part of the implantation of the model.

The experience reveals that the implementation of a new model is not a straightforward process. It will require several technical improvements but, more important and difficult, is the adaptation of human capital (developers, testers, validators) to new paradigm.

REFERENCES

[1] B. E. Glendenning, and G. Raffi., "The ALMA computing project: initial commissioning", Proc SPIE, Vol. 7019, 701902 (2008).

[2] B. E. Glendenning, J. Ibsen , G. Kosugi , G. Raffi, "ALMA software management and deployment", Proceedings of SPIE Vol. 7740, 77401L (2010).

[3] B. Lopez, R. Araya, N. Barriga, et al., "Software regression testing: practical experience at the ALMA test facility", Proceedings of SPIE Vol. 7019, 70192X (2008).

[4] V. Gonzalez, M. Mora, R. Araya, et al., "First year of ALMA site software deployment: where everything together", Proceedings of SPIE Vol. 7737, 77371Z (2010).

[5] R. Soto, T. Shen, J. Ibsen, et al., "ALMA software regression tests: the evolution under an operational environment", Proceedings of SPIE Vol. 6541, 84511R (2012).

[6] T.C. Shen, J.P.A. Ibsen, R.A. Olguin, R. Soto, "Status of ALMA Software", Proceedings of ICALEPCS 2011, Grenoble, France.

[7] JIRA Atlassian planning and tracking tool, https://jira.atlassian.com

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

USE INTERRUPT DRIVEN MODE TO REDESIGN AN IOC FOR DIGITAL POWER SUPPLY AT SSC-LINAC

Shi An, Wei Zhang, Kewei GU, Junqi Wu, Xiaojun Liu, IMP, Lanzhou, China

Abstract

SSC-LINAC control system is based on EPICS architecture. The sub control system of digital power supplies is a kind of IOC send and receive custom command via Ethernet and TCP/IP protocol. The old IOC is designed to use period scan mode IOC, and there are so many digital power supplies, that we can't make sure every connect condition of digital power supply is fine. IOC must wait a long time if one of them can't connect correctly and other digital power supply's PV may also be blocked. An IOC that uses interrupt driven mode to avoid the shortcoming was designed. This will be described in this paper.

INTRODUCTION

SSC is a separated-sector cyclotron. To improve the efficiency of HIRFL, a linear accelerator is considered as a new injector for SSC of HIRFL [1]. The SSC-LINAC control system is based on EPICS. There are many sub systems such as power supply control system, vacuum monitor system and so on. The structure of SSC-LINAC control system is shown in Figure 1.

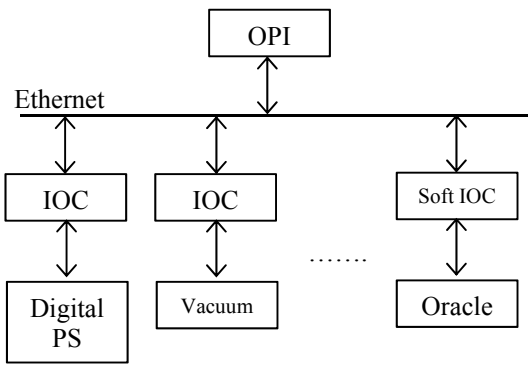


Figure 1: Structure of SSC-LINAC control system.

Almost all of the power supply in the system uses digital power supply. So control interface between IOC and power supply is Ethernet interface. IOC sends and receives predefined string command via the Ethernet with the PS (power supply).

DESCRIPTIONS

This part wills description the old PS IOC design and new design of the IOC, also description implement of the new IOC. Descriptions include two sections and each section is introduced in detail as following. The structure of the PS IOC is shown in Figure 2.

Structure of the Origin PS IOC

The bottom of the structure is digital PS. Above the PS is Ethernet interface and use TCP/IP as low level transfer protocol. In the middle layer is device support of EPICS architecture. The device support has two functions. First function is a periodic that sends a read command string to PS in every one second and receives current state of the PS. The second function is sends a write command string to PS. The top of the structure is record support.

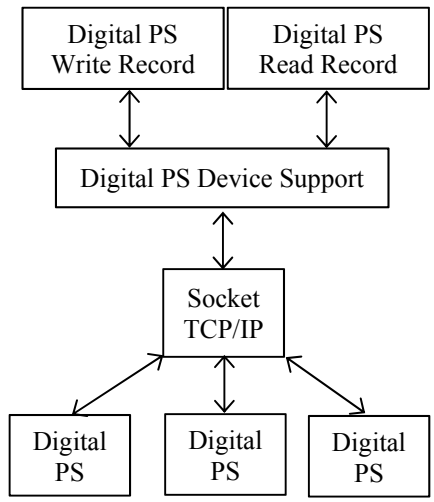


Figure 2: Structure of the PS IOC.

In the real case we create two PVs for every digital PS and there are 78 power supplies in SSC-LINAC. Actually all of PVs are run in one thread in the structure. If just one digital PS is down and all of other digital power supply's PV also be blocked (blocked time depend on the timeout setting). So we must design a new device support to avoid the bad situation.

Design a New Device Support for Digital PS

First switch the Record periodic SCAN ("1 second") to SCAN rate driven from a device specific source ("I/O Intr"). In our case separated thread for every one PS is used.

Second step is writing device support use interrupt model and implement in the following sections.

1. init

```
static long init(int phase)
{
        if(phase==0)
                initHookRegister(&start_ps_thread);
        return 0;
}
```

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Register a hook function at the device support init(int) function.

2. Init hook

```
static void start_ps_thread (initHookState state)
{
   ELLNODE *cur;

   if(state!=initHookAfterInterruptAccept)
                        return;

   for(cur=ellFirst(&allprngs); cur; cur=ellNext(cur))
   {
        struct prngState *priv = CONTAINER(cur,
struct prngState, node);
        priv->generator = epicsThreadMustCreate(
           "psThread",
           epicsThreadPriorityHigh,
           epicsThreadGetStackSize(epicsThreadStack
Small),
           &ps_thread, priv);
   }
}
```

The hook function starts a thread for each of the digital PS.

3. Digital PS thread

```
static void ps_thread(void* raw)
{
           struct prngState* priv=raw;
           char send_buf[16] = {0,};
           char buf[BUFF_SIZE] = {0,};
           char *endptr;

           //init socket
           int sockfd;
           struct sockaddr_in device_addr;
           ……
           //connect to PS
           …...

           while(1)
           {
           //send and receive something from the PS

           epicsMutexMustLock(priv->lock);
           //write useful data to record
           ……
           //queue a request
           scanIoRequest(priv->scan);
           epicsThreadSleep(interval);

           }
}
```

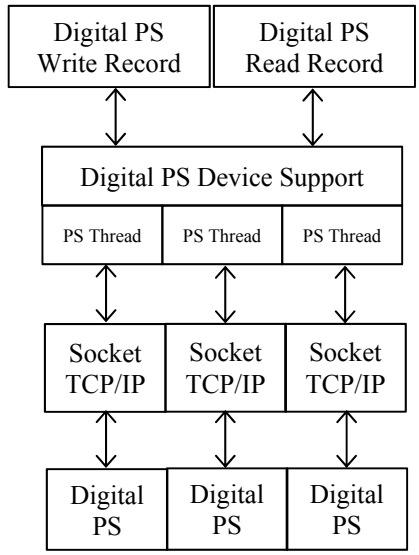


Figure 3: New structure of the PS IOC.

Every PS thread just connect to one PS and when the useful data is ready the function scanIoRequest () to process associated records. Figure 3 shows the structure of the new device support for digital PS.

CONCLUSION

The origin structure of PS IOC uses periodic method. All processing is done on a single thread. So any digital PS connect block will cause the entire IOC to become unresponsive [2].

New design uses separated thread to interact with each one of PS and results will push into record automatically. The new design solves the problem and make some new advantages.

First advantage is more real-time. Digital PS thread could run in a very fast loop and don't affect each other. So the up level records could get the data faster than before.

Second is more stability. One thread crash will not affect other thread, every thread just like run in a sandbox. Whole system become more stability.

REFERENCES

[1] He Yuan, Wang Zhijun, etc. "6-14 Design of Linear Injector for SSC", IMP & HIRFL Annual Report, 2009, P(256-257).

[2] Michael Davidsaver. "Basic EPICS Device Support", https://pubweb.bnl.gov/~mdavidsaver/#doc

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

MAGNET CORRECTOR POWER SUPPLY CONTROLLER FOR LCLS-I

S. Babel, K. Luchini, J. Olsen, T. Straumann, E. Williams, C. Yee, B. Lam, SLAC, Menlo Park, California, USA

Abstract

The MCOR 12 is a 16-channel modular architecture, precision magnet driver, capable of providing bipolar output currents in the range from –12A to +12A. A single, unregulated bulk power supply provides the main DC power for the entire crate. Currently the MCORs have a 1000ppm regulation on the B-field. The MCOR controller card upgrades, existing LCLS-I and future LCLS-II needed, controls for Magnet Corrector Power Supplies. The project shifts the existing functionality of the VME based DAC and SAM and an Allen Bradley PLC into a new slot-0 card residing in the MCOR chassis. Elimination of the VME crate and the PLC will free up rack space to be used in future. The new interface card has a long term stability of 100ppm and monitors ground fault currents and various other interlocks for the MCOR power supplies. The controller can interface to EPICS Channel Access and Fast Feedback system at SLAC using two Gigabit Ethernet ports and has an FPGA based EVR for getting "time stamps" from the Event Generator system at SLAC. The EPICS control system along with embedded diagnostic features will allow for enhanced remote control and monitoring of the power supplies.

INTRODUCTION

The MCOR (Magnet Corrector) Series is a multichannel corrector magnet driver system, capable of providing precision bi-polar output currents with minimal zero-crossover distortion. The MCOR design employs a modular architecture, consisting of a rack-mounted crate, with standardized slots for removable power modules, crate controller and diagnostics. A single, unregulated bulk power supply provides the main DC power for the entire crate. The MCOR power module is responsible for converting the unregulated DC bulk power into a precision bi-polar current source suitable for driving corrector magnets and beam line devices. The power modules are controlled using +10 to –10V FS analog command signals sent over the backplane from the control system. There are two types of available MCOR power modules: The MCOR12 (12A FS) and the MCOR30 as shown in Fig. 1 (30A FS). [1, 2] The MCOR30 uses a custom H-bridge configuration, operating in an overlapping volt-seconds feedback mode instead of a straight voltage mode to achieve a higher control bandwidth. The transfer function of input command voltage to output current is determined by resistor values on the programming (PGM) card. Each power module provides two independent measurements of its output current; one that closes the regulation loop, and another that returns an independent monitoring signal to the control system. [1, 2] Both of these independent current measurements are available to the control card for diagnostic purposes. [1, 2] The MCOR12 uses a commercial switch mode servo driver (The AMC 30A8) operating in a voltage feedback mode to regulate the output current. The MCOR 12 system as the name suggests is a 16-channel precision magnet driver, capable of providing bipolar output currents in the range from –12A to +12A. The output current can be adjusted smoothly through zero. [1, 2] The MCOR 12 employs a modular architecture, so that any individual channel is serviceable without disturbing the operation of adjacent channels in the same crate. This feature significantly improves the overall availability of the accelerator, since in most cases the beam lattice can tolerate the loss of a single corrector and continue to operate, but could not handle the loss of an entire crate of correctors during the repair effort. [1, 2]

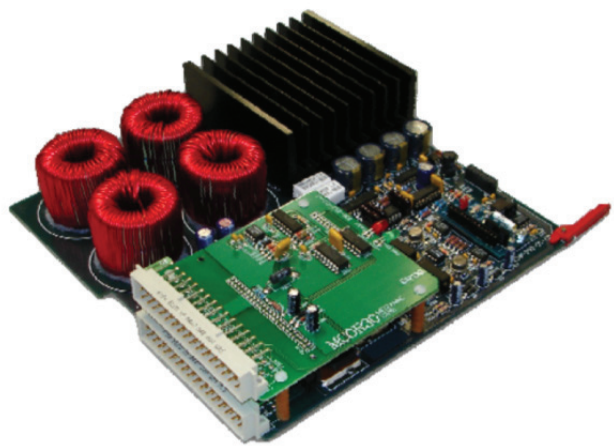


Figure 1: MCOR-30 Power Supply Card.

SYSTEM ARCHITECTURE

The LCLS Fast Feedback Loop Controller wants all MCOR controllers to be controllable at 120Hz (The MCOR Controller card must be able to write set points and read back at 120Hz). The MCOR Controller card must also receive set points in PVs from the Channel Access Network, but it is not necessary that it act on commands from both the Channel Access and the Fast Feedback Network at the same time. LCLS Users want to see the current set point ('desired value'), read back ('actual' value), and status of each channel, regardless of which network the set point command came from. LCLS Users want features like Calibration of the DACs,

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Ramping, Reset Interlocks, Ripple measurement for each MCOR corrector channel, so the controller card supports it. MCOR Controller card as shown in Fig. 2 has Gigabit Ethernet links to both Fast Feedback network and Channel Access to reduce latency. The MCOR Controller card has a long term temperature stability requirement of 0.01 % of maximum current for each MCOR channel. (100 ppm). The DAC accuracy for a desired set point is 0.1% [1000 ppm] of full scale output voltage. The MCOR controller card is able to operate within the ambient conditions 40 degree F to 122 Degree F (50 degree C).The controller card is able to read both the "Monitor" and "Feedback" analog outputs along with the digital "Fault" status from each MCOR module and an alarm is raised if any of the MCOR channels indicate a digital fault condition. There are 32 channels of 18 bit 100 KSps ADCs and 16 channels of 16 bit DAC outputs with settling time of less than 10uS interfaced to a Xilinx Virtex-5 FPGA. The controller card is able to do waveform capture and store for selected MCOR channel for diagnostics. The controller card incorporates an Event Receiver which is able to decode the LCLS Timing System [EVR] to provide timing information. The EVR supports Multi Mode/Single Mode and daisy chaining. The EVR may also have digital TTL outputs for triggering. MCOR Controller card monitors "GND fault current" and triggers an alarm if the GND fault current is not within limits (+/-25mA). It is able to read up to +/-100mA. The Bulk Power Supply is turned off if the GND fault current is not within limits. Beam Synchronous Acquisition (BSA) feature is also desired by LCLS, which is implemented on the EPICS IOC on board the controller card as a daughter card. The controller card uses a Computer On Module (COM) Express Type-2 daughter card as shown in Fig. 3 with an Intel Atom Processor to host the EPICS IOC. The COMX card communicates to a Xilinx Virtex-5 FPGA on the mother board to interface to all the DACs, ADCs, Digital Fault Signals, External Interlocks, Gigabit Ethernet and Voltage, Temp and Current Monitoring chips on the motherboard. The MCOR Controller runs a version of Linux-Real Time Operating system on the COMX based Intel Atom Processor. The Operating system is loaded on the Atom processor at boot-up using TFTP. Controller specific Board Support Package is then used to interface to EPICS. The block diagram is shown in Fig. 4.

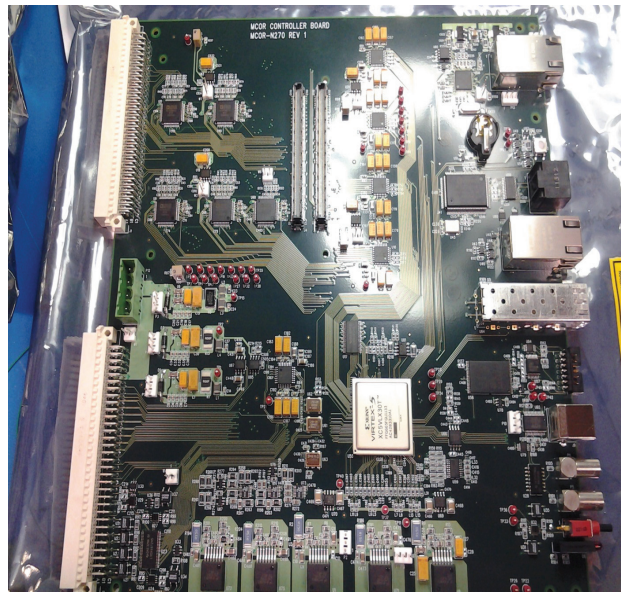


Figure 2: MCOR Controller Motherboard.

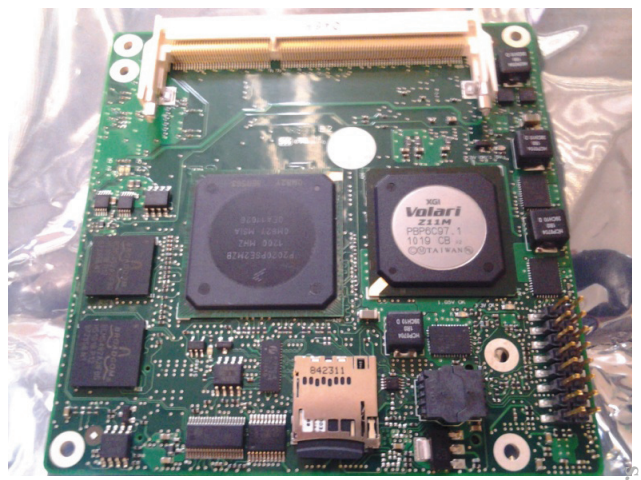


Figure 3: MCOR Controller COMX.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

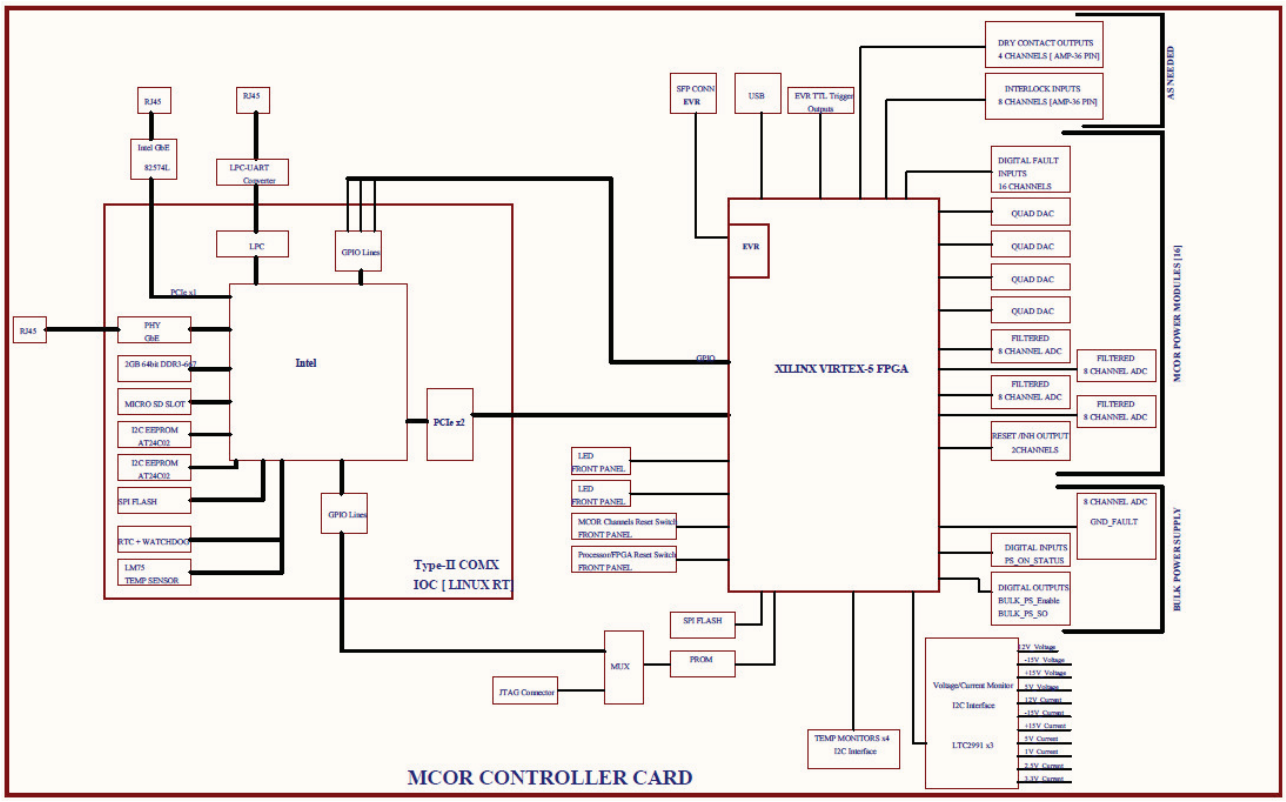


Figure 4: MCOR Controller card Detailed Block Diagram.

TEST AND RESULTS

The MCOR Controller was installed in LCLS-I gallery in Sector-28 of the Linac. Below is the result as shown in Fig. 5 that was obtained for 8 hours duration for MCOR-12 Power supply module.

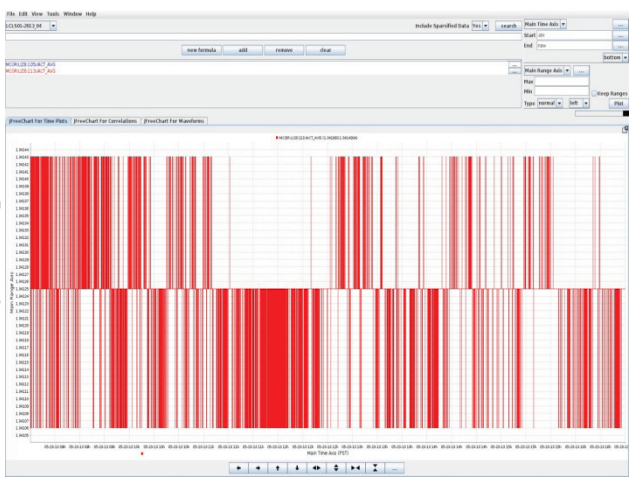


Figure 5: MCOR: LI28 Duration 8 Hrs, Iset: 1.34 A Iact Pk-Pk ripple 375uA.

CONCLUSION

The MCOR controller card upgrades, existing LCLS-I and future LCLS-II needed, controls for Magnet Corrector Power Supplies. The project shifts the existing functionality of the VME based DAC and SAM and an Allen Bradley PLC into a new slot-0 card residing in the MCOR chassis. The new interface card has a long term stability of 100ppm and monitors ground fault currents and various other interlocks for the MCOR power supplies. The controller can interface to EPICS Channel Access and Fast Feedback system at SLAC and has an FPGA based EVR for getting “time stamps” from the Event Generator system at SLAC. The EPICS control system along with embedded diagnostic features will allow for enhanced remote control and monitoring of the power supplies.

REFERENCES

[1] G.E. Leyh, et. al. .A Multi-Channel Corrector Magnet Controller,. PAC 95 and IUPAP, Dallas, Texas, 1-5, May 1995.

[2] S. Babel, S. Cohen, Digital Control Interface for Bipolar Corrector Power Supplies for LCLS, PAC07, Albuquerque, New Mexico, USA.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

RENOVATION OF THE CERN CONTROLS CONFIGURATION SERVICE

L. Burdzanowski, C. Roderick CERN, Geneva, Switzerland

Abstract

The Controls Configuration Service (CCS) is a key component in CERN's data driven accelerator Control System. Based around a central database, the service also provides a range of client APIs and user interfaces - enabling configuration of controls for CERN's accelerator complex. The service has existed for 35 years (29 based on Oracle DBMS) [1]. To cater for changing requirements and technology advances there has been substantial evolution of the CCS over time. Inevitably this has led to increases in CCS complexity and an accumulation of technical debt. These two aspects combined have a negative impact on the flexibility and maintainability of the CCS, leading to a potential bottleneck for Control System evolution. This paper describes on-going renovation efforts (started mid-2014) to tackle the aforementioned issues, whilst ensuring overall system stability. In particular, this paper covers architectural changes, the agile development process in place - bringing users close to the development cycle, and the deterministic approach used to treat technical debt. Collectively these efforts are leading towards a successful renovation of a core element of the Control System.

INTRODUCTION

The CERN Control System is a data-driven multi-layer infrastructure including:

- *Low-level hardware and software* – e.g. timing infrastructure, equipment drivers, Front-End Computers (FEC), end-user developed C/C++ binaries representing operational "devices", etc.
- *Middleware layer* – e.g. read/write access to processes running on FECs and Role Based Access Control (RBAC).
- *High-level software* – e.g. high-level settings management, data acquisition and archiving.

The Controls Configuration Service (CCS) helps bind all of the layers together by providing them with complete and coherent configurations that are necessary for the proper functioning of the Control system [2].

The current architecture of the CCS is based on:

- An Oracle database (2-node RAC cluster)
- A set of high-level client Java APIs
- Database level client APIs (PL/SQL interfaces)
- Numerous Graphical User Interfaces based on proprietary Oracle technologies: Application Development Framework (ADF) and Oracle Application Express (APEX).

The database is implemented using a relation model, with approximately 700 domain tables and ~7GB of core domain data (excluding binary, log and history data – which collectively accounts for ~115GB).

The criticality of the service for safe operation of the accelerators chain is high (though not required for their safe shutdown): The CCS is essential for proper accelerator configuration and start-up – especially during Technical Stops when equipment and other components of the Controls System undergo maintenance and upgrades.

The CCS exists for 35 years, during which the scope, architecture, implementation technology and development methodology have kept evolving. In the middle of 2014 the first major service-wide renovation and overhaul has started – marking the beginning of a new chapter in its long history.

RENOVATION STRATEGY

The motivation behind the complete renovation of the service can be summarized as the need to increase service flexibility while lowering total cost of development, and to advance service functionality to the state required by activities planned for the next CERN Long Shutdown (LS2 – scheduled start early 2019).

The cornerstones of the renovation strategy are:

- Suppression of the technical debt accumulated over the years.
- Changes in the overall architecture
- Adaptation of the Lean software development process [3].

All of these aspects are closely related as suppression of technical debt is essential in order to advance the system architecture, while taking proper architectural and design decisions prevent further "erosion" in the system and limit existing technical debt. The adapted software development process facilitates implementing changes: enabling a lower overall cost of development and increased agility. The first two aspects are a mid–to–long–term perspective (from mid–2014 to the start of LS2). The implementation of the Lean software development process is already well advanced and can be considered finished by the end of 2015.

In order to fully understand the context of the renovation efforts it is necessary to briefly look back at the evolution of the CCS scope and technologies used.

Service Evolution

The scope of the CCS was initially limited to the PS (Proton-Synchrotron) complex controls system, meaning that the service and its database were oriented towards a concrete accelerator and its specific control system. The first relational database was introduced in 1986. Over the years the scope grew following the evolution of CERN's accelerator complex [4]. 1995 marks the introduction of graphical user interfaces (GUI) based on Oracle Forms and PL/SQL Web Toolkit (OWA). The first Java based data access API was implemented in 1999 facilitating access for

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

high-level applications. Starting in 2006, another Oracle based GUI solution (ADF – a Java Server Faces implementation) was put in place to replace existing OWA and Forms applications. In 2009, APEX (a subsequent framework for building database-driven GUI's) was adapted alongside ADF.

Besides the technical changes, the service role and scope has been expanding in recent years:

- Multiple additional device-property models were introduced,
- The Front-End Software Architecture (FESA) framework reached the next major version [5],
- The service incorporated configurations specific to various sub-systems like:
  - Beam Interlock System,
  - Power Converters,
  - Role Based Access Control.

Together with these changes certain functionalities of the service became partially suppressed, specifically legacy high-level settings management, for which the functionality was incorporated into LSA [6] for the majority of the accelerators.

The following patterns emerge when we look back at the aforementioned evolution: reliance on Oracle proprietary GUI technologies, and the progressive growth of the service well beyond its original system design and architecture that was tightly coupled to a specific accelerator. Both of these aspects have contributed to the current state of the service and triggered the renovation.

ADDRESSING TECHNICAL DEBT

The evolution of every complex hardware and software system inevitably results in increased technical debt and progressive erosion. The availability of new technologies and solutions, as well as the human factor of growing experience – keep changing the perception of quality and adequacy of former technical decisions [7].

In most cases, end users are not directly aware of the technical debt but as software engineers we should perceive it as negative value. It is adverse to system architecture and design, which are planned, deliberate and visionary. By clearly establishing the system boundaries and facilities to assess and measure its evolution the technical debt can be addressed whilst minimizing impact on the end users.

Accidental Complexity

At its basis the technical debt is equal to accidental complexity happening in the system. On contrary to deliberately complex solutions for equally complex problems the accidental complexity happens by itself naturally along evolution of the system.

System complexity can be described by both quantitative and qualitative factors. For the CCS, the quantitative factors include: counts of tables, views, triggers, PL/SQL packages, number of grants, grantees, accounts, lines of code, length of packages and procedures, levels of views nesting etc. All such factors can be measured and automatically classified as a potential problem based on established thresholds. The qualitative factors might be based on quantitative data but cannot nor should not be automated. Such factors are based on experience and often common sense – ultimately a human factor. For example, de-normalization of a database table may be justified by performance requirements or for the sake of clarity in a model, although in most cases it is a sign of shortcomings in the design.

The majority of complexity in a database-oriented system is caused by dependencies (direct and indirect), but rarely by algorithmic complexity. Examples of direct dependencies are relations between objects, database views referencing tables or other views, PL/SQL packages executing code based on database structures. Indirect dependencies include aspects like dynamic/background execution or exposure of the database objects to external users (grants) – an inevitable need contributing to coupling between systems.

Targeted Re-Factoring

In order to guarantee system stability during renovation – specifically when re-factoring areas with high technical debt – a methodical targeted approach with a clear strategy is essential. The CCS renovation efforts fall into two categories: targeted and ad-hoc re-factoring.

Ad-hoc refactoring is considered as a natural part of regular development and does not require extensive planning or analysis. For example: small improvements in the code base like eradication of "dead-code", addition of missing test cases, updates to stale documentation, and re-naming inadequately named objects. In general, such activities shouldn't take more than an additional 20% of the base development time needed to deliver the required functionality.

Targeted re-factoring is predefined as a concrete group of tasks based on the following criteria:

1. *Identify boundaries* – to clearly know when the activity should finish.
2. *Identify clear gains* – to justify the effort. The gains should be tangible, based on facts, and ideally quantifiable.
3. *Identify risks* – to know the impact both within – and outside the service.
4. *Define rollback / fall-back strategy* – to limit any potential negative impact, mainly in critical areas.
5. *Estimate and prioritize* – to realistically plan the effort alongside regular development activities.

The value to be gained from the re-factoring can be classified into distinct areas, and includes in descending order of importance:

1. *Consistency* – i.e. limiting the likelihood of data corruption and/or of non-deterministic states.
2. *Performance* – improving the response times for data reporting and querying for clients.
3. *Maintenance* – lowering the total cost of development, likelihood of introducing new errors, and the usage cost paid by clients (e.g. by obscurity APIs or lack of documentation).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

4. *Agility* – ensuring the extendibility of the architecture and limiting the cost / time of delivering new features to clients.

By following the assessment criteria above and understanding their associated gains, it is possible to identify and prioritize renovation activities alongside regular development.

Determinism and Static Code Analysis

Static code analysis (SCA) is the analysis of computer software source code on the contrary to dynamic analysis, which is based on code execution. SCA can be applied to any type of software including databases like Oracle. In this case the analysis includes both database structures (tables, views, constraints, indices, etc.) and executable stored procedural code (PL/SQL packages and triggers).

To match our needs a custom SCA framework has been developed which enables analysis within the database engine. The framework includes a pre-defined set of analysis rules, which can be customized and extended. The analysis can be executed on demand or periodically, and generates reports summarizing the number of rule violations, severity and links to the source. These reports are used to identify areas for in-depth analysis and planning of the re-factoring. Below is an example of a metric indicating the evolution of the count of invalid objects over the past year:

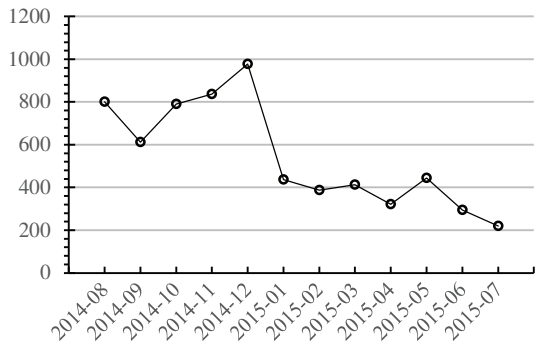


Figure 1: Count of invalid objects in development database, aggregated per months.

The analysis results, fluctuations and evolution of the metrics are the inputs to qualitative assessments and serve as a basis for future planning. The metrics cover various aspects ranging from standardisation concerns (e.g. code not adhering to naming standards, usage of disallowed or obsolete functions, etc.) to abstract complexity of the system (expressed via factors such as code size, table sizes, triggers counts, nesting of database views). With SCA in place we are able to evaluate our efforts over time, relying on factual data rather than assumptions.

ARCHITECTING FOR THE FUTURE

The renovation and supporting changes in system architecture fall into four main categories: suppression of accidental complexity and lowering overall system complexity, context based access to the data, phasing-out of proprietary GUI technologies, and applying a system wide infrastructure for tracing, auditing and monitoring. Combined, these categories form the foundations for simpler, easier and agile development, with an increased quality of service.

The accelerator controls system domain is inherently complex; the corresponding software components are therefore inevitably complex accordingly. During the process of suppressing accidental complexity / lowering overall complexity, we have started to progressively adapt an event driven architecture which has proved to increase cohesion and lowered coupling of system components. New developments and on-going re-factoring conforms to GRASP [8] (General Responsibility Assignment Software Patterns) patterns of Object-Oriented design, tailored to the world of relational databases. To support these changes we have adapted Commons4Oracle (C4O) – a set of PL/SQL libraries for Oracle database, which is actively developed in the CERN Controls group. The library assures further standardization and foundations for future development. Moreover it streamlines solutions in the CCS with other core database projects of the Controls group thus enabling transfer of knowledge and expertise.

Based on direct feedback from CCS users and domain experts we are gradually increasing the scope of context surrounding the data entreated to the system. By utilizing Commons4Oracle extensions, a set of high-level domain specific events is being implemented. Triggering of such events can be based on direct user actions, or a workflow based transition. By attaching state information to core domain entities in the system (e.g. devices), we can now automatically notify users interested in a given “domain event”. For example when a computer hosting a device changes its state, the responsible for the device can be notified and take actions if necessary. It is important to add that such notifications may be filtered based on criteria specified by individual users.

The dedicated CCS GUIs are the primary means for end users to access or edit data. On average per day there are over 150 distinct user sessions (from a total of ~400 distinct registered users). Experiences of past years as well as user feedback contributed to the decision to phase-out the existing proprietary technologies in favour of widely adapted solutions of Java based RESTfull services and HTML5/JavaScript web interfaces. In addition this technology stack steadily gains popularity within the software engineering community and in turn facilitates hiring of well-trained specialists.

With tracing and auditing extensions in place, not only the time or user behind a given action is captured, but also contextual information like client IP address, database session and transaction IDs, and name of the program unit and invoked action. This information is used to augment historical data tracing, with which changes to any entity in the system are precisely tracked and can be presented back to users in a context specific manner (e.g. as a log of differences starting from a specific moment in time or by presenting dependencies between objects in a human readable way). This system-wide architectural extension

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

considerably limits the time needed to support users in investigating suspected data problems, and potentially recovering data.

PRAGMATIC AGILITY

As a part of CERN Controls System the CCS is actively used and maintained on regular basis. Its development lifecycle is similar to software products released to end-users that have continuous support for the new extensions and improvements. Such a lifecycle requires responsiveness when addressing end-user's needs but also requires realistic planning which fits into the tight schedule of accelerators operations. The challenges of undergoing renovation and aforementioned aspects motivated the adoption of a Kanban development process [9].

The Kanban emphasizes focusing on continuous improvement, importance of human factors and bringing maximum value to the organisation. Over just a few months the new Kanban development process has been implemented, and in less than six months the efficiency of the team increased noticeably. By visualising the work on a Kanban board, bottlenecks were quickly identified (e.g. too much work in progress, too many unrelated tasks started, or too many new features waiting in quality assurance queue). By not relying on fixed development iterations / sprints – trust from end-users increased as their requested features and bug-fixes are not systematically subjected to prolonged wait times due to extensively planned ahead sprints. The agility and reactivity of the team and CCS as a whole has increased thanks to the Kanban / Lean philosophy of just-in-time delivery and constant focus on activities that bring the most value to end-users. By separating planning and reviews into short-term (weekly / monthly) and mid- term (quarterly / yearly) the CCS team balances providing new functionality within realistic time frames with reacting quickly (hours instead of days) in case of urgent problems.

Thanks to the new methodology, the overall development throughput has increased. More importantly the satisfaction of both CCS users and team members has increased. Changing the way tasks are prioritized and visualized has led to a reduction in pressure and stress on developers. CCS end-users are now much more closely involved in the development process and act as true stakeholders thanks to effective visualization of work in progress and clearly identified stages of the development cycle. These human factors are proving to be essential to the success of the on-going renovation.

In retrospective, the adaptation of a Kanban approach has already resulted in high returns on the time invested into configuring supporting tools to the CCS team needs, and regular discussions and analysis of the changes being implemented. The returns from the Kanban are that more time is spent on delivering actual value, and assuring that new features and improvements are not over-planned nor hurried to production (which would result in faults and frustration of the end-users). Regular retrospectives and critical analysis of changes applied to the working process have positively transformed the way the CCS team works.

CONCLUSIONS

The renovation of a mission critical service with many years of history is a challenge. Alongside changing requirements, growing expectations and needs to consolidate various sub-systems of the Control System, the CCS started to play an even more important role during recent years. The necessity to adapt to these changes and satisfy new requirements is the driver for the on-going CCS renovation. Progressively reducing technical debt increases overall agility, but more importantly it also helps to design a better system for the future. CCS users now have a much better understanding than previously of the value of these changes and together with their increased satisfaction the renovation and technical debt reduction is perceived as added value. The Kanban way noticeably improved the CCS team efficiency and contributed to increased end-user satisfaction. New architecture solutions lay foundations for an advanced, cohesive and agile system that embraces the context and workflows of how CCS users work. The renovation started over a year ago and marked the beginning of a new and exciting era in the long history of the Controls Configuration Service of the CERN Controls system.

REFERENCES

[1] J. Cuperus et al., ICALEPCS1997 – ID085.
[2] R. Gorbonosov, *The Control Systems of the Large Hadron Collider*, CERN Academic Training Lecture Regular Program, http://cds.cern.ch/
[3] T. Ohno, *Toyota Production System: Beyond Large-Scale Production*, ISBN 978-0-915299-14-0, Productivity Press, (1998).
[4] J. Cuperus et al., ICALEPCS2003 – WE114.
[5] M. Arruat et al., ICALEPCS2007 – WOPA04.
[6] G. Kruk et al., ICALEPCS2013 – MOCOBAB05.
[7] MM. Lehman, *Laws of Software Evolution Revisited*, EWSPT '96.
[8] C. Larman, *Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd ed.), ISBN 0-13-148906-2, Prentice Hall, (2005) [2004].
[9] H. Kniberg, *Lean from the Trenches: Managing Large-Scale Projects with Kanban* (1st ed.), ISBN 978-1934356852, Pragmatic Bookshelf, (2011).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

EMBEDDED ENVIRONMENT WITH EPICS SUPPORT FOR CONTROL APPLICATIONS

Y. S. Cheng, Demi Lee, C. Y. Liao, C. H. Huang, K. T. Hsu
National Synchrotron Radiation Research Center, Hsinchu 30076, Taiwan

Abstract

System on a chip (SoC) is widely used in embedded environment. Current generation SoC commercial products with small footprint and low-cost have powerful in CPU performance and rich interface solution to support many control applications. To deal with some embedded control applications, the "Banana Pi" which is a card-size single-board computer and runs Linux-based operation system has been adopted as the EPICS IOC to implement several applications. The efforts for implementing are summarized in this paper.

INTRODUCTION

Using a circuit board to implement functions as a computer is called SBC (single-board computer) [1]. Its applications cover in telecommunications, industrial control, blade and high density servers, and lately laptops and mini-PCs, etc. Thanks to latest generation SoC technology, putting all major functionality into an integrated chip, educational used credit-card size SBC [2] likes the Raspberry Pi (RPi) and BeagleBone Black (BBB) are highly successful products. The Banana Pi (BPi) is the latest product of such category with powerful CPU, low power consumption SBC indeed, and the area of circuit board is only as credit card size.

The Banana Pi which design idea is similar to the RPi-style SBC, and it is a fork of the RPi project using different components while maintain compatibility as much as possible. Moreover the Banana Pi is added the functions of SATA interface, infrared transmission, microphone, USB-OTG ports, power button, reset button, etc. Then the BPi has 26-pin/40-pin GPIO which is compatible with the RPi. The A20/31 SoC as CPU/GPU, 1GB DDR3 memory and Gigabit Ethernet connection are applied on the Banana Pi. The hardware specification of Banana Pi is shown as Table 1 [3-4]. Linux-based OS can be worked well on the Banana Pi.

Table 1: Hardware specification of the Banana Pi

	Banana Pi M1/M1+/M2
CPU	A20 ARM Coretx-A7 1GHz Dual-Core A31S ARM Coretx-A7 1GHz Quad-Core
Memory	1GB DDR3 DRAM
Network	1Gbps Ethernet RJ45, Wi-Fi
Storage	SD card slot (up to 64GB), Extensible with SATA interface
I/O	GPIO, UART, I2C bus, SPI bus with two chip selects, CAN bus, ADC, PWM, +3.3V, +5V, GND
OS	Debian, Bananian, Lubuntu, Android ...

The EPICS (Experimental Physics and Industrial Control System) [5] is a set of open source software tools, libraries and applications developed collaboratively and used to create distributed soft real-time control systems for scientific instruments such as particle accelerators. Many facilities have good practical experiences for the EPICS and adopt it as accelerator control systems. Many resources and supports are available as well as numerous applications for accelerator have been developed.

The TPS (Taiwan Photon Source) control system of 3 GeV synchrotron light source is also based on the EPICS framework [6]. The EPICS toolkit provides standard tools for display creation, archiving, alarm handling, etc. The big success of EPICS is based on the definition of a standard IOC (Input Output Controller) structure together with an extensive library of driver software for a wide range of I/O cards. The EPICS framework which has various functionalities is employed to monitor and to control on embedded applications of accelerator system.

BANANA PI AS EPICS IOC

The stability and performance of Banana Pi is enough as the EPICS IOCs for specific control applications. The EPICS framework can be built on the Linux-based Banana Pi successfully [7]. At the TPS project, some control functions, such as frequency divider, direct digital synthesizer, radiation-sensing reader, alarm announcer, etc., are implemented by use of the Banana Pi platforms with EPICS support. The efforts for implementing are summarized as followings.

Software Architecture

To implement the Banana Pi as the EPICS IOC for specific control applications, the EPICS base and modules are necessary to be set up on the Banana Pi platform which operation system can be the Debian or Ubuntu Linux. The device driver of SPI (Serial Peripheral Interface) bus is built for communicating with DAC/ADC modules, and the device support interface is also developed as the glue between the EPICS records and device drivers. The EPICS records support with databases are created according to the specific functions. The application module, such as “autosave” function, is installed for logging setting parameters values and recovering last setting parameters values automatically when the IOC is start-up. Based on the EPICS PV (Process Variable) channel access, the archive server is set up to record various parameters variations for long time observation, and the PHP webpage can be developed to show the status information. At the client console side, the operation interfaces are created by used of the EDM, CS-Studio, etc. to control and monitor via EPICS PV channel

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

access, and the archived data can be retrieved with using a form of graphical representation of the CS-Studio based data browser. The schematic is shown as Fig. 1.

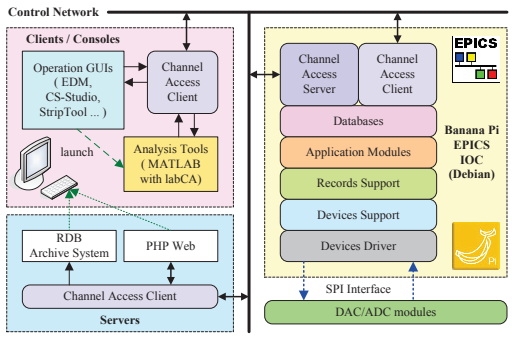


Figure 1: Software architecture of the Banana Pi with EPICS support.

Programmable Frequency Divider

Machine clock of the accelerator system generated discrete fast logic chip (ECL/PECL) or combined of fast logic and field programmable logic array (FPGA) usually. Typical jitter is in a few picoseconds order. The programmable clock generator has been implemented by using the AD9508 clock and delay generator to generate clock with 100 femtosecond jitter for some applications (laser clock, filling pattern measurement timing, etc.). The system schematic is shown as shown in Fig. 2. The chip divider and delay parameters can be controlled by use of the Banana Pi EPIC IOC via SPI interface. The implementation is shown as in Fig. 3.

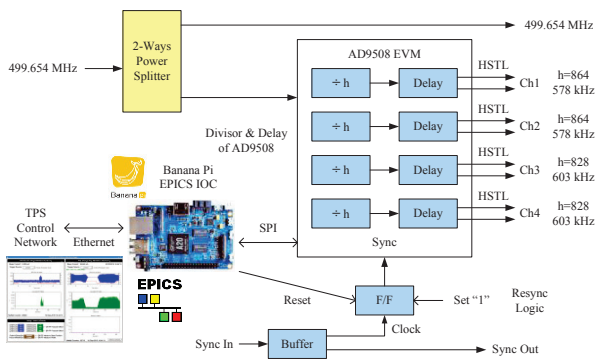


Figure 2: Block diagram of the programmable clock generator.

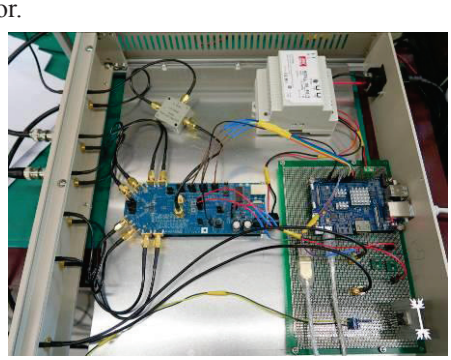


Figure 3: Photo of frequency divider unit for the TPS filling pattern measurement timing.

Direct Digital Synthesizer Control

To make possibility of different RF frequency without a similar multiplication factor (six) work for linear accelerator (Linac) and booster synchrotron to optimize machine performance without adjust too many parameters in Linac system, a RF signal generator direct digital synthesizer (DDS) which can synchronize at injection instance have been implemented. Functional block diagram of the prototype is shown in Fig. 4. The Banana Pi EPICS IOC is used to control the DDS to achieve goal. Synchronization is achieved to reset the phase of the DDS just before booster synchrotron injection to ensure constant phase relationship between RF system of Linac and booster synchrotron. Figure 5 shows the prototype DDS signal generator.

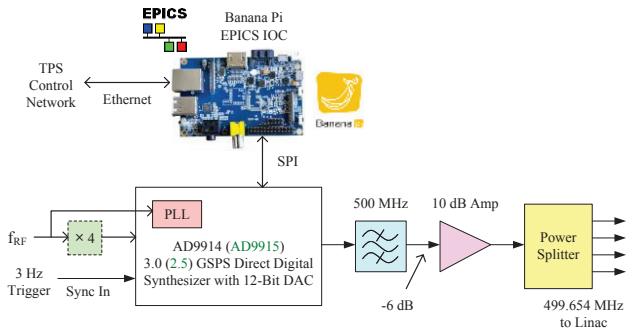


Figure 4: Block diagram of the direct digital synthesizer control.

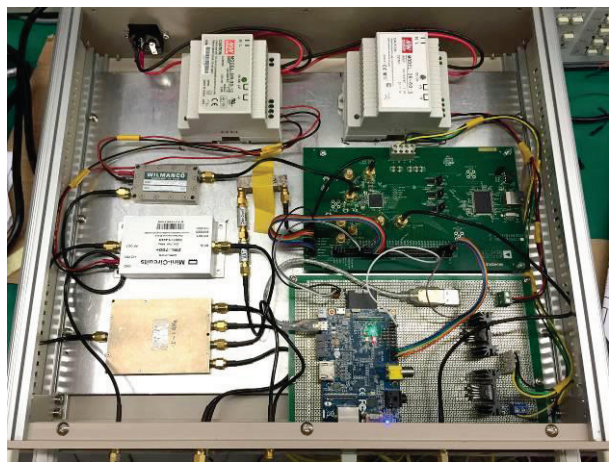


Figure 5: Photo of the prototype DDS signal generator.

RadFET Reader

To investigate the beam loss and its distribution during commissioning and operation phases of TPS and TLS, a sixteen-channel readout box was initially designed and implemented to read the threshold voltage of the RadFETs (radiation-sensing field-effect transistor) which were installed at accelerator tunnel [8]. The initial version design was that the reader plays a role of remote I/O for the EPICS IOC and the IOC collects voltage from readers distributed at the accelerator to deduce the integrated dose and dose rate.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The next version design is that the EPICS IOC will be embedded into the RadFET reader box. The Banana Pi will be also adopted as the EPICS IOC for collecting the threshold voltage of the sixteen-channel RadFETs. The data transmission time between the IOC and SPI bus with ADC modules will be improved.

The EPICS IOC performs data acquisition, calculation, and publishes the specific EPICS PVs of dosage. Dosage rate is calculated by the EPICS record processing. All of the threshold voltage values based on the EPICS PVs channel access can be recorded into the archive server for further off-line data processing. The MATLAB toolkit can be also used to analyze the RadFET threshold voltage archived data which retrieved from the RDB archive system directly. The control system also provides on-line display for virtualization usage. The system schematic of RadFET reader is shown as Fig. 6. The test prototype of RadFET reader with the Banana Pi EPICS IOC is shown as Fig. 7.

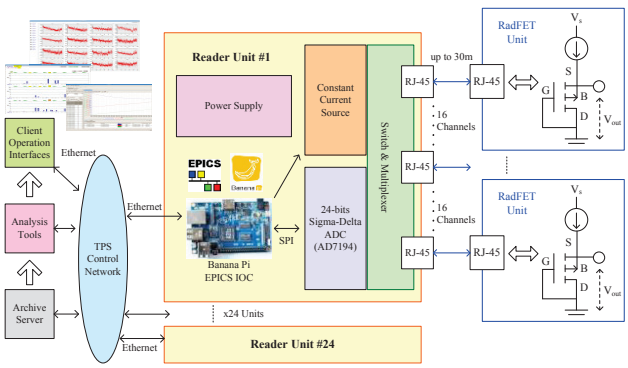


Figure 6: Block diagram of the RadFET readers system.

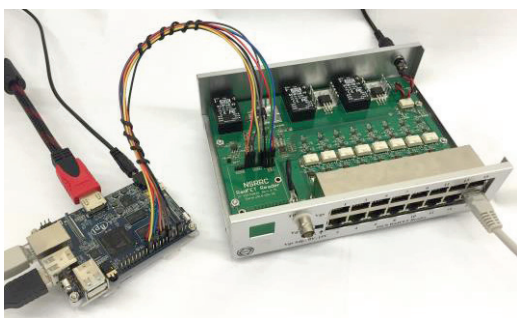


Figure 7: Photo of the prototype 16 channels RadFET reader with Banana Pi EPICS IOC.

Alarm Announcer

During the TPS commissioning and operation phases, the abnormal status may occur from one of sub-systems, and operators need to find out which sub-system problem happened from machine interlock interface. Due to many interlock signals need to be noticed, the sum signals of each interlock signals are necessary. According to the sum signals, the specific alarm message to be trigged and shown, and the Banana Pi is used as the EPICS IOC to receive the request and send alarm announcement sound to loud speaker for noticing. The system schematic is shown as Fig. 8.

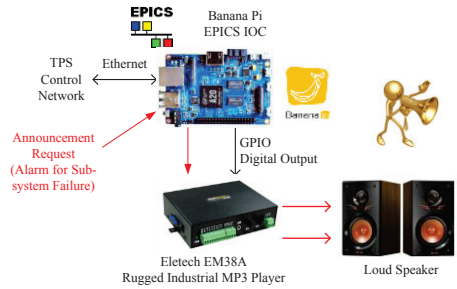


Figure 8: Block diagram of alarm announcer.

BEAGLEBONE BLACK APPLICATIONS

Vibration monitoring and voltage signal monitoring play an important role in the accelerators system for beam stability characterization. A commercial product DT7837 was adopted for this purpose. This device is a high accuracy dynamic signal acquisition module for vibration, and voltage signal measurements by the BeagleBone Black SBC [9]. Four 24-bit voltage input or IEPE sensor input data can be algorithmically processed in real time and the results presented to a host for analysis. Support of Linux-based OS and software development kit includes numerous components help the development. The EPICS is installed into the DT8837 for integration and data access. Synchronization interface is easy to integrate with accelerator timing system. Coherent data acquisition is under implementation.

CURRENT STATUS

Low cost credit-card size SBC is widely adopted for educational purpose and also suitable for small scale embedded applications. The BPi and BBB are chosen for several applications at the TPS control environment as auxiliary supports which are not suitable to use standard platform in existed control system due to economics, simplicity, speciality view points. More applications will be explored and implemented in near future.

REFERENCES

[1] Single-board computer website: https://en.wikipedia.org/wiki/Single-board_computer
[2] Educational single board computer website: http://edutechwiki.unige.ch/en/Educational_single_board_computer
[3] Banana Pi website: http://www.banana-pi.org
[4] Banana Pi website: https://en.wikipedia.org/wiki/Banana_Pi
[5] EPICS website: http://www.aps.anl.gov/epics/
[6] C. Y. Liao, et al., "Commissioning of the TPS Control System", FRB3O01, these proceedings, ICALEPCS'15, Melbourne, Australia (2015).
[7] EPICS Pi website: http://prjemian.github.io/epicspi/
[8] D. Lee, et al., "Online RadFET Reader for Beam Loss Monitoring System", MOPTY070, Proceedings of the IPAC'15, Richmond, USA (2015).
[9] DT7837 website: http://linuxgizmos.com/signal-analyzer-runs-linux-on-beaglebone-black-like-core/

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

LLRF CONTROLS UPGRADE FOR THE LCLS XTCAV PROJECT AT SLAC*

S.Condamoor#, Y.Ding, P.Krejcik, H.Loos, T.J.Maxwell, J.J.Olsen
SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA

Abstract

SLAC's Low Level Radio Frequency (LLRF) controls software for the S-Band deflecting structures needed to be upgraded significantly when a new X-Band transverse deflecting cavity (XTCAV) was installed downstream of the LCLS undulators in Spring 2013 to assist in FEL diagnostics such as characterizing the temporal profile of X-ray pulses that vary shot-to-shot. The unique location of the XTCAV in the beamline posed several challenges. A new design of the Modulator and Klystron control Support Unit (MKSU-II) for interlocking was added at the XTCAV controls station that required new software development. The timing setup was also different from the rest of the Linac. This paper outlines the LLRF controls layout for the XTCAV and discusses the manner in which the challenges were addressed. XTCAV has now become a successful tool for gathering data that enables reconstruction of X-ray FEL power profiles with greater resolution.

XTCAV FOR DIAGNOSTICS

XTCAV is a powerful diagnostic tool for SLAC's Linac Coherent Light Source (LCLS) Free Electron X-ray Laser (FEL). This is a successor to SLAC's historic S-Band TCAV[1], two of which are still used in LCLS for electron beam bunch length measurements. XTCAV provides useful FEL diagnostics without interfering with LCLS operations since it is installed downstream of the LCLS undulators that are the source of the x-ray pulses. XTCAV deflects spent e-beam bunches after they have contributed to the lasing process just after they exit the undulators. Energy spectrometer is then used to capture and analyse in detail the profile of this spent e-beam stream on a bunch-by-bunch basis. Comparison of the longitudinal profile of these electron bunches when they are lasing against when lasing is suppressed allows us to reconstruct not only the longitudinal profile of the incoming electron beam but the time-resolved profile of the X-ray pulses generated during the lasing process. Figure 1 shows the general layout of this diagnostics system.

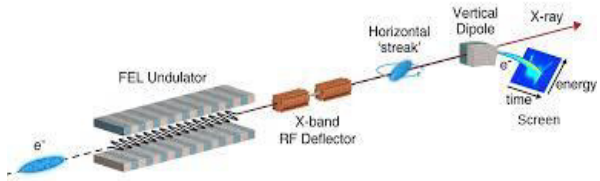


Figure 1: The system includes the transverse deflector "XTCAV", the magnetic spectrometer and the Ce:YAG screen located downstream of the FEL undulators. Horizontal streaking is followed by a vertical-bend dipole magnet for measuring the energy spectrum. A camera captures the transverse images of the electron beam density distribution on the diagnostic screen. (Image: Y.Ding et al., PRSTAB 14, 2011).

XTCAV'S HIGH POWER RF

The SLAC-designed X-Band RF cavity shown in Figure 2 below is powered by SLAC's XL4 50 MW X-Band Klystron that deflects the beam by giving it a 48 MV kick[2]. The Klystron operates at 120 Hz and delivers pulses that can be up to 1.5 uS long. The structure fill time is 200 nS. A legacy 6575 modulator was reused from SLAC, the controls for which were upgraded recently.

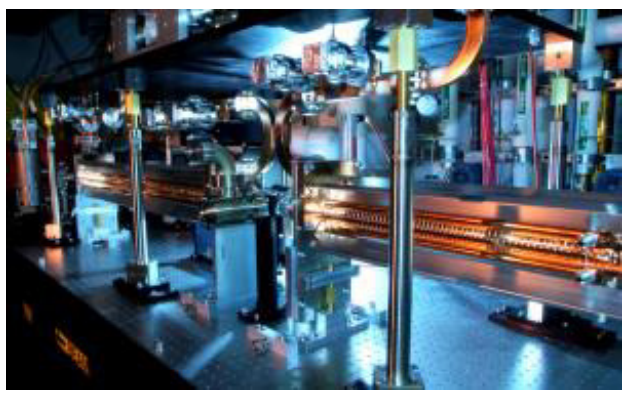


Figure 2: Two one-meter long X-Band RF structures are in the beamline's path towards the e-beam dump. (Image: P.Krejcik/SLAC).

XTCAV CONTROLS

The Controls hardware for XTCAV is located in a service building above the cavity in the tunnel. In addition to the klystron, this building houses the modulator, LLRF controls hardware, power supplies, waveguides, vacuum, temperature and water controls. The LLRF phase reference is derived from the 476 MHz Linac Reference via a heliax cable from the tunnel. Timing and trigger generation for controls is obtained with an Event Receiver

* Work is supported by the U.S. Department of Energy, Office of Science under Contract DE-AC02-76SF00515.
#scondam@slac.stanford.edu

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

module which receives precisely timed event codes from the LCLS Event Generator (EVG) on optical fibre links. Networking and serial communication devices are also located in this building. An upgraded model of the MKSU-II keeps the HPRF devices safe by interlocking the whole system with fault detection and protection.

XTCAV LLRF CONTROLS LAYOUT AND ITS CHALLENGES

Implementing the controls for the XTCAV station posed several challenges due to the unique nature of its location in the beamline. It was a structure that was added later to LCLS which meant that the supporting infrastructure needed for this add-on device did not exist when the upgrade project began.

XTCAV Location

While most of the LCLS RF controls are located either in the temperature controlled RF Hut or in the RF stations along the Linac's gallery, the XTCAV RF controls are located in a service support building farthest away from the rest of RF stations towards the end of the Linac.

This resulted in several key differences in the way the XTCAV station is setup as opposed to the other LCLS RF stations.

All other RF stations use the 476 MHz Phase Reference and other X-Band or S-Band RF signals that are distributed from a central location, amplified or boosted and synchronized locally as needed along the Linac. The XTCAV is located close to the end of the beam line and much beyond where the last of the RF stations end in Sector 30 of the Linac. This necessitated that the phase reference and other RF signals needed for the XTCAV station be derived locally. This problem was overcome by sharing the phase-stabilized 476 MHz Reference line from the tunnel that was used for the beam phase monitor in the vicinity of XTCAV. This Phase Reference was brought up to the service building via low-noise heliax cables. A local frequency generator used this reference to derive X-Band LO, Clock and all other RF which were then amplified and distributed locally.

LLRF Controls

In general, LCLS LLRF controls for each RF station comprises of an S-Band or X-Band Phase and Amplitude Controller (PAC), a Phase and Amplitude Detector (PAD) and an MKSU for modulator and klystron fault detection and protection. It may have an additional diagnostics PAD for monitoring the RF output from the PAC and the klystron forward and reflected power. A Master Controller controls a group of PACs and PADs at several stations to provide local or beam-based feedback. Figure 3 is a block diagram of the LLRF controls in service building B921.

Temperature and Feedback

The phase and amplitude stability provided by the closed loop LLRF feedback controls is tied very closely to the temperature of both the structure as well as the LLRF hardware. Unlike the rest of the Linac where the key feedback PAD digitizer is located in a temperature controlled environment to guarantee phase and amplitude stability within the tolerance limits, the XTCAV feedback PAD is located in s service building that is not temperature controlled and thus prone to ambient diurnal temperature variations. The X-Band PAC and PAD's temperature is regulated with water which minimizes this issue. Improvements to feedback algorithm further improved stability. The behaviour of feedback is also closely related to the overall accuracy of timing at the station which will be discussed next.

XTCAV TIMING

The PAC and PAD are controlled by an EPICS IOC running in a PowerPC based CPU board in a VME crate.

XTCAV was the first and is still the only LCLS RF station that has all EPICS controls and the new MKSU-II. This station does not have any of the older SCP, CAMAC or PIOP related controls that are used in combination with EPICS in several portions of the Linac. This minimized XTCAV station setup complexity but at the same time posed newer system integration challenges and some training challenges for the operators.

Unlike the rest of the LCLS controls which use just an EVR module, the XTCAV VME crate has the EVR-RF module for timing. This board has an event clock that is divided from the externally input 119 MHz RF signal. This 119 MHz RF signal is derived directly from the same 476 MHz Phase Reference which was used to derive locally all LLRF RF signals and clock for the PACs and PADs as discussed earlier. This special setup with the EVR-RF module ensures the best possible synchronization between local RF and timing system and provides accurate timing and triggers for all HPRF and LLRF components at the station. This is paramount for keeping the phase and amplitude of the very narrow X-band pulses stable since the XTCAV is located near the beam dump where tolerances for reference phase deviations may not be as stringent as in the rest of the beamline in the Linac.

The trigger setup complexity is minimized in the XTCAV station. Unlike the rest of the RF stations where a number of different triggers from a variety of sources provide discrete triggers for the modulator, PACs and PADs, and whose delays have to be setup up carefully in careful coordination, the XTCAV station takes in a single trigger output from the EVR and fans this out to all the LLRF HW via the MKSU-II that automatically routes the appropriate trigger based on the station's operating mode which can either be beam 'Accelerate' or 'Standby'.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

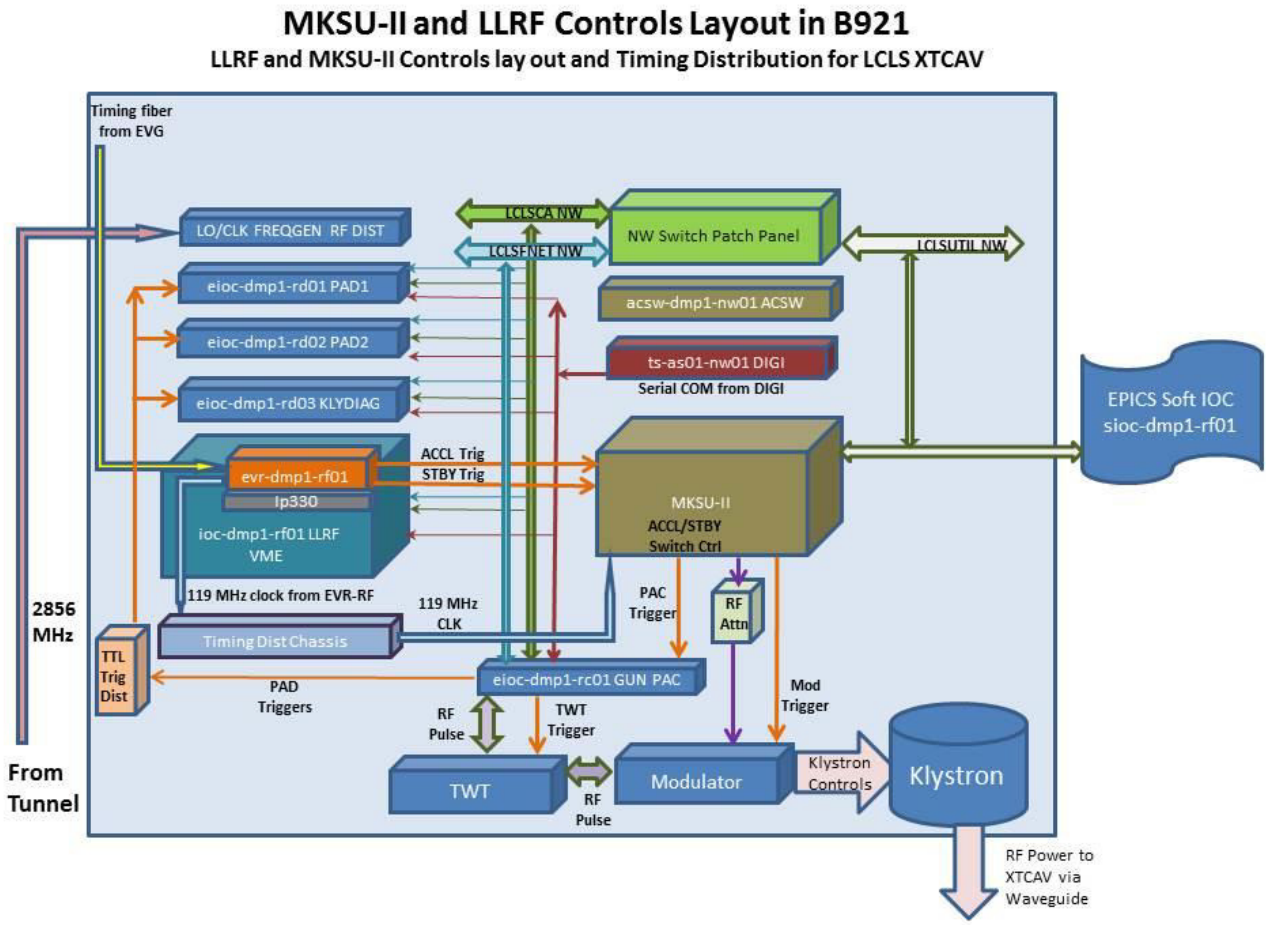


Figure 3: Block diagram of LLRF and HPRF controls layout in XTCAV service building.

XTCAV LLRF SOFTWARE UPGRADES

The XTCAV station was the first RF station that adopted a new LLRF Software model that overcame several limitations and issues in the existing design. The PAD was designed primarily to down-mix and digitize the raw LLRF signals such that I and Q can be derived from the raw waveforms for phase and amplitude feedback control. In the original software, the IQ calculations were performed within the PAD which faced several limitations and issues due to the low processing power of the PAD CPU at higher trigger rates. To overcome this limitation, a new software model was implemented where the PAD was relegated to its primary role of digitizing the raw signal and all CPU-intensive calculations were offloaded to the master VME controller. This vastly improved the performance of LLRF controls software. This upgraded software model was first tried out at the XTCAV station since it was considered a diagnostics device, thus not invasive to the beam and was not expected to interfere with LCLS operations. Following the success of this new LLRF software design at XTCAV, the same model was used to upgrade all other EPICS-controlled LCLS RF stations.

MKSU-II

The XTCAV is the only LCLS station that has an upgraded Modulator Klystron Support Unit called as the MKSU-II. This chassis protects the Klystron by monitoring various sensors and removing high power pulse to the klystron when the system is compromised. The MKSU-II hardware and firmware were modified specifically for the LCLS XTCAV project. A new driver and accompanying business logic were written for it with a new user interface. The new UI mimicked the chassis front panel and provided a remote view of the current status of HPRF at the station. Among several other new features, the MKSU-II supports continuous monitoring of fast waveforms such as the klystron beam voltage and current and forward and reflected power. The MKSU-II chassis front panel, its user interface and fast waveforms are shown in Figure 4. XTCAV's new MKSU-II was clearly a big improvement in RF station fault detection and protection albeit the transition to this new hardware in a station that already differed from the rest of the RF stations in several aspects posed some challenges for operations. Training has improved adaptation.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

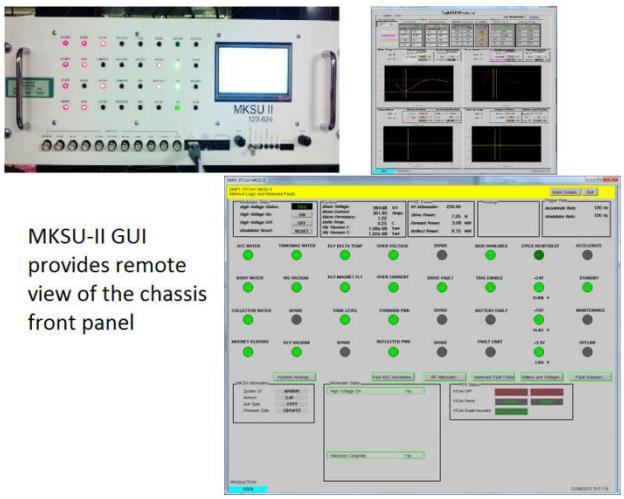


Figure 4: MKSU-II chassis front panel, user interface and fast waveform panel.

CONCLUSION

Commissioning of the controls for XTCAV came with its own set of unique challenges as outlined in this paper but each one of them were overcome in a manner that actually set the path for improvements in LLRF controls for the rest of the Linac. Ever since it was commissioned in May 2013, the XTCAV has become a successful tool in gathering data that enables reconstruction of X-ray FEL profiles with greater resolution. Its success has called for further experimental upgrades which are already underway with the installation of a new SLED cavity at the station that is expected to double the peak power at the structure with longer RF pulses.

ACKNOWLEDGMENT

Several SLAC engineers contributed towards the XTCAV project and our thanks goes to all: Dan Van Winkle and the TID AR RF Department, Erik Jongewaard, John Eichner and the TID RFAR Department, Jim Lewandowski and the AD FACET and Test Facility Department, Dave Steele and AD ICD ACC Power System group, LCLS Operations team and AD ICD Controls Department Software and Networking team members for their valuable contributions during the development of the project and for their continued support in making it a very successful one.

REFERENCES

[1] “New Tool to Measure X-ray Pulses Borrows from SLAC History”, SLAC Today News, June 25, 2013.

[2] P.Krejcik et al., “Engineering design of the new LCLS X-BAND Transverse Deflecting Cavity”, Proceedings of IBIC2013, Oxford, UK

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

FAST WIRE SCANNER UPGRADE FOR LCLS

J. M. D'Ewart#, M. Campell, P. Krejcik, H. Loos, K. Luchini, SLAC, Menlo Park, CA 94025, USA

Abstract

Wire scanners are a main diagnostic tool for transverse beam size and emittance measurements at LCLS. The original SLAC wire scanners were not optimized for speed (taking minutes to scan), and can't perform at the desired level of position resolution necessary for measuring LCLS' small beam size. A new fast wire scanner, based on a dc linear servo motor, has been designed and installed in the LCLS. The new fast wire scanner has several advantages over the original wire scanner: scan times are reduced from minutes to seconds while minimizing wire vibrations. Rather than counting open-loop step pulses, the new fast wire scanner uses real time position capture for beam synchronous sampling of the wire position, enhancing beam profile accuracy.

INTRODUCTION

The primary purpose of the wire scanner system is to provide a high resolution measurement of electron beam profile averaged over many shots. Profile measurements can also be used to determine transverse beam emittance.

The LCLS wire scanner system had been based on a drive system consisting of a stepper motor and a ball screw [1]. A 10X gear reducer was added to the system reducing wire vibration, but also reduced maximum wire speed [2]. A linear variable differential transformer (LVDT) was used to calibrate the drive system. The motion was operated in open-loop with no feedback about actual drive or wire position.

The design goals of the fast wire scanner are: reduce scan time from minutes to seconds, minimize vibration during the scan, improve position resolution of the wire scanner motion to 1μm or better, incorporate real-time readback of the wire position through the data acquisition system, and to integrate the wire scanner into the LCLS EPICS control system [1].

The advent of the LCLS-II project at SLAC has also resulted in the need for a wire scanner that can move at speeds sufficient to prevent wire destruction during CW beam operation.

For the nominal LCLS-II beam parameters of 100 pC charge per bunch, a transverse emittance of 0.5 μm, and a bunch repetition rate of 0.6 MHz the minimum wire speed to avoid damage is calculated to be 0.34 ms^{-1}.

FAST WIRE SCANNER

The fast wire scanner, shown in Fig. 1, uses an external actuator for linear motion. The drive system is based on the LinMot dc linear motor [3]. The wires are mounted on an interchangeable card that holds an x, y and u wire at 45° to the beam. A scan of all 3 wires can be done in a single motion of the scanner before returning to the home position. An increment linear encoder is attached to the drive system and limit switches are used at the end of travel. The linear motor eliminates the discrete steps of a stepper motor and the linear encoder allows for closed-loop servo control.

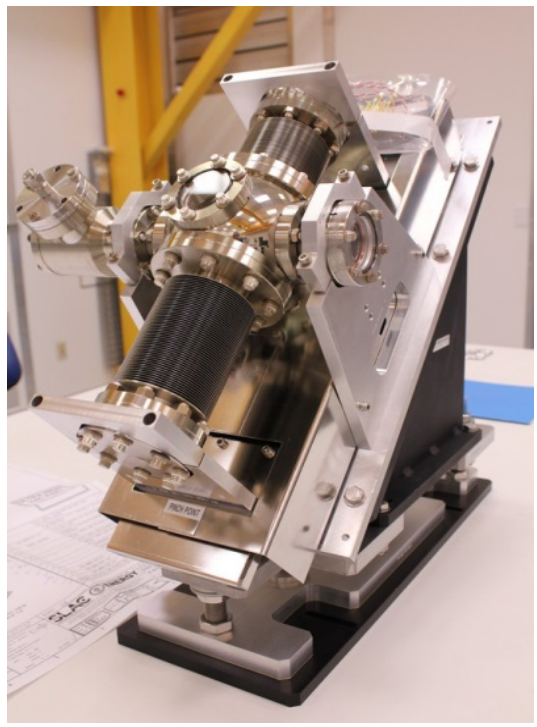


Figure 1: Wire scanner mounted at 45°.

CONTROL SYSTEM

Fast Wire Scanner Control System Architecture

The fast wire scanner control system architecture is based on the original LCLS VME wire scanner architecture. The original LCLS wire scanners use a VME crate with a MVME 6100, MRF EVR, CAEN V965 QDC board, Isig high voltage power supply module, and Hytec IP8601 stepper motor controller. The fast wire scanner uses an additional Prodex MAXv VME motion controller for servo motion control. A MAXv breakout chassis and LinMot panel are also installed into the control rack to support the fast wire scanner. The MAXv breakout chassis separates each of the 8 axis control and I/O signals and interfaces to the LinMot panel. The LinMot panel houses the servo drive as well as drive and logic power supplies. This architecture allows a rolling upgrade for current LCLS wire scanners. A block

*Work supported by the U.S. Department of Energy under contract number DE-AC02-76SF00515
#mdewart@SLAC.Stanford.EDU

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

diagram of the system is presented in Fig. 2.

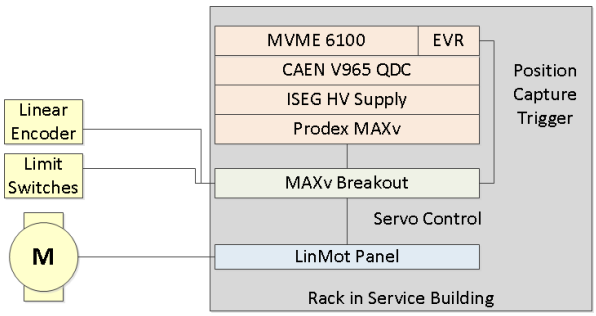


Figure 2: Fast Wire Scanner Control System Architecture.

Motion Control

The motion control system consists of an EPICS IOC (MVME 6100 running RTEMS), a Prodex MAXv 8000 VME motion controller [4], and a LinMot B1100 servo drive. The MAXv provides control for up to 8 servo or stepper axes. In servo mode the MAXv provides PID servo control with a 122 µs update rate. The MAXv is capable of custom, parabolic, S-curve and linear trajectory profiles. The MAXv servo signal is connected to the LinMot drive panel through the MAXv breakout chassis.

The LinMot panel, shown in Fig. 3 and Fig. 4, consists of a LinMot B1100 servo drive as well as a 72 VDC and 24 VDC power supplies. A Beckhoff BK9000, along with terminals KL1408 and KL2408, is used for digital I/O to interface with the LinMot B1100 servo drive. The digital outputs enable the drive and the digital inputs monitor status. The LinMot panel interfaces to the LinMot linear motor power and feedback signals and the MAXv control signals.

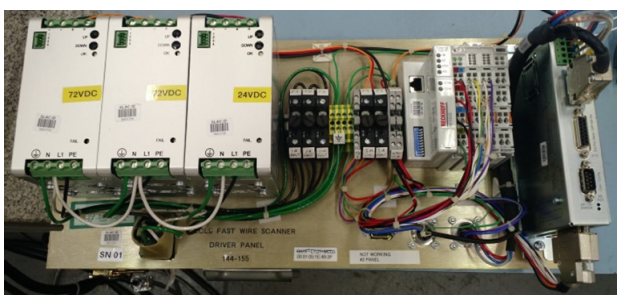


Figure 3: LinMot Panel, Front View.

Figure 4: LinMot Panel, Rear View.

Encoder Processing

An incremental quadrature linear encoder is attached to the linear slide of the LinMot drive system. The linear encoder provides 1 µm position resolution. The fast wire scanner system uses the real-time position capture feature of the MAXv. An EVR trigger is used to trigger a beam synchronous position capture event. The captured position data includes the axis, encoder position, positive and negative edge I/O bits, and home events [2]. The captured data is put in a ring buffer in VME shared memory and processed by the EPICS IOC. The position data is then processed by the beam synchronous acquisition (BSA) facility.

Homing Procedure

Homing is performed to ensure a consistent reference position subsequent to each power-up. The 45° install angle means to motor rests on the lower hard stop when no power is applied. The homing procedure is then as follows:

1. Power on motor
2. Move just off of hard stop
3. Move toward negative hard stop
4. Detect hard stop based on demanded torque and position error
5. Set known position on hard stop

Beam Synchronous Position Capture

The fast wire scanner uses the real-time position capture feature of the Prodex MAXv. An EVR trigger is used to trigger a beam synchronous position capture event. The captured position data includes the axis, encoder position, positive and negative edge I/O bits, and home events [4]. The captured data is put in a ring buffer in VME shared memory and processed by the EPICS IOC. The synchronously acquired position data is processed by a high level Matlab application to calculate beam profiles and transverse emittance.

Scan Trajectory

A high level Matlab application is used to initiate a scan. The wire scan application selects which of the three wires to scan and the appropriate scan speed based on beam rate. The wire scanner moves at max speed between wires and at the appropriate scan speed ± 2 mm of the nominal wire position. A full S-curve trajectory is used for the motion. An S-curve trajectory allows for smoother wire trajectory by setting jerk, the derivative of acceleration, constant. An ideal wire scan trajectory, covering all three wires, is shown in Fig. 5. A snapshot of the high level graphical user interface is shown in Fig. 6, along with x, y and u scan data.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

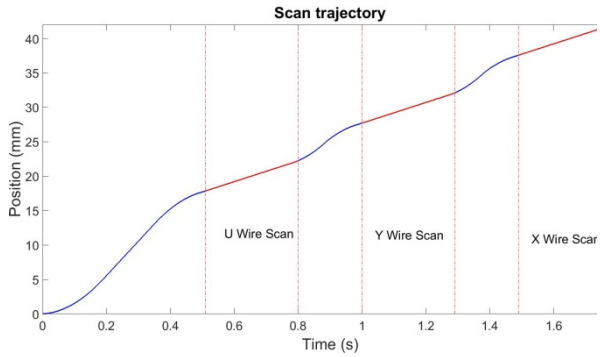


Figure 5: Ideal wire scan trajectory.

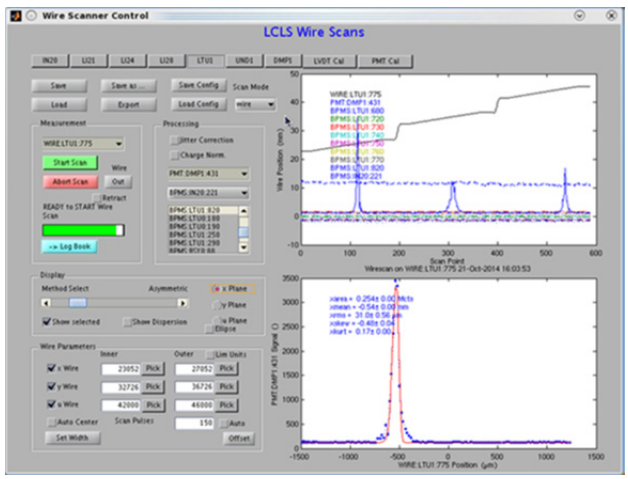


Figure 6: Wire scan graphical user interface showing x, y and u wire scan signals as well as motor position (top right) and individual wire scan profile (bottom right).

CONCLUSION

We have designed a new fast wire scanner based on a LinMot dc linear motor. The control system is designed to be backward compatible and allow for rolling upgrade of the LCLS wire scanners. Real-time position capture is used to beam synchronously acquire wire position. Three new fast wire scanners have been installed in the LCLS to date. The remaining slow wire scanners will be upgraded in a rolling fashion throughout the LCLS.

REFERENCES

[1] M.C. Ross, et. Al, "Wire Scanners for Beam Size and Emittance Measurement at the SLC", Proceedings of the 1991 IEEE Particle Accelerator Conference, p. 1201, May 1991.

[2] J. Frisch et al. "Beam Measurements at LCLS", SLAC-PUB-15018.

[3] P. Krejcik et al, "Evaluation of New Fast Wire Scanner Designs for the LCLS", BIW2012, Newport News VA, USA.

[4] http://www.omsinmotion.com/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

IMPROVING THE COMPACT MUON SOLENOID ELECTROMAGNETIC CALORIMETER CONTROL AND SAFETY SYSTEMS FOR THE LARGE HADRON COLLIDER RUN 2

D. Di Calafiori, G. Dissertori, L. Djambazov, O. Holme, W. Lustermann, ETH Zurich, Switzerland
P. Adzic, P. Cirkovic, D. Jovanovic, University of Belgrade, Serbia
S. Zelepoukine, University of Wisconsin-Madison, USA

Abstract

The first long shutdown of the Large Hadron Collider (LS1, 2013-2015) provided an opportunity for significant upgrades of the detector control and safety systems of the CMS Electromagnetic Calorimeter. A thorough evaluation was undertaken, building upon experience acquired during several years of detector operations. Substantial improvements were made to the monitoring systems in order to extend readout ranges and provide improved monitoring precision and data reliability. Additional remotely controlled hardware devices and automatic software routines were implemented to optimize the detector recovery time in the case of failures. The safety system was prepared in order to guarantee full support for both commercial off-the-shelf and custom hardware components throughout the next accelerator running period. The software applications were modified to operate on redundant host servers, to fulfil new requirements of the experiment. User interface extensions were also added to provide a more complete overview of the control system. This paper summarises the motivation, implementation and validation of the major improvements made to the hardware and software components during the LS1 and the early data-taking period of LHC Run 2.

INTRODUCTION

The Compact Muon Solenoid (CMS) [1] is a general-purpose particle physics detector built for the Large Hadron Collider (LHC) [2] at CERN. The CMS detector is composed of several types of sub-detectors: trackers, calorimeters and muon chambers. Based on the technical specifications and requirements of each sub-detector, custom Detector Control Systems (DCS) were designed, implemented and commissioned. A central CMS DCS [3] infrastructure provides general and common services, as well as the unification of controls and monitoring of all sub-detectors. A central CMS Detector Safety System (DSS) [1] is responsible for all off-detector safety-related matters. In addition, individual sub-detector safety systems are also available to provide extensions to the CMS DSS to guarantee the on-detector safety and integrity. The experience acquired during the first LHC long run (March/2010 – February/2013) ensured a better understanding of the detector operation and consequently the identification of possible fields for improvements to the DCS/DSS software and hardware components. The first long shutdown (February/2013 – June/2015) of the accelerator provided an opportunity for the implementation of most of these improvements, systems extensions and the preparation of even more robust, efficient and reliable systems for the LHC Run 2. This paper discusses the main improvements made to both DCS and DSS of the CMS Electromagnetic Calorimeter (ECAL) [1] throughout the periods described above. Focus is given to hardware upgrades and system consolidation.

Full details of the architecture of the CMS ECAL DCS have been reported previously [4,5].

CMS ECAL DCS OVERVIEW

This section presents a general description of the CMS ECAL DCS software and hardware. A simplified block diagram of the complete system is illustrated in Fig. 1.

Software

The CMS ECAL DCS application has been developed with a Supervisory Control and Data Acquisition (SCADA) software package named SIMATIC WinCC OA from SIEMENS, formerly Process visualization and control system II (PVSS II) from ETM. To facilitate the development process, the JCOP Framework [6], developed by CERN to provide extensions to WinCC OA for the high-energy physics domain, is extensively used. Through a Finite State Machine (FSM), manual and automatic commands are issued to lower nodes and node states are propagated and summarised upwards. Furthermore, the main control and state monitoring of all sub-detectors are accessible to the CMS DCS via the FSM to enable global CMS control.

The partitioning of the control nodes follows the sub-detectors architecture, in this case: CMS ECAL Barrel (EB), CMS ECAL Endcaps (EE) and CMS ECAL Preshower (ES).

Hardware

The DCS hardware comprises all devices used for connection to the low voltage and bias voltage powering systems, the electronics for environment monitoring and for additional support services, such as remote crate power cycle units and bias voltage current measurements.

Due to software changes at the CMS DCS level, presented in the section "*Controls Software Redundancy*", all field buses are being converted to Ethernet, allowing their physical decoupling from the DCS servers.

Spare parts and, in some cases, complete spare units are available to provide full and optimal support in case of hardware failures.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

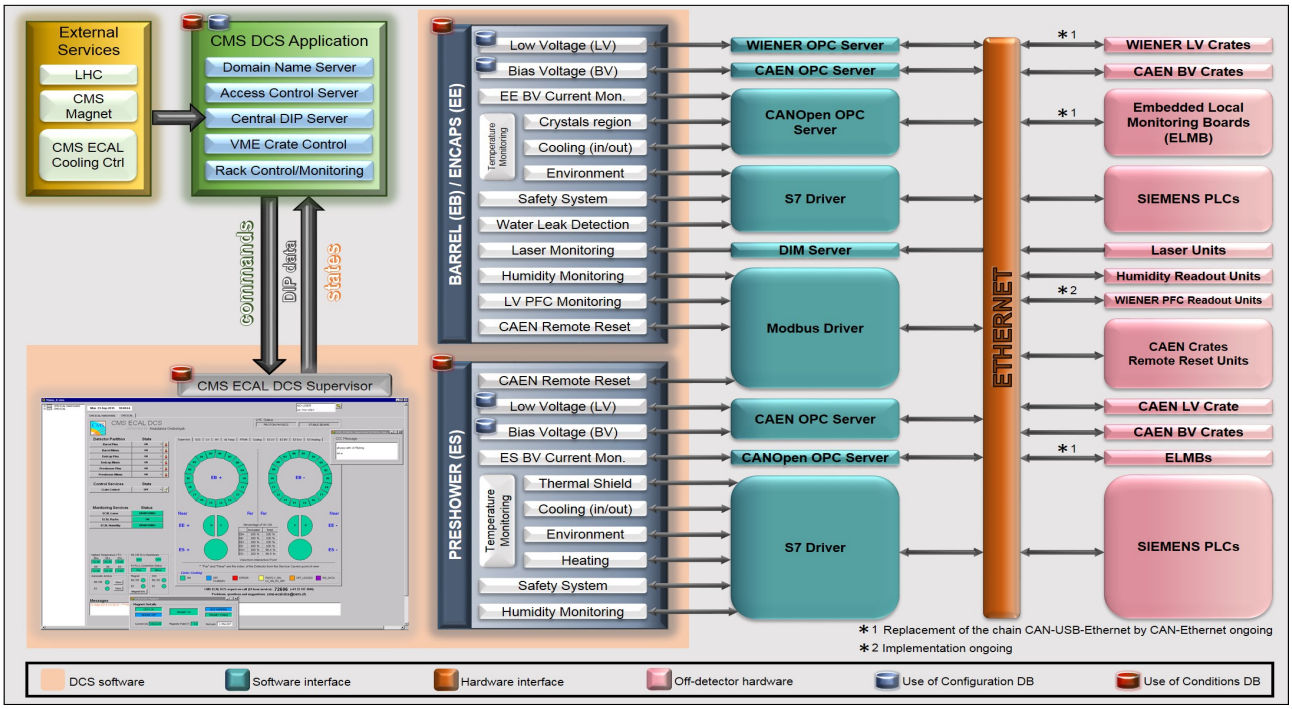


Figure 1: Simplified block diagram of the CMS ECAL Control System.

CMS ECAL DSS OVERVIEW

Designed to ensure the calorimeter safety and integrity, the CMS ECAL DSS [4,5] is based on a fully redundant SIEMENS Programmable Logic Controller (PLC) system with hardwired connections and interlocks to all relevant systems to ensure real-time actions in case of harmful conditions. Furthermore, it provides complementary resources to the main CMS DSS by monitoring parameters such as the cooling flow status and water leak detection for the low voltage power supplies racks.

UPGRADES AND IMPROVEMENTS

Controls Software Redundancy

Following a requirement from the central CMS DCS for running the complete control system in a redundant hot standby mode, with one set of servers installed in the experiment service cavern and another in the surface data centre, all sub-detectors had to confirm compatibility of their control system applications with this new operating environment. The preparation consisted of changes to the SCADA software and the adaptation of all hardware connections to Ethernet. The latter will be discussed in the next section "*Hardware Interfaces*".

In line with the implementation of software redundancy, a complete project for merging the CMS ECAL DCS sub-applications to run in fewer servers was carried out. Applications were initially combined based on their role and in a second step grouped according to their software interfaces to the hardware. As an example, all bias voltage applications were re-designed to run side-by-side, and then grouped with the Preshower low voltage application that shares the same CAEN Object Linking and Embedding for Process Control Data Access (OPC DA) Server.

The reduction from fourteen to three servers was a fundamental step towards the redundancy implementation. The software layer for the switchover mechanism was implemented by the CMS DCS group [7].

Hardware Interfaces

To allow the implementation of redundancy at the software level, the DCS servers had to be physically decoupled from the hardware. The Ethernet layer was chosen to ease the logical switchover between the two identical, but geographically separated, sets of servers. No changes were required for the connections to the PLCs and to the Arduino-based devices, which were originally Ethernet-based. To realise the connection to the EB/EE humidity readout units, commercial Modbus RTU-Ethernet adapters were selected and installed. Several approaches were evaluated for conversion of Controlled Area Network (CAN) buses. As no commercial CAN-Ethernet device was available at the beginning of this project, a chain of CAN-USB-Ethernet was tested. This solution worked for buses with few devices and/or low data traffic, but issues were identified with large, heterogeneous CAN hardware installations running in parallel on a single server. Furthermore, the latency added by this implementation triggers a significant number of false positive hardware disconnection timeouts.

In 2014, a commercial multi-port CAN-Ethernet adapter that fulfils the system requirements became available. Together with the necessary changes to the WIENER OPC server, the Triplicated AnaGate CAN Quattro was exhaustively tested and certified for installation at the end of 2015 [8].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Remote Reset of CAEN Mainframes

A complete analysis of all expert interventions during the LHC Run 1 enabled the identification of the most common failures and sources of the experiment downtime related to off-detector hardware. In particular, two cases were identified related to the CAEN mainframes that provide the bias voltage for the whole CMS ECAL and the low voltage for the Preshower. The first, the fact that after a power cut, some mainframes were not properly initialized and required a human intervention to perform a full reset. The second, occasionally, the communication with the mainframe was interrupted and only restored by a reset of the mainframe Central Processing Unit (CPU) controller. The minimum time of thirty minutes to identify the problem, access the experiment cavern and reset the faulty controller motivated the implementation of a remote reset unit, accessible through the main CMS ECAL DCS supervision panels.

As no commercial devices were suitable for an easy integration to the controls software, microcontroller-based units were developed. Each unit features fourteen ports and an implementation of Modbus over TCP on an Arduino Ethernet [9] for communication with the native Modbus driver from WinCC OA. The unit can issue Transistor-Transistor Logic (TTL) pulses with lengths of 200 ms for resetting only the CPU controller and 1000 ms for a full mainframe reset. A user-friendly panel allows the detector experts to trigger both actions in a few minutes for any of the twenty-three existing mainframes.

WIENER PFC Monitoring System

Another outcome from the analysis of the CMS ECAL DCS expert interventions was the need of an extension for the EB/EE low voltage application to read out the WIENER Power Factor Corrector (PFC) parameters. The failure of these devices was one of the most common hardware issues since the commissioning of the CMS detector. The monitoring of internal parameters might provide an opportunity to detect issues and schedule interventions prior to a total failure of the device during the detector operation.

To profit from the existing Modbus driver of WinCC OA, the readout Printed Circuit Boards (PCB) feature Modbus over TCP implemented with an Arduino Yún, which controls four ADG726 (dual 16-port multiplexer) to switch the RX and TX lines for up to 64 devices. In addition, the 5V and 9V required for the PFC serial circuit are permanently provided.

Humidity Monitoring System

The original design of the EB/EE Humidity Monitoring (HM) [4,5] system did not take into account the lengthy cables between the probes inside the detector and the readout hardware, which for some probes reaches more than 100 meters. The additional capacitance introduced by the cables and seen by the humidity transmitters imposed a limitation on the humidity readout range to 60 – 80%. To overcome this issue, a custom readout unit featuring very low frequency (1 Hz) transmitters was designed. To evaluate any possible degradation of the probes due to their operation out of the manufacturer specifications, tests setups were built at CERN and in Belgrade. After more than a year of evaluation, the new hardware was certified for installation in the experiment.

A calibration procedure was developed and each individual readout channel was calibrated using the same type of probe from the experiment, inserted in a controlled humidity unit with a very precise humidity sensor as reference. The new readout units were installed in the experiment and are able to measure from 10 – 80% relative humidity.

PTM Readout – Improved Power Distribution

The Precision Temperature Monitoring (PTM) is an ELMB [10] based system responsible for the temperature monitoring of the EB/EE crystals region and cooling. In the original design, three 12V power supplies, one for the analog, one for the digital and one for the CAN part, powered all ELMBs in parallel. An event where a single failure of one ELMB degraded operation of the whole monitoring system motivated the upgrade and improvement of granularity of the 12V power distribution.

Two sets of 3x12V supplies were installed, one for each half of the detector. A terminal block with fuse was installed for each power line at the experiment service cavern, allowing the disconnection of a possibly faulty ELMB without compromising the complete system.

Preshower Bias Voltage Monitoring

The Preshower Bias Voltage (BV) is distributed through a fully flexible matrix patch panel [11], where any input channel can be connected to any detector element or group of elements. For each individual line, a resistor bridge containing a reference resistor and branches to scale down the voltages are available to measure the current delivered to each output channel.

The period between the patch panel production and their installation in the experiment was insufficient for a complete evaluation of the readout and consequently for the definition of a proper calibration procedure. These tasks were performed later with a spare patch panel.

All readout channels were calibrated for current measurements with precision better than 2%. A complete and successful validation was carried out with several values of current from 10 µA to 2000 µA.

Due to a configuration featuring multiple grounds without proper isolation between them, a condition that was not previously tested, the results at CMS were not as expected. The patch panel that was removed from the experiment is currently being analysed.

Safety System Preparation for the LHC Run 2

A complete re-evaluation of the safety system PLC code and hardware was carried out to not only identify possible improvements but also issues related to ensuring long term support.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- Replacement of PLC CPUs

The CPUs, which were operational throughout the LHC Run 1, reached the end of their life cycle in July 2015 and were discontinued by SIEMENS: no more spare parts, repair service or technical support. Both CPUs were replaced by a newer model, with full support guaranteed until October 2022. Spare parts for the complete PLC system are available 24/7 from the CERN PLC Spare Parts Critical Stock;

- Production of spare readout units

The non-commercial hardware comprises the units reading out the EB/EE water leakage and temperature sensors and the units handling the interlock signals. To ensure a number of spares representing a minimum of 33% of the complete system, four spare readout units were produced;

- Recovery mechanism for CP341 failures

The communication between the SIEMENS CP341 modules and the readout units described above is realised through a RS-485 bus with a custom protocol. Due to external conditions, such as electrical noise, associated with extremely sensitive conditions to trigger safety actions, communication issues have triggered unwanted detector shutdown events. To restore the communication to the readout units, a human intervention to power cycle the CP341 modules was required. A mechanism to recover communication without the need of a power cycle was implemented in the PLC code and is under evaluation. To avoid any impact on the system reliability, it currently can only be manually triggered. Once this mechanism is certified, an automatic recovery procedure will be implemented.

24/7 OPERATOR AND EXPERT SUPPORT

Since the commissioning phase of the experiment, the CMS ECAL DCS has maintained 24/7 support services to ensure the maximum availability of the calorimeter. Initially, three of the main developers of the system were responsible for the daily operation of the detector and expert support in case of failures. Due to the significant workload involved, the support scheme was reviewed and divided in two roles, operator and expert. The operator on-call can be any person from the experiment collaboration and, with some basic training, is able to handle the detector daily operations and basic failures, while the expert on-call provides the most advanced level of support, either to the operator or directly to other CMS ECAL experts for more complex failures or interventions. This solution has proven to be very efficient and results in more balanced workload and increases the pool of people who have operational knowledge of the CMS ECAL DCS.

A log of all system failures, related to both hardware and software, has been maintained by the operators and experts to allow further analysis and consequent improvements based on preventive actions. Furthermore, based on feedback from operators and shifters, the available monitoring resources and software tools are periodically revised and, if required, extended to improve the system operation and troubleshooting.

CONCLUSION

The CMS ECAL DCS and DSS successfully supported the detector operation during the LHC Run 1. By logging the CMS shifters and DCS on-call operators and experts activities for over three years, several fields for improvements were detected. The LHC LS1 provided an optimal opportunity for these upgrades and system extensions. Efforts were made to ease operations, by providing additional troubleshooting tools and extended user-friendly operating panels. Most of the changes discussed in this paper have already been deployed and an excellent support to the CMS ECAL is being provided during the LHC Run 2.

ACKNOWLEDGMENT

The authors would like to thank the Swiss National Science Foundation and the Ministry of Education, Science and Technological Development of Serbia for the financial support.

REFERENCES

[1] CMS collaboration, S. Chatrchyan et al., “The CMS experiment and the CERN LHC”, JINST 3 S08004 (2008).

[2] Lyndon Evans and Philip Bryant, “The CERN Large Hadron Collider: Accelerator and Experiments”, JINST 3 S08001 (2008).

[3] Robert Gomez-Reino et al., “The Compact Muon Solenoid Detector Control System”, ICALEPCS’2009, Kobe, Japan (2009).

[4] D. Di Calafiori et al., “The CMS Electromagnetic Calorimeter Detector Control System”, CHEP’2010, Taipei, Taiwan (2010).

[5] D. Di Calafiori et al., “The CMS ECAL Detector Control System”, ICALEPCS’2009, Kobe, Japan (2009)

[6] O. Holme et al., “The JCOP Framework”, ICALEPCS’2005, Geneva, Switzerland, (2005).

[7] L. Masetti et al., “Increasing Availability by Implementing Software Redundancy in the CMS Detector Control System”, ICALEPCS’2015, Melbourne, Australia (2015).

[8] G. Thomas et al., “CAN Over Ethernet Gateways: A Convenient and Flexible Solution to Access Low Level Control Devices”, ICALEPCS’2015, Melbourne, Australia (2015).

[9] ARDUINO website: https://www.arduino.cc/en/Main/ArduinoBoardEthernet

[10] H. Boterenbrood and B. Hallgran, “The Development of the Embedded Local Monitor Board (ELMB)”, 9th Workshop on Electronics for LHC Experiments, Amsterdam, Netherlands, (2003).

[11] P. Vichoudis et al., “The upgraded CMS Preshower high voltage system”, JINST 8 C01050 (2013)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

EXPERIENCES AND LESSONS LEARNED IN TRANSITIONING BEAMLINE FRONT-ENDS FROM VMEBUS TO MODULAR DISTRIBUTED I/O

I. J. Gillingham, T. Friedrich, S. C. Lay, R. Mercado,
Diamond Light Source, Oxfordshire, UK

Abstract

Historically Diamond's photon front-ends have adopted control systems based on the VMEbus platform.

With increasing pressure towards improved system versatility, space constraints and the issues of long term support for the VME platform, a programme of migration to distributed remote I/O control systems was undertaken.

This paper reports on the design strategies, benefits and issues addressed since the new design has been operational.

INTRODUCTION

Diamond Light Source is a 3 GeV third-generation light source with a 561m 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 28 photon beamlines, with a further two beamlines in an advanced stages of design and construction.

In 2010, Diamond adopted a new control system design for future beamlines, adopting fieldbus distributed control as a viable alternative to the long standing VMEbus architecture [2]. In planning for the present phase of photon beamlines, in 2015, the control system architecture and choices of hardware used was revisited.

PREVIOUS CONTROL SYSTEM ARCHITECTURES

The original accelerator and beamline control systems at Diamond are based on 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) were used. The field signals are interfaced via either rear transition modules or front-panel connections. A VME microprocessor (MVME5500) runs VxWorks and EPICS to serve up the control information to client applications.

REASON FOR CHANGE

The original control system architecture served well for the existing accelerators and beamlines; however it was defined a number of years ago, so in the context of the recent and future phase of beamlines the opportunity to reconsider the standards was taken. In doing so, it is clear that not all the hardware capability of VME was required for beamline control; neither was the use of a hard real-time operating system such as VxWorks. It was also apparent that most I/O functionality required for control of beamline and accelerator 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 beamlines, the following technical systems were identified:

- Motion control
- Vacuum instrumentation and other serial devices
- Video cameras
- Analogue and digital signals
- Programmable logic controllers
- Timing signals

The disturbance to beamline operation through restarting an IOC, should be minimised.

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 or Storage Ring (SR) vault and by the space available in the some beamline hutches.

NEW CONTROL SYSTEM ARCHITECTURE

Each EPICS IOC runs on a 1U Linux PC located within the relevant Controls Instrumentation Area (CIA). The Linux servers have several physically separate network connections to support the different systems.

Linux Server

The IOCs are standard Dell 1U machines (currently R320), without a hard-disk installed, that boot from an image over the network into a RAM based disk.

The current operating system is 64 bit RedHat Enterprise Linux 6, which has real-time extensions compiled in to facilitate determinacy for the EtherCAT driver, when used at high bandwidth.

On boot, the server picks up its host-name and IP address via DHCP, determined by the MAC address of eth0. The DHCP server also points to the TFTP server which provides the kernel and ramdisk image used to boot the machine.

A startup-script in the host's soft IOC directory is used to configure the server. For instance, network interface

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

eth1 could be assigned an IP address, the kernel-module for the event-receiver card could be installed and the EtherCAT service started. This gives users the ability to configure the machine using a version controlled script.

Remote I/O Interfaces

In order to manage various classes of signal associated with control and acquisition, two distinct types of interface have been adopted, PLC Remote I/O (PLC RIO) and EtherCAT. The following table lists and summarises the attributes of each type of interface:

Table 1: RIO Interface Types

Interface	Attribute	Description
PLC RIO	Signal processing	Ladder logic in PLC generates machine protection interlock permits, in rapid and predictable response to input signals
	Interface with IOC	Ethernet or serial communications with slow bandwidth in the order of 1Hz. Sharing of field signals between the PLC and IOC is facilitated by mapping in PLC memory and reading it periodically from a client (e.g. IOC) using the FINS protocol.
EtherCAT	Signal processing	Various signal categories are supported for devices not necessarily associated with machine protection, i.e. for monitoring only. Analogue signals may be sampled at rates exceeding 10kHz.
	Interface with IOC	The EtherCAT network is scanned by a dedicated service on the IOC's host Linux server and readings made available to one or more IOCs via network sockets.. This can take advantage of preemtive real-time kernel extensions when high bandwidth, determinate signal processing is required.

NETWORKS

The servers are supplied with four network interfaces. This is to facilitate multiple dedicated and isolated networks, typically three are used as:

- Primary control network
- Instrumentation network (private to the server and network connected peripherals)
- EtherCAT network

The context of key network components is shown in Fig 1.

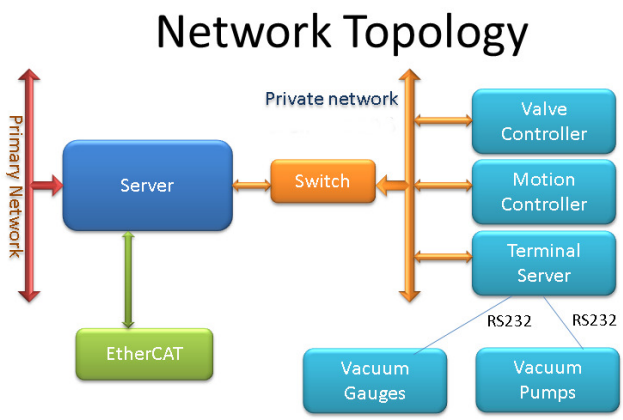


Figure 1: Network topology.

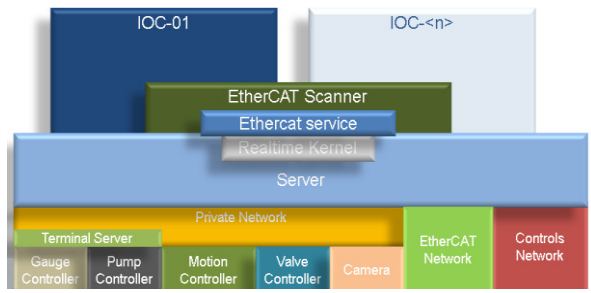


Figure 2: Logical Model.

Instrumentation Network

A number of instrument controllers are now equipped with network interfaces to facilitate control and data acquisition. It is desirable in most instances to secure these on a dedicated network, which is private to the Linux server. This avoids unnecessary traffic on the Primary Controls network and shields the instrumentation from connection outside the scope of the control system dedicated to that area.

EtherCAT Network

Where remote I/O (RIO) is required and not available on a RIO PLC, EtherCAT modules have been employed. The EtherCAT master module Linux may be used seamlessly with either Linux kernels with PREEMPT-RT enabled, or for standard development and "soft" test systems without the real-time extensions. For applications which require greater determinacy, such as high frequency analogue signal sampling, the pre-emptive real-time kernel extensions must be used.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Tests were performed to measure the effect of jitter on the master process [3]. The kernel patch pre-emptive real-time (PREEMPT_RT) reduces latency and adds the ability to pre-empt most kernel critical sections. However the user-space APIs are unchanged as the necessary calls are already present as part of the POSIX ADVANCED REALTIME standard [4], so that applications can be developed and tested on a standard system. For this application Diamond is using kernel 3.6.11.2-rt33 from MRG.

The following API features are used in the EtherCAT scanner:

- high-precision timers using clock_nanosleep
- SCHED_FIFO pthread scheduling
- PTHREAD_PRIO_INHERIT mutexes

The use of mlockall is mentioned in a previous publication [3] but has since been removed from the code. No adverse effect has been noticed.

Improvements have been realised by:

- Adopting a scheme of using a slave serial number as identifier to allow reconfiguration of the devices.
- Patching the Etherlab library to stop syslog messages overwhelming the system when a section of the bus is disconnected.
- Adding support to read and write Service Data Objects (SDOs) from the bus scanner application and from the higher level control system. The support for SDO came out of the need to support an LED controller, but has wider applications because SDOs are used to configure various EtherCAT Modules.

On a note on the hardware revisions of slaves, recent experience revealed that some PDO names vary between slaves' revisions in the EtherCAT Slave Information (ESI) files from the manufacturer. The scanner configuration files are generated from a description slave models and revision numbers. This information is then used to collect data from ESI files. The EtherCAT driver software propagates the slave information upstream as Asyn parameters. These name changes have prompted the need to revise the management of the slave information files from the manufacturer to screen for name changes before adopting the vendors' ESI files. The impact is limited to a subset of EtherCAT modules and solutions are being discussed to avoid revision specific version of templates. This simplifies hardware swaps in case of failure on an slave module.

The use of EtherCAT for acquisition and control of digital and analogue signals has been in use in beamlines and front ends built in the last four years. The devices have replaced VME signals and solutions have been developed for signal acquisition up to 10 kHz.

The separation of signal acquisition and the scanner publishing the bus signals over a socket, within a Linux server, allows for reusing a signal with different names in different EPICS IOCs. This feature, included in the original design, has proven useful when adding waveform acquisition from an ADC without the need to bring down the running system.

REMOTE I/O PLC

A new generation of Omron PLCs has been selected to enable the adoption of Remote I/O (RIO) and an Ethernet interface between the server and the PLC.

PLC RIO facilitates the reduction of cabling complexity and improving flexibility on deploying I/O into the field. PLC RIO uses standard Ethernet cabling and Profinet, and RIO modules are built up into the standard configurations, one for thermal monitoring and another for general purpose I/O. The Remote I/O does not perform any logic on any of the signals; it simply acts as a receiver and passes everything back to the PLC crate. This necessitates configuration of RIO output modules such that in the event of failure of RIO communication I/O fails to the safe state. The PLC logic is protected by a watchdog timer to protect against failures in communication with RIO.

The new PLC CPU (CJ2M) has a built-in Ethernet connection which allows both EPICS and the programming software to share the connection. EPICS communication is handled over UDP using FINS protocol whilst programming is realised over TCP/IP.

The remote I/O modules were originally prototyped into two standard configurations to enable spares to be maintained and allow faster recovery in the event of a problem. However the reality is that hot swap is not practical without interruption of the RIO communications bus and hence the PLC integrity, as a number of the modules have some intelligence which means that commands have to be sent to the card to configure them.

Problems that Needed Resolving

A significant issue, related to analogue modules, is that the values for temperatures freeze at the last known value when RIO communications is lost. Through the use of a watch-dog timer and detection of communications errors, the analogue is set to an error value.

Benefits of RIO

The biggest benefit of RIO is flexibility of design and the reduction in cabling. This has significantly reduced the number of connections to just those in interface boxes directly mounted out in the field. This has also significantly sanitised the wiring on beamlines. Often the beamline equipment is delivered with different sensors to those expected and we are able to adapt the system for this. The ability to add extra modules into the system as equipment arrives is very useful and suits the piece-meal commissioning of beamline equipment as more complex equipment is delivered.

SUMMARY OF PROGRESS TO DATE

The Diamond Control Systems now implement some 470 Linux based IOCs; a proportion of which utilise distributed I/O, such as EtherCAT. This design pattern has realised an integrated, versatile and maintainable control system, upon which enhancements and upgrades

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

can be confidently planned on platforms which provide inherent long term stability and support.

REFERENCES

[1] R. P. Walker, “Commissioning and Status of The Diamond Storage Ring”, APAC 2007, Indore, India.
[2] I. J. Gillingham, “Diamond’s transition from VME to modular distributed I/O”, PCaPAC 2010, Saskatoon, Saskatchewan, Canada.
[3] R. Mercado, I. J. Gillingham, J. H. Rowland and K. Wilkinson "Integrating EtherCAT based IO into EPICS at Diamond." ICALEPCS 2011, Grenoble 2011”
[4] IEEE and The Open Group “The Open Group Base Specification Issue 7” http://pubs.opengroup.org/onlinepubs/9699919799/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DETECTOR AND RUN CONTROL SYSTEMS FOR THE NA62 FIXED-TARGET EXPERIMENT AT CERN

Piotr Golonka, CERN, Geneva, Switzerland
Riccardo Fantechi, CERN, Geneva, Switzerland
Manuel Gonzalez-Berges, CERN, Geneva, Switzerland
Fernando Varela, CERN, Geneva, Switzerland
Valeri Falaleev, JINR, Dubna, Russia
Nicolas Lurkin, University of Birmingham, Birmingham, UK
Ryan Frank Page, University of Bristol, Bristol, UK

Abstract

The Detector and Run Control systems for the NA62 experiment, which started physics data-taking in the autumn of 2014, were designed, developed and deployed in collaboration between the Physics and Engineering Departments at CERN. Based on the commonly used control frameworks, they were developed with scarce manpower while meeting the challenge of extreme agility, evolving requirements, as well as integration of new types of hardware. The paper presents, for the first time, the architecture of these systems and discusses the challenges and experience in developing and maintaining them during the first months of operation.

INTRODUCTION

NA62 is a fixed-target experiment at CERN aiming at high-precision measurements of rare Kaon decays, currently taking physics data in its second run. Operation of the detector and its infrastructure is supervised with dedicated control systems, the most complex being the Detector Control System (DCS) and Run Control (RC). The two are similar in many aspects and share common infrastructure, as well as development and maintenance workflows. The Run Control has already been introduced in [1,2]. This paper focuses on the - often neglected - topic of complete architecture, including infrastructure, project organization and development life cycle.

The NA62 detector is composed of a set of sub-detectors developed and maintained by semi-independent groups. Some equipment is reused from previous experiments (such as the Liquid Krypton Calorimeter), while others must be built anew. Certain sub-detectors or their parts were not ready for integration before the first run, and they are added progressively In consequence, the development, commissioning and maintenance of control systems is a *process* that spans over years. This is unlike many turn-key control systems in industry. Assuring consistency and long term maintenance in the project, taking into account a high-turnover of people in experimental collaboration is one of the primary challenges.

REQUIREMENTS

The DCS and RC systems for NA62 resemble those of other CERN experiments. Relative to the LHC experiments they are an order of magnitude lower in number of devices, yet their complexity remains similar.

The systems have to allow for operation and monitoring of a variety of devices, react on value changes, evaluate alarm conditions and notify the operators about the alarms. The operators need to be presented with an alarm screen, as well as synoptic views that summarize the state of parts and subsystems of the experiment, and be able to drill down and quickly identify or locate the required device. They need to easily command individual devices, as well as groups of them (e.g. complete subdetectors)

The experts need to tune the set-points and alarm threshold, and bring up the trend plots of historical values. Reconfiguration should be applied to a selected device or using configuration sets (*recipes*) reflecting the run modes of the experiment. The experts must be able to detect and diagnose malfunctions and communication problems, and dynamically reconfigure the system, e.g. re-wire faulty sensors or mask alarms.

The history of acquired data (*conditions*) needs to be recorded to an Oracle database to be used in physics analysis. CERN computer security rules need to be applied. Robust infrastructure and hardware should be employed to guarantee round-the-clock operation during 8 months of data taking with minimal human supervision.

OPERATION

The independent operation of the DCS and RC is performed on dedicated consoles in the NA62 Control Room. The User Interface window requires authentication with CERN credentials, and enables access to shifter or expert operations, on various parts of detector, depending on privileges defined by administrators.

Standard shifter operation employs two tools: the alarm screen, displaying a time-sorted list of anomalous incidents ("alarms") and the hierarchical control supervision screen. The latter, also known as "FSM" (Finite State machine), abstracts out the hardware to present a tree view of the detector's parts and subsystems. Hardware states are evaluated and represented by colour-codes and text labels ("*ERROR*", "*RAMPING*"). These states are propagated in the upstream direction of the tree using summarization logic. This allows for rapid root-cause identification by expanding the coloured tree view. High level commands ("SWITCH_ON", "CONFIGURE") are broadcast down the tree allowing control of large parts

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

of the hierarchy with single button-presses. It is possible to exclude devices and sub-systems from central hierarchical operation and hand them over to an expert in a standalone or shared mode of operation. Whereas the shifters may only see the current values for set points, the experts can also modify them, as well as mask the alarms or change their thresholds. The principles of hierarchical operation, and the employed tools, are identical to those used by the LHC experiments.

Remote access and expert operation is also possible through dedicated Terminal Servers and secondary consoles which run the same "User Interface" application as the main control room console. Conflicting operations from two operators are avoided thanks to the concept of partitioned operation and sub-tree ownership.

PROJECT STRATEGY

Following the good experience with the LHC Detector Control Systems, it was proposed to develop and maintain the NA62 controls projects by a Central Team with assistance of experts from the CERN EN/ICE group. The Central Team, recruited from NA62 members, would work closely with sub-detector experts to implement their requirements; the experts would provide the necessary expertise, training as well as technical supervision of the projects. To maximize the use of scarce development manpower allocated for the projects, main developer(s) would work in proximity and under direct supervision of EN/ICE experts especially at the initial stage of project development.

A general policy was adopted to apply standard, centrally supported and recommended technologies and make use of existing services available at CERN. In addition, in order to enhance the stability of the running system, simplify long-term support and assure evolution of the project, guidelines and a development process with elements of quality-assurance and testing were agreed on.

ARCHITECTURE AND TECHNOLOGIES

The systems are developed with standard CERN set of technologies, and a layered architecture. Figure 1 presents the geographically scattered layout of the system, with the Supervision, Process/Frontend and Field/Equipment parts.

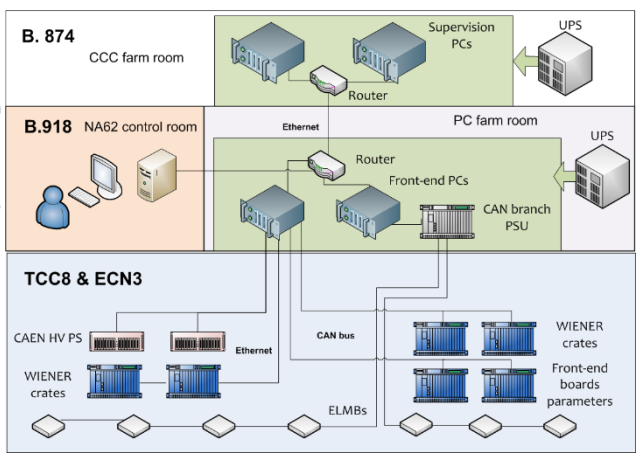


Figure 1: Architecture of the NA62 Control Systems with equipment location.

In the Supervision Layer there is a set of 10 supervision applications: one for the Run Control, one per sub-detector (or a set of smaller ones), and one Central DCS application that orchestrates the execution of sub-detector applications. The cores of SCADA applications are running on dedicated control servers (located in the server room which hosts other CERN control systems) and the User Interfaces (UIs) programs running on a set of consoles located in the the control room. The server room and the experiment facilities are connected by a direct optical fiber link.

The Frontend layer is composed of dedicated computers, hosting hardware-interface cards (for CAN bus). They aggregate the traffic from hardware, translate the necessary middleware technologies eliminating platform-dependent factors and interface with the Supervision layer over a dedicated experiment network. A few PLCs are used to interface with relays and sensors.

The Field and Equipment layer is populated by the high and low voltage systems (over 4000 channels), electronics crates, and I/O boards with sensors. A significant number of custom frontend-electronics boards are configured and monitored.

Technologies

The supervision layer is built with the standard CERN controls software stack based on the commercial *WinCC Open Architecture* SCADA [3], and the *JCOP* and *UNICOS* frameworks [4,5,6].

Communication with the hardware layer is mediated by CERN's standard DIP, DIM [7] and OPC-DA middleware protocols; Ethernet and CAN bus media are used to reach the field layer.

Standard hardware used by other CERN Detector Control Systems integrated within the JCOP Framework [4] are used for high voltage, electronic crates and analog and digital I/O.

The readout for the unique, legacy Liquid Krypton Calorimeter is interfaced through a genuine solution based on the mixture of control frameworks and technologies. A PLC powered by UNICOS is used to control relays, ELMB I/O boards integrated through JCOP Framework are used for voltage readouts, while the electronics monitoring is provided by a dedicated low-level software interfaced through DIM.

A particularly challenging task was to interface the control for the thresholds frontend electronic boards. Initially interfaced using the CanOpen technology (used also by ELMB boards), they suffered from stability and performance issues in large setups over long CAN bus lines for the LAV detector. Efforts by the developers of the custom firmware as well as low-level protocol debugging brought only a moderate improvement. An alternative control channel was then exploited: a set of RaspberryPi mini computers connected over USB to the frontend boards was programmed to expose a control and monitoring interface over DIM, which allowed for

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

integration into the Run Control. DIM was also used to interface non-standard high- and low-voltage system for the STRAW subdetector.

A notable achievement is the part of DCS for the KTAG subdetector that performs a "pressure scan": it commands the NA62 Gas Control System, through the DIP protocol (uni-directional version of DIM) to set the gas parameters in the subdetector as necessary, and then automates the task of sequencing the parameters which otherwise would need to be executed manually by the operator using the Gas Control System console.

Infrastructure and Services

To reduce the maintenance effort, NA62 control systems share infrastructure and use available CERN services. Even though additional coordination is needed, technical expertise of others, for a wide range of technologies, is employed.

Numerous services provided by the CERN IT Department were employed to construct the infrastructure. A dedicated NA62 experimental network was set up, isolating the traffic for controls and data acquisition from the CERN General Purpose Network. Inter-network communication (such as the Technical Network or GPN) was configured to make the CERN central services (disk servers, domain controllers, DNS, DHCP or databases) available as necessary while securing the access to the experiment's equipment. Remote access to this internal network is provided by the Windows Terminal Server service, run on the CERN virtualization infrastructure, fully maintained by the service providers. The same virtualization infrastructure is used to form the integration test setup. The Oracle database service provides performant and highly-scalable storage for historical process data as well as for configurations. All the services come with the possibility of adaptation or upscaling and offer test and validation setups as well as expert's help.

Infrastructure maintenance for the supervision layer (hardware, operating system software installations, security, monitoring and backups) is in the hands of the experts from the CERN BE/CO group. The NA62 servers and consoles are maintained together with around 200 other production control servers and over 500 operator consoles by a team of experts.

The software and hardware for the Frontend Layer are specific to the equipment it controls, and hence their maintenance need to be under the responsibility of NA62 experts. Nevertheless, the computers were purchased with the help of the BE and IT Departments, to profit from established contracts for the long-term availability of spare parts and on-site support. Support for the PLCs, as well as the WinCC OA service, middleware software and standard protocols (OPC Server software) are provided by the EN department and guarded by maintenance and support contracts between CERN and hardware vendors.

The frontend computers, consoles and data servers are powered through UPS units providing up to an hour of autonomy in case of a power-cut.

SYSTEM DEVELOPMENT LIFE CYCLE

Control systems for physics experiments evolve constantly: subsystems are upgraded, replaced or added, hardware and software faults need to be resolved, infrastructures change, new versions of software packages installed. To optimize new operation modes new features are requested throughout the lifetime of the experiment. On the other hand, uncertainties in hardware prototyping often lead to late delivery of hardware and drastic changes in specifications and requirements. Requirements for operation may only be specified on very high level, and they evolve according to the phase of the experiment and operation experience. Planning and development of control systems, in particular with highly constrained manpower became complex, and the classical system development life cycle (SDLC) [8] workflows were therefore not applicable to NA62 controls projects. The boundaries between the classical planning, analysis, design and implementation phases are blurred and overlap. However, we tried the best practices from these models and came up with a method that tries to match the resources, constraints and requirements.

At the heart of the methodology is the requirement of maintaining a running system, while being able to constantly evolve and enhance it and fix issues. For that, we apply a policy of packaging all development into so called components, which are released with version numbers - the concept underpinning the JCOP Framework [4]. Engineering tasks are done on development machines, and changes are saved in the Subversion software repository. Then new features sets and bug fixes are released as a new version of the component, and installed in the integration setup to perform acceptance tests. Finally, they may be applied to a production system, with a possibility of a rollback.

This workflow goes against the need for rapid debugging and bug-fixing directly on the production system, often desired for operation. The changes prototyped in production need to be transferred to the code repository and go through the whole component release and acceptance process. While working under high time pressure, it is tempting to abandon the workflow, and proceed with development only on the production system. This results in software regression bugs whenever a new version is deployed. Potential time saving needs to be over-compensated by increased efforts in debugging.

The timing and phases of the development cycle are driven by the physics data-taking schedule. The system is only partially used during experiment shutdown time (winter/spring) which makes it the perfect time for scheduling development and maintenance tasks. Prior to the startup of the run, new requirements and new tasks for integration typically start to appear, as new hardware is becoming available. During the data-taking, feedback is given by operators and experts, who request new ad-hoc features to be necessarily added.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

EXPERIENCE

The operation of the subsystems that are well advanced in their development, notably the Run Control and DCS for KTAG, GTK and LKr subdetectors was smooth. The availability of the infrastructure and services was very high, with smooth interventions and good coordination. Integrated operation of the detector through the Central DCS (rather than ad-hoc hardware control) has become regular, despite minor instabilities and rough edges. The priority areas where development efforts need to be intensified are the homogeneity and consistency of the hierarchical control (FSM tree) and consistent use of recipes stored in a configuration database. Experts demand more diagnostic tools for hardware communication and system integrity problems. Integrated with the supervision layer these should allow for better feed-back to operators. Overall stability of the, sometimes unstable, frontend software needs to be improved.

The component-based development method and integration tests worked well, even though it required significant initial effort from developers. Not only did it facilitate the integration of developments without affecting system stability, but also allowed for smooth migration to a newer version of SCADA software. Similarly, it makes the complete reinstallation of the system (in case of unlikely failure of hardware or infrastructure) possible within a short time.

A layer of NA62 common software components (e.g. widget libraries) allowed for new features to be easily deployed across all parts of the system. The use of standard hardware components supported by the JCOP Framework allowed for the rapid integration of significant parts of the system. For non-standard hardware, the use of DIM proved to be particularly effective, where applicable.

The development of certain systems benefited from having hardware that was at least partially installed before the physics run. This allowed the development chain to be completed in full before deploying on the production system and having the system operational in 2014.

We noticed that despite the fact that the core features and tools were planned and developed during the inter-run period, a significant amount of work was required during data taking. Many of the sub-system specific expert and diagnostic tools were requested as the need arose with the availability of new custom hardware. These needed to be implemented quickly to pursue the commissioning of the experiment.

Additional burden on development was induced by the non-standard, undocumented behaviour of hardware. Debugging and integration-testing needed to be interleaved with the development. Software that was initially modelled according to official documentation needed to be adapted (e.g. state transitions for the HV system).

We also stress the importance of the user requirement formulation process. Negligence, inaccuracies and "last-minute" changes led to important increase in development efforts or even the complete redesign of certain subsystems. Combined with lack of developers, this delayed the delivery and commissioning tasks.

The complexity of the systems makes the learning curve for new contributors very steep and requires an education phase of a few months. A number of developers were trained only to leave the project shortly after they delivered initial (often not yet complete) results. In effect, it was not possible until now to put in place the collaboration model of the strong central team made of experienced users.

Drawing proper balance between (often very ambitious) requirements and available developer resources was among the main challenges in the project management.

CONCLUSION

We presented the architecture, project management and development life cycle for the NA62 control systems. By maximizing the reuse of existing components, applying standard solutions and best practices and using the services made available by CERN groups we were able to deliver working control systems, and perform their maintenance with minimal effort and very scarce resources. The need to enhance and maintain control systems throughout their life time was stressed by gathered experience. Constant effort and resources need to be assured in order to maintain the robustness of the system and to implement requests coming from the operators. High agility in project management is required to accomplish this task.

REFERENCES

[1] F. Varela, *et al,* "Reusing the Knowledge from the LHC Experiments to Implement the NA62 Run Control", ICALEPCS 2013, San Francisco, USA

[2] N. Lurkin, "The NA62 Run Control", proceedings NSS/MIC Conference IEEE 2013, Seoul, Korea

[3] SIMATIC WinCC Open Architecture (previously PVSS) SCADA software from ETM (Siemens subsidiary), http://www.etm.at

[4] M. Gonzalez-Berges *et al*, "The Joint COntrols Project Framework", CHEP 2003, La Jolla, USA

[5] H. Milcent *et al*, "UNICOS: An open framework", ICALEPCS 2009, Kobe, Japan, 2009

[6] J. Arroyo Garcia *et al*, "Integrating Controls Frameworks: Control System for NA62 LAV Detector Test Beams", ICALEPCS 2011, Grenoble, France

[7] C. Gaspar *et al*, "DIM, a portable, light weight package for information publishing, data transfer and inter-process communication", Computer Physics Communications 140 (2001) 102-109

[8] "System Development Life Cycle" wikipedia article, https://en.wikipedia.org/wiki/Systems_development_life_cycle

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DATABASE ARCHIVING SYSTEM FOR SUPERVISION SYSTEMS AT CERN: A SUCCESSFUL UPGRADE STORY

P. Golonka, M. Gonzalez-Berges, J.Hofer, A. Voitier
CERN, Geneva, Switzerland

Abstract

Almost 200 controls applications, in domains like LHC magnet protection, cryogenics and vacuum systems, cooling-and-ventilation or electrical network supervision, have been developed and are currently maintained by the CERN Industrial Controls Group in close collaboration with several equipment groups. The supervision layer of these systems is based on the same technologies as 400 other systems running in the LHC Experiments (e.g. WinCC Open Architecture, Oracle). During the last two-year LHC Long Shutdown 1, the 200 systems have been successfully migrated from a file-based archiver to a centralized infrastructure based on Oracle databases. This migration has homogenized the archiving chain for all CERN systems, and at the same time has presented a number of additional challenges. The paper presents the design, the necessary optimizations and the migration process that allowed us to meet unprecedented data-archiving rates (unachievable for the previously used system), and liaise with the existing long-term storage system (LHC LoggingDB) to assure data-continuity.

INTRODUCTION

Storage and retrieval of historical process values and alarms is one of the invisible yet essential tasks of SCADA systems, and is always delegated to a dedicated *historian* or *archiver* tool. WinCC OA [1] offers three technologies for archiving: the file-based archiver ("valarch"), the simple database logging facility ("DBLogger", available as of version 3.13) and the high-performance, scalable Oracle database archiver ("RDB Archiver").

Even though a number of large control systems (Detector Control Systems for the LHC experiments) made use of the RDB archiver since the beginning, the control systems for infrastructure and LHC services initially chose to use the file archiver. The divergence in the choice of archiving technology stems from the requirements. The detector control system had to have the historical values available as the so called "conditions" in an Oracle database to make the data analysis possible, and it was this requirement that drove the collaboration between CERN and ETM to develop and optimize the database archiver. On the other hand, the priority for the LHC and infrastructure systems was the use of a stable, proven technology to maximize the reliable operation of the systems.

With an increasing number of control systems to deliver and maintain, and the need to scale-up applications to meet the challenges of Run II of the LHC, it became clear that use of the file-based archiver reached its limit, and a migration to the more performant and centralized archiving architecture became a necessity. In addition, the centralized management of archived data offers several advantages such as better tools for data analysis, simplified management, easier handling of the SCADA server nodes, etc.

In the paper we present the process of migrating the 200 applications of the SCADA Application Service from file-based archiver to Oracle database archiver: the initial requirements, thorough testing and optimisation phase and proof of scalability, redesign of the long-term data storage architecture, actual migration effort. We will then summarize the current state of this enterprise.

EVOLUTION OF ARCHITECTURE

To enable the analysis of data from the control systems for the LHC, data needs to be transferred to a long-term storage Oracle database called the LHC LoggingDB designed to store large amount of data throughout the life time of the LHC [2].

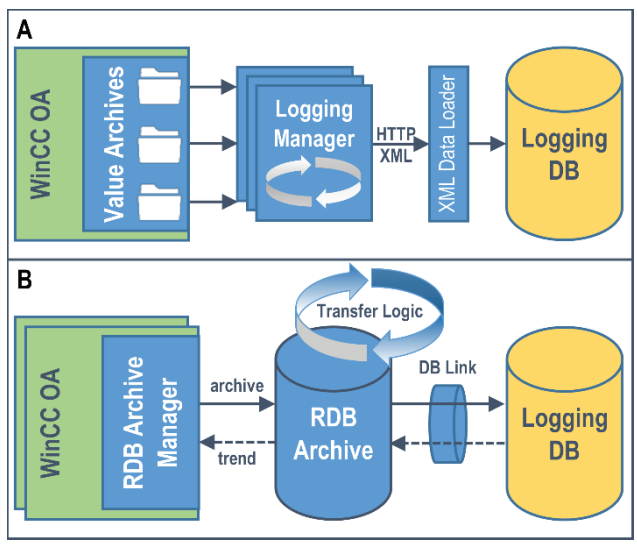


Figure 1: Archiving architecture: (A) with old logging manager; (B) with Oracle archiver and DB-link data transfer.

The evolution of the architecture for archiving is presented in Figure 1. System and process data retrieved by WinCC OA systems used to be fed to the LoggingDB using a dedicated program (Logging Manager) and complex infrastructure (XML Data Loader). This transfer was repeated every couple of minutes, and hence the data was available in the LoggingDB with a lag of

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

approximately 10 minutes. The reliability of the whole process was however affected by the complexity of the system and the significant number of points of failure. A number (sometimes exceeding 10) of WinCC OA archive manager programs needed to be configured in every control system, and configured correctly. Proper tuning of file archiver's parameters was also challenging for systems where the archive rate was varying during operation. Managing disk space and properly configuring data retention policies for disk-stored archive data was not an easy task. Many of these issues would vanish should one use the Oracle RDB archiver instead. Moreover, centralizing all archiving infrastructure in a single database service would simplify many management tasks.

With Long Shutdown 1 the LHC Quench Protection System had to be modernized in order to safely operate LHC magnets at the energy of 13 TeV to be achieved during Run II. The amount of data acquisition channels and data rates have been multiplied leading to the requirement to handle a sustained rate of 150 000 value changes per second, acquired and transferred to the LoggingDB. With the existing architecture, this was not possible to achieve. In addition, the XML Data Loader service of LoggingDB was going to be abandoned.

The new architecture for archiving was therefore proposed, based on the WinCC OA Oracle archiver and a central Oracle database (RDB Archive) providing service to all controls applications. Data transfer from the RDB Archive database to the LoggingDB would then be handled directly between the two databases, over a data link, using the already existing and recommended mechanism used by other clients of LoggingDB.

From a functionality point of view, the elements necessary to build the complete data path already existed and could be adapted using standard database development techniques and tools (views, database links, stored procedures, job scheduling) as well as standard features of the WinCC OA Oracle RDB archiver. However, initial tests uncovered that with existing tools and data organization, the archiving database would not be able to handle the throughput required. It became necessary to optimize the components of the system, starting at WinCC OA, through database schema and storage technology, up to the data-transfer database procedures.

Having data transferred from the RDB archive to the long-term LoggingDB storage resulted in a general policy for data retention for the former one: the data would only be kept for a limited period of time (typically three months). All the data that needs to be conserved for long term storage has to be transferred to the LoggingDB. This led to the formulation of another requirement: as the RDB and LoggingDB are already logically connected to assure data push, it should be possible to transparently retrieve historical data physically present in the LoggingDB from the RDB database; as a consequence, the operator could see the trends of historical values not limited to the three month data retention period.

RDB ARCHIVER OPTIMIZATIONS AND DATABASE REDESIGN.

To probe the performance of the archiver system a large scale test set-up was arranged for: a 2-node RAC Oracle cluster and around 50 servers capable of running up to 200 WinCC OA projects. A test database was provided by CERN's database service, with a highly detailed monitoring tool (Oracle Enterprise Manager) and the support of experts and database administrators interpreting the measured database metrics, providing guidance for tests and tuning database parameters. Similarly, the LoggingDB team provided a replica of their infrastructure and necessary expertise. This allowed us to proceed with integration tests and tuning of the performance of the transfer mechanisms.

The initial tests with the standard settings demonstrated that the requested performance would not be achieved. Even though the performance of data-insert was achievable [3], the readout of data was performing badly due to a bottleneck on the I/O subsystem (i.e. too slow access and low throughput to the disk-based large storage) of the database. Certain additional options like database block check-summing were identified as having high processing costs as well and were disabled.

The key ingredient for performance optimization was the optimization of the readout path, for the most critical task of data transfer to the LoggingDB. This operation requires that a vast majority of data stored into the RDB database is read-out within next 15 minutes to be pushed to the LoggingDB. Data needs to be accessed for all registered signals, one-by-one, for the specified period of time. This access pattern is common with another typical use case for SCADA systems, i.e. access to historical data for a specific signal, in a specific time period to make a trend plot of historical data, or to retrieve data for analysis.

The classical solution for this access pattern is the Index Organized Tables (IOT) data organization, whereby data is organized in the storage such that entries for the same primary key (in this case, the signal) are stored adjacently, making their lookup and retrieval faster. In the previous rounds of RDB archiver optimization efforts [3] the use of IOT was disfavoured for requiring more computing resources during the data-insertion process. We decided to re-visit the performance of IOT with modern hardware and software. To our surprise, the IOTs seemed to be only marginally (<5% of CPU load) more computing-intensive for workloads where 200 systems pushed their data at the accumulated data-change rate of 200 000 values/second. The estimated improvements in the data-readout performance was at least an order of magnitude. Moreover, due to the fact that the indexing data was stored in-line with data, storage space consumption dropped by more than 50% in certain cases [4].

This modification was introduced directly on the database schema, without impacting the source code, and with full compatibility. The optimization could therefore

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

be easily applied to new systems; for all existing systems the re-structuring of large existing data sets would be prohibitively time- and resource-consuming, hence the optimization could be performed for newly gathered data, while keeping the old data in unmodified structure.

However, the database still suffered from the I/O throughput problems related to the so-called REDU log size (auxiliary structure which allows for the restoration of the database in case of failure). The only viable method of reducing its size while still keeping the high data rate was to reduce the size of a row inserted to the database. Then we compared the average row size in the RDB Archive, 59 bytes when using IOTs (or 120 with no IOTs and with additional indices) to 19 bytes in the LoggingDB. It became clear, that a lot of reduction could potentially be achieved.

We reviewed the meta-data stored with every data row of archive data, and identified those containing non-vital information. An extension was then proposed to the already existing concept of WinCC OA *archive groups* to have the archived meta-data parameterizable. The implementation required modification of the database schema (layout of tables, stored procedures), and the WinCC OA Oracle RDB archiver (C++ source code, configuration panels). A prototype was developed at CERN, in collaboration with ETM, and put to test in a test set-up. Fast-changing data generating the bulk of the throughput was directed to the specially crafted custom archive group, where meta-data was truncated to the minimum, whereas data for other signals were archived into the default archive group, with all the details intact. The reduction in data throughput (and also disk space consumption) approached a factor of three, and allowed to keep the performance at the required level.

With a growing amount of data accumulated in tables over time, the performance of data queries may degrade. Even with the queries that access data according to index organization, scanning the huge index tree penalizes the overall performance. We reached for one more optimization level by applying time-based data partitioning. At the same time we also optimized the native data segmentation of RDB archiver, based on table sets and a view accumulating them, by applying pushed predicates (an Oracle hinting technique) on the view. As a result, again, the overall perceived performance of interactive data access as well as data retransmission improved significantly.

With the above optimizations in place we conducted the ultimate performance test to simulate the "worst case scenario" in a set-up mimicking the LHC Quench Protection System, with conditions presented in Table 1. WinCC OA systems were generating value changes with varying rates of changes per second; the data was stored into RDB Archive test database and then transferred to the integration database acting as the LoggingDB. After 24 hours of stable work at this nominal throughput we simulated the scenario of RDB database failure by shutting it down completely. As expected, WinCC OA projects reacted by storing pending data in a buffer formed on their local disks. Then, the database service was re-established and we observed the buffered data being pushed to the RDB test database and then retransmitted to the LoggingDB test database at maximum achievable rate. At the same time, new data was still arriving at the steady rate.

Table 1: Test Setup for QPS Database-Recovery Scenario

WinCC OA projects	50
Archived signals	150 000
Accumulated data input rate (values/s)	200 000
Database outage duration	8 hours
Recovery time	2 hours
Peak recovery rate (insert + transfer)	1 mln rows/s

Even in a degraded mode, with only a single database node in service, it was possible to recover the buffered data. The transmission rate to the database was largely exceeding the rate at which data was generated, resulting in complete recovery in an acceptable time.

The results of these tests convinced us to deploy a production database service for our controls applications. Two dedicated (dual-node) databases were arranged for, maintained by CERN central database service. One database, called QPSR, is dedicated for the QPS applications due to the very high throughput and data reliability requirements. The other, called SCADAR, is used to archive data from all other applications.

Managing data partitions for around 200 database schemas, and executing partition-maintenance tasks (allocating new ones, dropping unnecessary ones) would not be possible without an automated tool. Partition management code contributed by the LoggingDB team was adopted, and then enhanced to allow for more dynamic control on partition allocation. Partition management allowed us also to easily configure and apply data-retention policies for each application: majority of them will have up to 90 daily partitions kept in a moving-window policy, and older ones would automatically be dropped. For some applications specific policies are applied according to requirements (e.g. hourly partitions and 20 days data retention period for the QPS). In any case, data that needs to be kept for longer is preserved in the LoggingDB.

As has happened already in the past, the collaboration between CERN and ETM allowed us to work with the source code of WinCC OA, and have the changes reviewed, accepted and included in the official WinCC OA distribution. Migration tools for existing projects were developed and made available to CERN users.

MIGRATION AND CURRENT STATUS

Migration of existing applications to the new archiving architecture infrastructure was a challenging task not only

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

from a technical standpoint, but also from a coordination point of view. Many of the tasks such as the reconfiguration of WinCC OA projects were automated using the central deployment tool [5]. Provisioning of database accounts (one per application) and maintenance of account credentials was streamlined yet still required manual coordination. A WinCC OA configuration and diagnostic tool for the LoggingDB transfer was developed from scratch and delivered on time to migrate the applications. Each application in turn needed to be migrated with human-supervision. A lot of attention was required when reconfiguring and initiating the LoggingDB transfer for every application to minimize possible data loss. Even though the migration was performed during the Long Shutdown 1 of the LHC (June-December 2014), many applications needed to be already available for pre-commissioning of the LHC (cryogenics, QPS), or were in production (Cooling & Ventilation). This required that the migration, requiring application downtime, is performed as quickly as possible (no longer than 4 hours). During the migration and first months of operation we encountered numerous performance and stability problems, mainly due to improper configuration of data-retention and time-based partitioning. These have since been resolved.

The migration was successfully completed in January 2015 for all production applications. Assistance was also given to the LHC experiments to upgrade their archiving systems and activate the new optimizations.

As of September 2015 there are around 200 WinCC OA projects which make use of the RDB archiver for historical data and data transfer to the LoggingDB. The statistics for the two production databases are summarized in Table 2. We consider that the old method of data transfer to the LoggingDB may therefore be decommissioned now.

Table 2: Statistics for Production Archive Databases

	SCADAR	QPSR
Projects (DB schemas)	140	48
Registered signals	2 000 000	135 000
Rows recorded/transferred	20 mln/h	400 mln/h
Storage space used	11 TB	11 TB
I/O throughput	40 MB/s	70 MB/s

Efforts are currently under way to generate and present daily reports with database statistics, so that the experts might detect unexpected changes in the data rates for their applications and also tune data-reduction (smoothing) algorithms. With the large number of applications to cover and huge data payload this is a non-trivial task.

CONCLUSION

The migration of archiving system for controls application in CERN SCADA Application Service was completed with success and on time for Run II of the LHC. Single, consistent archiving technology is deployed in production CERN-wide, and the need to maintain custom programs for the transfer to the LoggingDB was eliminated. After the initial adaptation and tuning period, the reliability and performance of the service is acceptable, and new functionality was delivered to users.

ACKNOWLEDGMENT

The success of the migration presented in this paper is a fruit of collaboration of experts from three CERN departments: Engineering, Information Technologies and Beams. We would like to express our particular gratitude to Oracle database administrators and LoggingDB experts: Zbigniew Baranowski, Luca Canali, Emil Pilecki, and Chris Roderick. We also acknowledge the development and testing effort of Kacper Szkudlarek, who worked as an ETM/Siemens openlab fellow on improvements in the WinCC OA Oracle archiver.

REFERENCES

[1] SIMATIC WinCC Open Architecture (previously PVSS) SCADA software from ETM (Siemens subsidiary), http://www.etm.at

[2] C. Roderick *et al*, "The LHC Logging Service: Handling Terabytes of On-line Data", ICALEPCS 2009, Kobe, Japan, CERN-ATS-2009-099

[3] M. Gonzalez-Berges, "The High Performance Database Archiver for the LHC Experiments", ICALEPS 2007, Knoxville, USA, (2007)

[4] P. Golonka *et al*, "Performance and Scalability of Oracle RDB Archiver in WinCC Open Architecture 3.11; Test Report", CERN EDMS note 1271192, November 2013.

[5] F. Varela *et al*, "High-level Functions for Modern Control Systems: A Practical Example", ICALEPCS 2013, San Francisco, USA, (2013)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SIS18 UPGRADE: THE FAIR COMPLIANT RENOVATION OF THE DATA ACQUISITION SYSTEM FOR PARTICLE DETECTORS

R. Haseitl, H. Bräuning, T. Hoffmann, K. Lang, T. Milosic
GSI Helmholtzzentrum für Schwerionenforschung GmbH, Darmstadt, Germany

Abstract

In preparation of FAIR, several well-established beam instrumentation systems of the GSI heavy-ion synchrotron SIS18 and its connected high-energy beam transfer lines (HEBT) have to be modernised. In this contribution, the data acquisition upgrade of particle detectors such as ion chambers and plastic scintillators is described. This covers the replacement of outdated custom-built readout- and control hardware by modern FMC (FPGA mezzanine card) based I/O hardware, new multi-channel high voltage power supplies and a new data acquisition system (DAQ) for the VME based scalers. The latter will replace the old Kylix-based ABLASS software by LASSIE (Large Analog Signal and Scaling Information Environment) to fit into the new FAIR control system concept. LASSIE is based on FESA (Front End Software Architecture). FESA was originally developed by CERN and enhanced by GSI-specific modifications. Furthermore, the new particle detector DAQ will be able to take full advantage of the new FAIR timing system which is based on the White Rabbit protocol.

INTRODUCTION

For the upcoming FAIR project, many parts of the existing GSI facility have to be modernised. The new control system for FAIR is already operational and currently under test by different GSI departments. It is based on the FESA framework [1] which originally was created by CERN and is now being developed in a collaborative effort with GSI. FESA controlled devices are automatically integrated in the control system and can be accessed in a standardised way. If existing hardware has to be replaced or new hardware is installed, a FESA class is programmed to provide future-proof operation. Recently, a renovation project was started to replace outdated high voltage power supplies and to upgrade the readout software called ABLASS [2] of the SIS18 and High Energy Beam Transfer (HEBT) particle detectors.

HIGH VOLTAGE CONTROL FOR PARTICLE DETECTORS

Several particle detector systems need high voltages (HV) to operate. Currently three main types of particle detectors are used by the GSI beam instrumentation (BI) group in the synchrotron and the HEBT: secondary electron monitors (SEM), ionisation chambers (IC) and plastic scintillators (SC). It is possible to mount these three detectors onto a single pneumatic actuator which is called Particle Detector Combination (PDC). Along the existing synchrotron and the HEBT 13 PDCs are installed. Besides the PDCs, also other devices like beam loss monitors or profile grids need HV resulting in over 120 required channels. The different detectors need specific voltages ranging from -1500V to +1500V.

The existing HV system is based on SY127 40-channel crates by the company CAEN [3]. The crates have been in operation for over 20 years and replacement modules are slowly phasing out. Moreover they are controlled via the slow CAENET bus which can delay new settings for HV channels.

Succeeding HV systems from the companies Wiener [4], iseg [5] and CAEN are already installed at GSI and prepared for operation. These systems provide Ethernet enabled controllers and various HV modules with different voltage and current ranges.

Although both systems are controlled over Ethernet, they use different protocols: The crates from Wiener, called MPOD, are accessible via SNMP (Simple Network Management Protocol). The HV modules for MPOD are manufactured by the company iseg and are also fully controllable via SNMP. At GSI, two different MPOD systems are used: The 'MPOD Mini' crate with 4 slots (see Fig. 1) and the standard MPOD crate with 10 slots for HV modules. The most commonly used HV modules at GSI are 16-channel modules providing a positive or negative voltage up to 3000V at a maximum current of 3 mA.

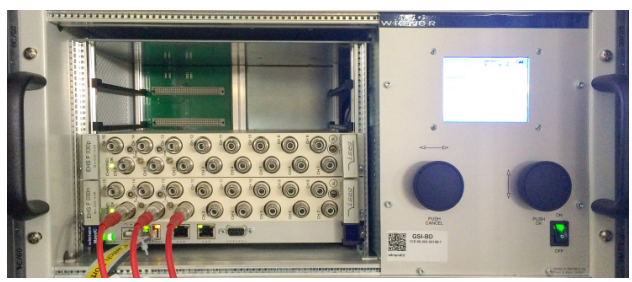


Figure 1: MPOD mini crate with four HV module slots. The lower two slots are populated with iseg modules.

CAEN features a different communication approach: The company provides proprietary libraries to interface with the hardware via Ethernet. The libraries are available for Windows and Linux and can be easily integrated into custom applications like FESA classes. The library offers an efficient monitoring feature for the current line of crates (SY4527 and SY5527). The so-called 'event mode' allows subscriptions to all available crate, board and channel properties. Whenever one of the subscribed values like current, voltage or status registers change, the hardware notifies the application of the change via the library. This way, the devices don't need to

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

be polled periodically reducing unnecessary network load significantly.

Due to the different communication methods, two separate FESA classes have been created to control Wiener and CAEN HV systems respectively. The FESA classes were designed with a common interface in mind. In most cases, it is not important for the control system or a human operator which underlying hardware is providing HV to a device. Using the same software interface in FESA, the hardware is completely hidden from the upper control layers. When voltages, current limits or ramping speeds are set, the FESA class for the specific HV system passes the values to the hardware over the hardware-specific protocol. Any settings or status bits that are specific to one particular HV system are mapped to special fields in the FESA classes and can be controlled through an expert's GUI.

DATA ACQUISITION UPGRADE FOR PARTICLE DETECTORS

The readout of the different particle detector types follows a common principle: Electrical pulses are counted in VME scaler modules. Several VME crates are equipped with Struck SIS3820 scaler boards providing 32 channels each.

Previous DAQ System

The ABLASS system (A Beam Loss measurement And Scaling System), in operation since 2004, is collecting the scaler values and presents them in various GUI applications in the main control room. It is a Kylix based application and hard to adapt to new hardware and software conditions like the new control system. Critical system components like custom-built timing receivers will be replaced with White Rabbit [6] timing receivers in the near future. Therefore, also ABLASS is replaced with a new system called LASSIE (Large Analogue Signal and Scaling Information Environment) [7].

Software Upgrade: LASSIE

LASSIE is already in operation and gradually replacing ABLASS. An overview can bee seen in Fig. 2. The signals of the detectors are pulse shaped and fed into VME scaler modules in three VME crates. The readout is done with FESA classes which provide the measurement values to the control system and GUI applications. Two crates are responsible for the readout of ICs, SEMs and SCs detectors. The third crate is used for advanced signal analysis. It provides the possibility to sample dedicated channels with higher sampling rates. Several GUI applications in the main control room allow the timing related correlation of multiple detector channels. Additionally, other data sources like magnet ramps or signals of rf systems (e.g. synchrotron cavities) can be displayed in conjunction with detector data to provide a powerful tool to the machine operators.

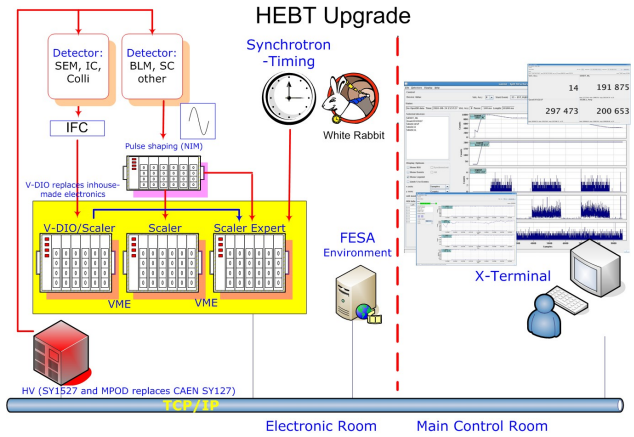


Figure 2: LASSIE overview. Signals from the hardware are fed into scaler modules in three VME crates. FESA classes running on the crates acquire the values and provide the data to the control system or specialised GUI applications.

Hardware Upgrade: V-DIO

SEMs and ICs do not provide a countable signal directly. They provide their measurement value as an electric current which is converted by a specialised hardware called IFC (Current to Frequency Converter) [8] into a countable signal.

The IFC hardware has several discrete control lines for calibration, test signal activation, range setting and polarity setting. Besides the countable measurement signal, it also provides hardware status information via separate lines for overload, device id and gas flow.

The IFCs have been integrated into the old GSI control system using a in-house built hardware setup. The upcoming new control system and the lack of replacement parts for the old hardware made the development of a replacement solution necessary: The VME Digital IO (V-DIO) system was developed by the company MagentaSys [9] based on GSI specifications.

The V-DIO System

The V-DIO system is the new control hardware for the IFCs at GSI and FAIR. To make it more versatile, also a 'general purpose' mode has been implemented, bringing NIM or TTL inputs and outputs for arbitrary applications into the control system. The VDIO system consists of several boards which can be seen in Fig. 3:

- FMC (FPGA Mezzanine Card): Provides ten outputs, two inputs and status LEDs at the front panel.
- SVEC (Simple VME FMC Carrier): Carrier board for up to two FMCs. It was designed by CERN as a White Rabbit Timing receiver and FMC carrier board [10].
- RTM (Rear Transition Module): A 80mm VME RTM board offering four slots for mezzanine boards.
- Mezzanine boards: Mounted on the RTM and connect directly to an IFC. They can be switched to 'general purpose mode' to provide I/O lines for arbitrary use. A close-up can be seen in Fig. 4

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Figure 3: The different boards of the VDIO system. From left to right: small FMC board with front connectors mounted on a 6 HE SVEC VME board, RTM carrier with four rear mezzanine boards, deported IO baseboard with attached mezzanine, single rear mezzanine board.

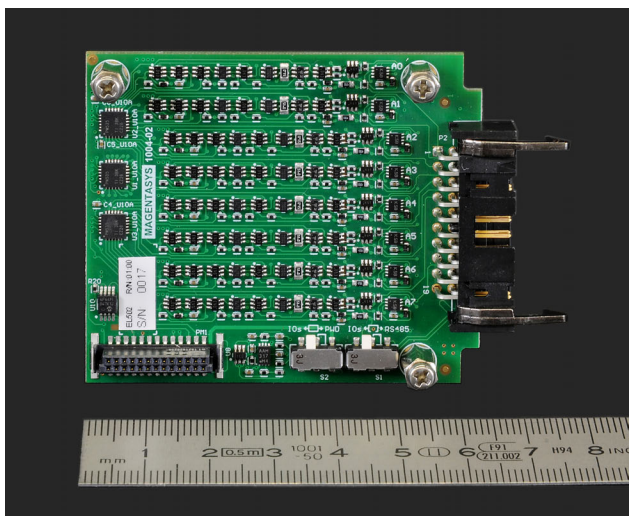


Figure 4: Close-up of the mezzanine board.

- Deported I/O: Optional external baseboard to avoid long IFC cables. It is connected to a special mezzanine on the RTM via separate Cat 5 patch cables for data and power which can be up to 350m long. For data transmission, the I^2C protocol over RS485 is used.

The IFCs are directly attached to the mezzanine boards on the RTM module. The SVEC board hosts an Spartan 6 FPGA which is able to route any signal from the mezzanines to the front connectors of the FMCs. From the FMC, the signal is plugged into a scaler module for pulse counting. Moreover, the mezzanine boards provide power to the IFC electronic. The flexibility of the system is very useful for debugging purposes: The signal of an IFC can be sent to the control system through one FMC output and can be observed using an oscilloscope or an expert system on a second output in parallel.

The front FMC connectors have even more capabilities: They can provide signals with different frequencies and can be logically combined with any input of the FMC modules.

Besides the functionality to control an IFC, the V-DIO hardware can provide and acquire arbitrary I/O signals. Each mezzanine board has 16 I/O lines which can be individually configured as in- or output at TTL or NIM level or as an open-collector output.

The whole V-DIO setup is controlled via FESA. The signal routing from the mezzanines to the FMC front connectors is freely configurable using the FESA instance file. This is an XML-based, editable file to configure the FESA class at startup. To make FESA development for the VDIO system easily adaptable, all hardware-specific functionality has been separated into an external C++ library. This way the IFC and 'arbitrary I/O' modes of the V-DIO system can be implemented quite easily in any application.

CONCLUSION AND OUTLOOK

During the 2015 beamtime, only the low energy beamlines of GSI have been in operation. Therefore, all HV dependent detectors in the synchrotron and the HEBT lines have been upgraded to the new HV system without machine interrup-

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

tion. The new system is currently under extensive testing and will be put in operation with the next beamtime starting in March 2016. LASSIE is already running since 2011 and replacing the old ABLASS system successively. The IFC hardware will be upgraded to the new V-DIO system after the 2016 beamtime, since White Rabbit Timing is not yet available at the existing SIS18 synchrotron.

REFERENCES

[1] M. Arruat et al., "Front-End Software Architecture", ICALEPCS'07, Knoxville, Tennessee, USA, p. 310-312.

[2] T. Hoffmann et al., "Experiences on Counter Applications and Beam Loss Measurement at the GSI Synchrotron", BIW'04, Knoxville, Tennessee, USA, p. 294.

[3] Company CAEN, `www.caen.it`

[4] Company Wiener, `www.wiener-d.com`

[5] Company iseg, `www.iseg-hv.com`

[6] J. Serrano et al., "White rabbit status and prospects', THCOCA02, ICALEPCS'13, San Francisco, USA (2013).

[7] T. Hoffmann et al., "LASSIE: The Large Analogue Signal and Scaling Information Environment for FAIR", MOPMN008, ICALEPCS'11, Grenoble, France (2011).

[8] H.Reeg, "A current digitizer for ionisation chambers/SEMs with high resolution and fast response", PS20, DIPAC'99, Chester, UK (2011).

[9] Company MagentaSys, `www.magentasys.com`

[10] Simple VME FMC Carrier (SVEC) in the Open Hardware Repository: `http://www.ohwr.org/projects/svec`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

UPDATE OF POWER SUPPLY CONTROL SYSTEM AT THE SAGA LIGHT SOURCE STORAGE RING

Y. Iwasaki#, T. Kaneyasu, Y. Takabayashi, S. Koda, SAGA Light Source, Tosu, Japan

Abstract

The control system for the SAGA Light Source storage ring power supplies is being upgraded to increase the ramp-up speed and allow the stored beam energy to be easily changed. By replacing the CPU module in the PLC used to control the power supplies, the ramp-up time was reduced from 4 to 2 minutes in a test bench prepared for the upgraded system. Until now, the allowable beam energy has been restricted to certain fixed values during ramp-up operation due to the original specifications of the PLC ladder program. To operate the storage ring at an arbitrary energy, the algorithm used in the PLC program was changed. The resolution at which the energy can be changed is less than 1 MeV. The upper layer of the control system, which uses National Instrument LabVIEW and ActiveX CA, was also reconstructed to produce a flexible GUI. Preliminary measurements of the energy dependence of the beam size and lifetime were carried out using the updated control system.

INTRODUCTION

The SAGA Light Source (SAGA-LS) is a middle-scale synchrotron radiation facility, at which electrons from a 255 MeV injector linear accelerator are injected into a 1.4 GeV storage ring [1]. The electrons are ramped up to 1.4 GeV in the storage ring during a period of 4 minutes. The storage current is 300 mA. The storage ring lattice is a double-bend type with 8-fold symmetry (see Figure 1). The power supplies for the storage ring magnets are listed in Table 1.

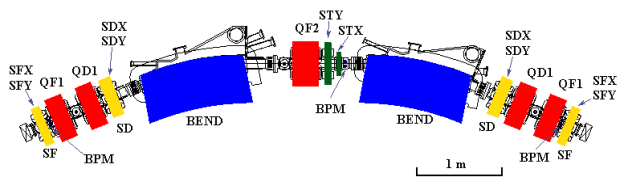


Figure 1: Unit cell of SAGA-LS storage ring.

The control system for the power supplies is constructed using personal computer (PC) and programmable logic controller (PLC) [2]. National Instruments LabVIEW is used for the graphical user interface (GUI). ActiveX CA [3], which provides EPICS channel access for Microsoft Windows, is adopted as the data communication protocol between server and client PCs.

#iwasaki@saga-ls.jp

Although the original control system worked stably, two problems existed with it. The first was the ramp-up speed. The ramp-up pattern file consists of 10,000 elements for each main power supply, and is stored in the internal register of a CUP module in the PLC. The ramp-up speed was limited by the performance of the CPU module. The second problem was a restricted accelerated beam energy. The allowable energy was fixed at certain values due to the original specifications of the PLC ladder program. We therefore replaced the CPU module: F3SP58-6S (YOKOGAWA Electric Corporation) with F3SP76-7S, and modified the algorithm of the ladder program slightly to allow arbitrary energy settings. The upper layer of the control system using LabVIEW and ActiveX CA was also reconstructed. Preliminary measurements of the energy dependence of the beam size and lifetime were performed using the updated control system.

In this paper, we describe the design of the control system for the power supplies at the SAGA-LS storage ring, the control algorithm and GUI, and the preliminary experimental results.

Table 1: Power Supplies for SAGA-LS Storage Ring Magnets

Power Supply	Control Interface	Number
Bending	16bit Digital	1
Quadrupole	16bit Digital	7
Sextupole	16bit Digital	2
BM-Corrector	Analogue , Digital	16
Q-Corrector	Analogue , Digital	40
Steering	Analogue , Digital	80

CONTROL LAYER FOR POWER SUPPLIES

A schematic view of the control layer for the power supplies is shown in Figure 2. Local control of the power supplies is performed by PLC. The PLC consists of one CPU module, an Ethernet module, I/O modules, and optical link bus modules. One CPU module controls all the power supplies of the storage ring. Sub-modules of the PLC are connected to the main unit with optical link bus modules. At the SAGA-LS storage ring, synchronization among power supplies is important, since the electrons are ramped up from 255 MeV to 1.4 GeV in a short time. The PLC outputs 10,000 set points for each

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

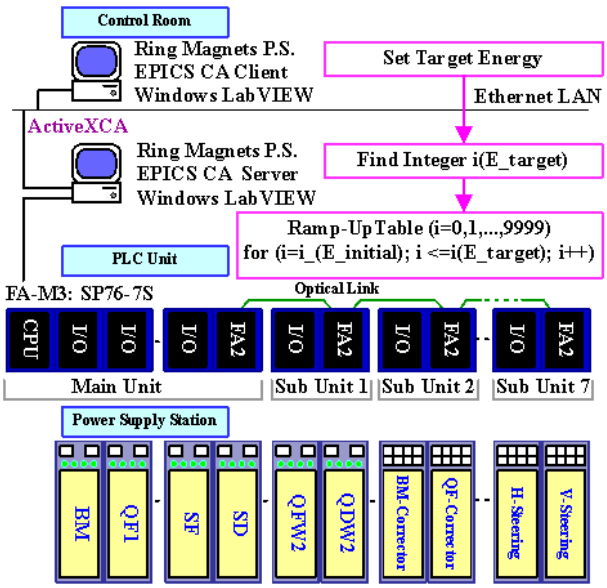


Figure 2: Control layer for power supplies.

of the main power supplies (bending and quadrupole magnets) during the ramp-up period. The configuration of the PLC (one CPU module, I/O devices, and optical liked sub-modules) ensures the synchronization among the power supplies.

Since the lattice of the storage ring is compact and the bending and quadrupole magnets are strongly excited (e.g., the maximum magnetic field produced by the bending magnet is 1.46 T and the maximum magnetic gradient in QF1 is 25 T/m), the ramp-up pattern for these power supplies is not linear. The sextupole and analogue controlled power supplies (BM-correctors, Q-correctors, and steering magnets) are excited in proportion to the QF2 power supply, because the magnetic field produced by the QF2 magnets, for which the maximum magnetic gradient is 18 T/m, increases in proportion to the excitation current. As a result, these magnets are excited linearly with the beam energy.

The server PC running ActiveX CA with the PLC acts as a PC IO controller (PC-IOC). The ActiveX CA client is used to access the server PC with the Process Values. The server and client applications are developed in LabVIEW.

In 2014, two new power supplies (QFW2 and QDW2) were installed for a second superconducting wiggler. The I/O and optical link bus modules for these power supplies were added to the existing PLC modules. An overview of the project for the second superconducting wiggler has been previously reported [4].

ALGORITHM AND GUI

The beam energy E in the storage ring is a function of the excitation electric current I in the bending magnet power supply,

$$E = f(I).$$

The function $f(I)$ is determined from magnetic field measurement data. The power supply current for the bending magnets is determined by an integer i ($i = 0, 1, \cdots, 9999$) stored in the internal register of the PLC,

$$I = g(i).$$

Thus, the beam energy is expressed in terms of i as

$$E = f\{g(i)\} \equiv E(i).$$

If the target beam energy E_{target} is given, then the integer i_{target} is easily obtained as

$$i_{\mathrm{target}} = E^{-1}(E_{\mathrm{target}}).$$

Figure 3 shows the subroutine for finding i_{target}, which is estimated by linear interpolation of $E^{-1}(E_{\mathrm{target}})$ and rounded off to the nearest whole number.

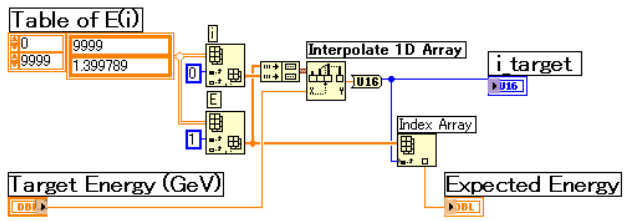


Figure 3: Subroutine for finding an integer for target energy (LabVIEW).

This calculation is performed in the server PC. When the button "Set Target Energy" is pushed on the client PC, the target energy value is sent to the server PC, which then calculates i_{target} and sends it to the PLC. During the ramp-up period, the PLC outputs the ramp-up pattern data for each main power supply from i_{initial} to i_{target} (see Figures 2 and 4). The accelerated beam energy can be easily changed using this PLC and PC based control system.

Because the ramp-up pattern consists of 10,000 set points for each main magnet for energies between 255 MeV to 1.4 GeV, the resulting energy resolution is less than 1 MeV. However, the absolute accuracy of the beam energy depends on both the precision with which the magnetic field produced by the bending magnets can be

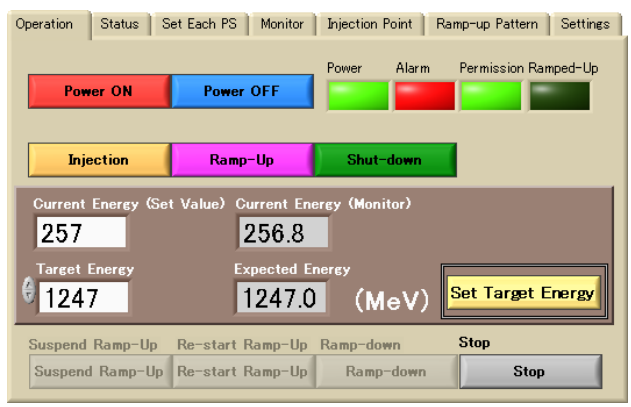


Figure 4: GUI of energy ramp-up client PC.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

measured and the form of the function f in the formula $E = f(I)$. In addition, the stability and the setting precision of the bending magnet power supply influences the beam energy. The absolute energy should be calibrated using another method. At a nominal beam energy of 1.4 GeV, the actual energy was estimated to be 1417±4 MeV by a laser Compton scattring experiment [5].

PRELIMINARY ENERGY DEPENDENCE EXPERIMENTS

Preliminary measurements of the energy dependence of the beam size were carried out using the synchrotron radiation (SR) interferometer system [6]. Figure 5 shows the transverse beam size for injection energies of 255 MeV to maximum energy of 1.4 GeV. Since the SR interferometer system was optimized to measure the beam size at 1.4 GeV, the results indicating horizontal and vertical beam sizes of greater than 800 and 1000 μm, respectively, are unreliable.

At the same time, the beam lifetime was measured. The product of the beam current and lifetime is shown in Figure 6. The peak at an energy of about 0.6 GeV may be related to the increase in the vertical beam size. Since the betatron tunes strongly vary during the ramp-up period (i.e., horizontal: 0.13, vertical: 0.11) at the SAGA-LS storage ring, quantitative estimation of the beam lifetime should take into account the optical modulation at each energy. Detailed measurements and quantitative estimations of the energy dependence of the beam size and lifetime have not been performed yet. However, energy-dependent experiments can be easily carried out using the updated power supply control system. In addition, we plan to investigate the beam energy dependence of ion trapping and intra-beam scattering (IBS) effects using this system.

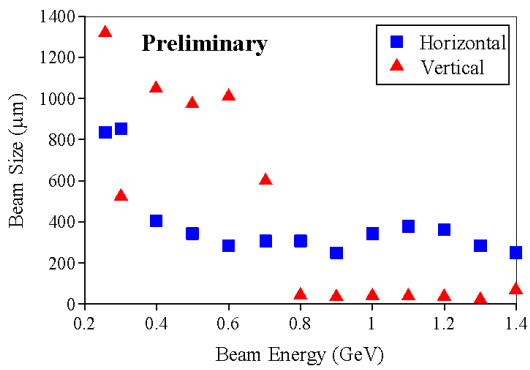


Figure 5: Dependence of transverse beam size on energy from 255 MeV to 1.4 GeV.

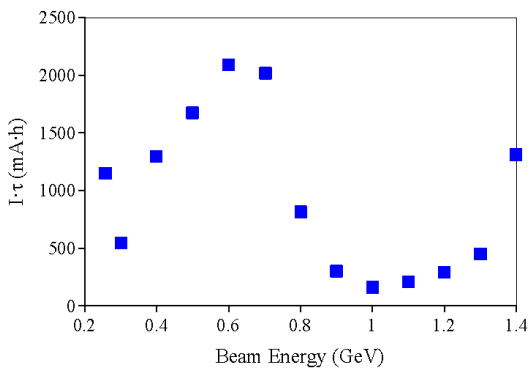


Figure 6: Energy dependence of product of beam current and beam lifetime.

SUMMARY

In order to increase the ramp-up speed of the SAGA-LS storage ring, we replaced the CPU module in the PLC. The ramp-up time was reduced from 4 to 2 minutes in a test bench prepared for the upgraded system.

The algorithm and the GUI for the control system were also modified to change the operating energy of the storage ring. The resulting energy resolution was less than 1 MeV. The accelerated beam energy can be easily changed using this PLC and PC based system.

Preliminary measurements of the energy dependence of the beam size and beam lifetime were carried out using the updated control system. The observed lifetime peak at about 0.6 GeV may be related to an increase in the vertical beam size. In a future study, we will investigate the beam energy dependence of ion trapping and IBS effects at the SAGA-LS storage ring.

REFERENCES

[1] Y. Iwasaki et al., "Lattice Design of SAGA Synchrotron Light Source", Proceedings of PAC'03, Portland, 3270, (2003).

[2] H. Ohgaki et al., "Design of Control System for SAGA Synchrotron Light Souece", Proceedings of PAC'03, Portland, 2387, (2003).

[3] Kay-Uwe Kasemir, https://www.slac.stanford.edu/grp/cd/soft/epics/extensions/ActiveXCA/readme.htm

[4] Kaneyasu, et al., "Installation of a Second Super Conducting Wiggler at SAGA-LS", 12th International Conference on Synchrotron Radiation Instrumentation (SRI2015), in press.

[5] T. Kaneyasu et al., "Generation of laser Compton gamma-rays in the SAGA light source storage ring", NIMA 659, 30, (2011).

[6] Y. Takabayashi et al., "Beam-size Measurement System at the SAGA-LS Storage Ring", Proceedings of ICALEPCS'09, Kobe, 140, (2009).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

TESTING FRAMEWORK FOR THE LHC BEAM-BASED FEEDBACK SYSTEM

S. Jackson, D. Alves, L. Di Giulio, K. Fuchsberger, B. Kolad, J. Pedersen
CERN, Geneva, Switzerland

Abstract

During the first Large Hadron Collider (LHC) [1] shut-down period, software for the LHC Beam-based Feedback Controller (BFC) and accompanying Service Unit (BFSU) [2] was migrated to new 64-bit multi-core hardware and to a new version of CERN's FESA [3] real-time framework. This coincided with the transfer of responsibility to a new software team, charged with readying the systems for beam in 2015 as well as maintaining and improving the code-base in the future.

In order to facilitate the comprehension of the system's 90'000+ existing lines of code, a new testing framework was developed which would not only serve to define the system's functional specification, but also provide acceptance tests for future releases. This paper presents how the BFC and BFSU systems were decoupled from each other as well as from the LHC plant's measurement and actuator systems, thus allowing simulation-data driven instances to be deployed in a test environment. It also describes the resulting Java-based Domain-Specific Language (DSL) which allows the formation of repeatable acceptance tests.

INTRODUCTION

LHC operators rely on a feedback system in order to stabilise/correct and adjust the beams' closed-orbit, betatron tune, energy and radial loop [4] during the various stages of the operational cycle from filling to flat-top. The system is triggered at up to 25Hz by input signals from over 1'000 Beam Position Monitors (BPM), along with 6 tune measurements from Base-Band Q (BBQ) [5] measurement systems. The BFC sanitises this data, before calculating the necessary currents to send to various LHC steering dipoles and quadrupoles - The effects of which should be observed in the next iteration of the feedback loop (via the BFSU 1Hz instrumentation layer).

When a new team took charge of the existing codebase, it was apparent that many changes would be required to port the code to new 64 bit hardware, and to migrate to the latest FESA framework. This meant that testing the new software before the LHC start-up would be imperative. Testing using real signals from over 1'500 I/O devices round the LHC would not only be difficult during the machine's shutdown, but would be impossible when the LHC became operational. Consequently it was decided to create a testing framework which could emulate the BFC's input signals, the effect of which could then be observed via the BFSU at 1Hz. Test cases would then be written, for asserting that the BFC and BFSU were reacting correctly to the conditions defined by the test. It was also acknowledged that test developers may not be software experts, and may therefore be uncomfortable learning how to use a potentially complex API. In this context, a descriptive DSL would abstract the testing framework to be easily used by test creators.

Realising the framework, would not only require new testing code, but would entail many changes to the BFC and BFSU themselves, in order to allow instances of the software to be deployed outside their specialised operational hardware. Also, spoofing the BFC's input data would require relaxing of the BFC's data integrity checking mechanisms without compromising security in the operational system.

ADAPTING TO LIFE IN THE LAB

During 2014, the BFC and BFSU were ported to the new 64 bit architecture / FESA3 framework and the team had a release candidate ready for testing. Before any testing framework could be developed however, several changes to the software would be required to allow the software to be tested.

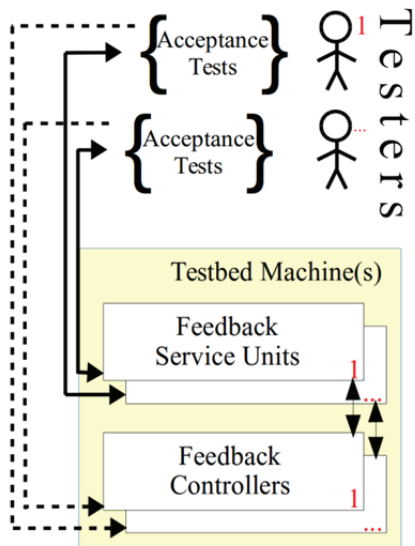


Figure 1: Software adapted so any number of BFC / BFSU pairs can be started, even on the same machine.

The new operational feedback system comprises of 2 multi-core Linux based (with Red Hat's MRG* kernel extension) HP Gen8 Blade machines. In order to assure real-time performance, the BFC is isolated from the LHC control system, with the BFSU acting as a client-facing data / setting proxy. A 2nd Ethernet link is used to transfer (~130Mb / sec) data between the BFC and BFSU

* Messaging Real-time Grid.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

during operational conditions. Reproducing this exotic hardware configuration would be both costly and unnecessary for our testing framework, so the code was modified to allow any BFC instance to be connected to any BFSU on any IP address including loopback (i.e. both on the same machine). Deploying in loopback proved problematic with the original software design, as the BFC and BFSU were both bound to similar network ports, so the 36 existing hard-coded port numbers where converted into offsets with respect to a single configurable base-port. This way, any number of BFC / BFSU unit pairs could be started on the same machine without having any port clashes (see Fig. 1).

The BFC design is different to most of the real-time software developed for the LHC due to its tight real-time constraints. Consequently, the system was compiled and linked on the operational hardware itself, using machine optimized compilations of CERN's Root libraries. This was not ideal for compiling and testing via the new framework, so the system build was adapted to use the standard SLC6 environment. It was assumed that with the latest 64 bit hardware increasing CPU power from 4 core to 24 core machines, real-time constraints would not be compromised.

TEST SYSTEM OVERVIEW

While testing a system would traditionally start with writing unit tests for individual components and then building integration and acceptance tests on top of these fundamentals, the approach taken here was a bit different. Starting with unit tests was problematic, because classes on the BFC have a high inter-dependency. Our main goal was to bring the system (focussing mainly on the BFC) into a test harness in order to observe and verify the global behaviour while gaining confidence for future changes. To accomplish this, it became clear that the BFC's I/O channels (UDP packets, described later) could already provide us with a clean way to inject our tests. This choice has the following advantages:

- Minimal changes required in the BFC.
- UDP is language independent; we were free to choose any language for the test system.

Given the option of choosing any language, we decided to use Java as the implementation language for the testing framework, for the following reasons:

- Java is widely used in the LHC environment, so more people will be able to easily use and extend the framework providing more tests.
- Interactions with other parts of the control system (e.g. settings management) are natively possible in Java. This allowed us to easily write tests involving operational settings and/or even verify consistency of such settings with the framework.
- It allowed us to gain first experience in the behaviour of Java in a real-time environment.

An important aspect taken into account in the design of the testing framework was the fact that tests of different layers would be required. We identified the following types of tests:

- **FESA mechanics**: e.g. asserting that setting a value in one FESA property has the desired effect in another. Despite the fact that this functionality is only for interfacing with the control system, these tests turned out to be the most frequently required, due to the complexity of the BFSU's API.
- **Communication**: e.g. send some predefined values for an orbit and check if the values are correctly processed through the layers.
- **Control loop behaviour**: e.g. send a constant orbit verifying the resulting corrections. From an operational viewpoint these are the most interesting tests, as they highlight instabilities and allow error predictions.

These different abstraction layers are reflected in the testing framework's DSL as described later.

PLANT VIRTUALISATION

Without input signals the feedback system does nothing apart from reporting timeouts. To operate correctly, the BFC needs the following (see Fig. 2):

- **BPM packets**: Sent at 25Hz from the BPM systems, they contain position measurements (~500 values per beam and plane).
- **BBQ packets**: Sent between 1 and 100Hz from 3 BBQ systems, they contain the measurement values for the tunes (1 value per beam and plane).
- **Orbit trigger**: Hardware trigger at 25Hz, which starts orbit averaging in the BPMs, and also data collection in the BFC. On reception of this trigger, the BFC opens an acceptance window (~10 ms), in which arriving packets are processed for one feedback iteration.

In order to test the system, the testing framework must virtualise these BPM, BBQ and Orbit Trigger input signals as well as capture the output signals (to avoid sending to the correction magnets). Luckily, most of the input signals already arrive via UDP packets, so spoofing the data was feasible. The only exception in the new feedback system is the Orbit Trigger which arrives via a cable from the Beam Synchronous Timing (BST) [6]. To facilitate the spoofing of this input signal, code was modified in the BFC, and the trigger's interrupt was handled in a dedicated thread which delivered the trigger into the BFC's main loop via a UDP packet. Although this added a small jitter to the trigger, the additional overhead of passing through a local UDP port was deemed acceptable.

All the operational I/O systems communicating with the BFC are written in C++, and the UDP packets' content are declared as C structs. Our chosen language for the test however was Java, so these structs had to be emulated by Java classes which could be de-serialised / serialised to / from a stream of bytes which mimicked exactly the structs from the real systems. Also, the contents of our packets had to adhere to the strict data

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

quality rules specified by the BFC, including information used for diagnostics. Even if we were not interested in testing things such as temperature from our framework, this diagnostics data must be within sane limits.

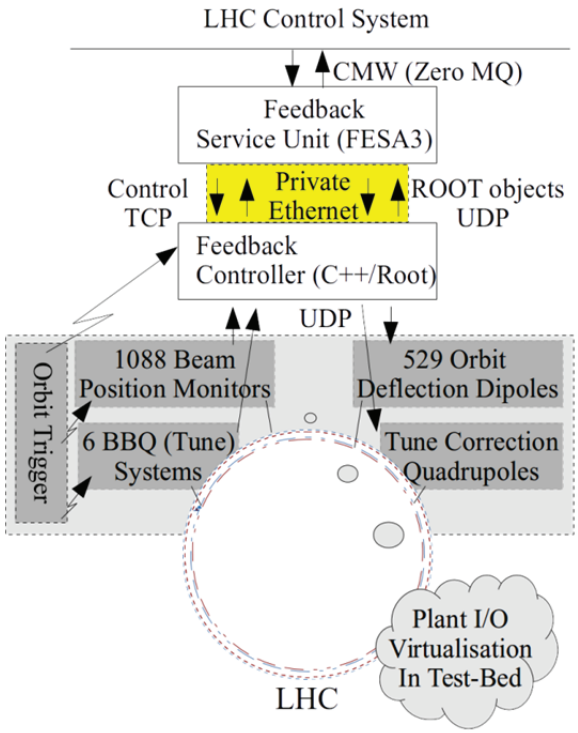


Figure 2: Framework must virtualise LHC I/O signals and decouple from private Ethernet dependency.

TEST FRAMEWORK ARCHITECTURE

Figure 3 shows the core components of the testing framework and their different layers as follows:

- The main work and internal logic is done in the service layer. It combines data from the lower layers and provides convenient methods for the upper layers for dedicated tasks (e.g. 'create an orbit with certain properties').
- The lowest *backend* layer contains various adaption / delegation classes for subscribing to FESA devices, creating simulated orbits, reading settings, creating UDP packets, etc.
- From a functional perspective, the service layer would be sufficient to write the tests. However, to make tests more expressive and readable by non-programmers, we followed domain driven principles and put a layer of a Java-embedded Domain Specific Language (eDSL) on top. It was implemented as a fluent Java API, using method chaining as the main syntax concept. The starting points for all the fluent clauses are grouped by topic in classes (e.g. OrbitCreationSupport for clauses which create different orbit types).

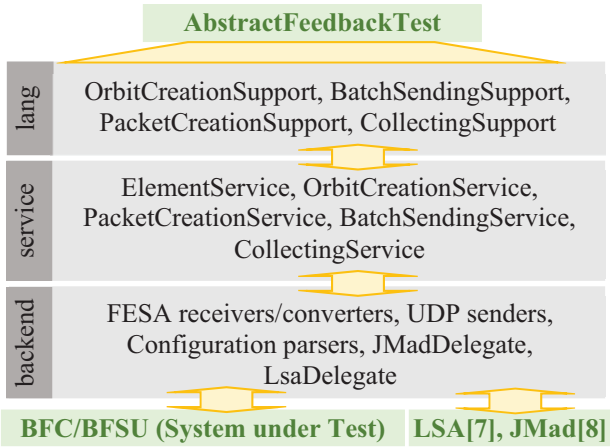


Figure 3: Overview of the testing framework architecture.

As a framework to execute our tests, we chose the JUnit framework, which simplified test creation and report generation as well as using well established matching libraries (hamcrest, assertj) to formulate assertions. The framework provides an abstract class, *AbstractFeedbackTest*, which can be used as a base class for all tests. It is preconfigured with instances of each support class and provides delegation methods to them, so that inheriting tests can formulate fluent clauses without instantiating any additional objects.

THE DOMAIN SPECIFIC LANGUAGE

To get a general impression of the different features of the testing framework and the eDSL in particular, we will show some selected examples. Since the execution of the tests is based on JUnit, the skeleton of a test will always look something like this:

```
public class ExampleFeedbackTest extends AbstractFeedbackTest {
    @Test
    public void assertThatSomeValueIsCorrect() {
        /* Test Code goes here */
    }
}
```

Most tests involve communicating with the BFC, so the language provides the means to construct packets of a particular type and send them:

```
Burst burst = dummyBpmPackets(NUMBER_OF_PACKETS_PER_BURST,
                              constant("X"), BEAM_POSITION);
prepareSendingOf(burst).during(5, SECONDS).
                every(40, MILLISECONDS).andStartAndAwait();
```

A burst represents a set of packets which simulate the state of the LHC at a moment in time (i.e. one orbit). A more meaningful test would be to construct an orbit with all zero values and send it (this time using defaults for duration / period, spawning the sending in the background, checking the publishing from the BFSU and then awaiting the termination of the sending):

```
prepareSendingOf(zeroOrbit()).andStart();
MultibeamOrbit acquiredOrbit = from(MultibeamOrbit.class).
                                skip(1).and().awaitNext();
awaitLastSending(); /* assertion here */
```

Creation of other orbits is also supported. e.g. Constant single outliers or (artificial) flat orbits with constant values:

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

```
MultibeamOrbitReading first =singleOutlierOf(1.2f, MILLIMETER).
                        at(Bpm.ofName("BPM.24R2.B1"), HORIZONTAL);

MultibeamOrbitReading second = sameValueOrbitOf(4, MILLIMETER);
```

Finally, the language also provides a convenient way to retrieve and manipulate FESA properties and fields.

```
String opticsName = from(OPTICS).doGet().getActiveOpticsName();

to(ACTIVE_OPTIC).partially().setFetchOFC_object(true);
```

This last feature turned out to be very useful for other applications and will be extracted in the near future to a dedicated package to facilitate its re-use by others.

RESULTS

Within minutes of first beam, the new feedback system was supplying operators with orbit data, as well as successfully applying orbit and tune feedback. When operators tried to load different reference optics however, the new software didn't behave as expected. Despite testing most of the basic system functionality, the framework didn't have a dedicated acceptance test for reference optic changes, but the team were able to quickly reproduce the phenomenon in the testbed, solving the problem without further disturbing operations. Another problem not covered by the tests involved a rogue BPM system sending corrupted timestamps which caused the Software Interlock System[9] to dump the beam several times. Again, the team reproduced the phenomenon in the testbed, and quickly supplied a patch to fix the problem operationally.

A weakness of the framework was highlighted during the first few months of operation, when problems with the delivery of timing events occurred during optics reference changes. In this case, the testbed was unable to reproduce the problem unless an accompanying test framework for the timing event delivery was also available. In the long term, it is hoped that such a framework will become available, but in order to solve this issue, the operational system had to be used to diagnose and solve this problem.

FUTURE PLANS AND OUTLOOK

The acceptance test coverage is still relatively low, so the team plans to continue developing new tests for existing functionality as time allows. However, any new feature added to the BFC or the BFSU must have accompanying acceptance tests if possible.

During the next 12 months, the team plan to implement a new streamlined BFSU API which will use new features now offered by the latest FESA3 framework. The delivery of this new API will be progressive, and released in parallel with the existing API. In order to assure that the new API is functionally equivalent, the plan is to duplicate the existing tests and adapt them to the new API. A third set of tests will then assert that the results of the tests with the old API and new API are equal.

Presently, the output of the BFC in test conditions is muted. There is an ambitious plan to capture the BFC output (UDP packets destined for the corrector magnets) within the acceptance test, and combine it with the input signals for the next iteration. With this in place, we effectively close the feedback loop within the testing framework.

There are also plans to use the experience from this testing framework to test other systems. Several systems such as the BBQ tune measurement system and the LHC beam-loss monitoring system will also require porting to the new FESA3 framework, and will therefore benefit from offline testing. Whilst the scope of tests on these systems will be less than for the feedback system (they rely on dedicated hardware), basic I/O tests can still be performed. In the long term, these systems could also benefit from more extensive acceptance tests, if like the feedback system, the systems are re-written to accommodate simulated data in place of acquisition cards for example.

CONCLUSION

The testing framework enabled the new software team to understand the code they inherited, and provided functional documentation in the form of test assertions. The tests highlighted many bugs which would probably have gone un-detected until the LHC start-up; bugs which would have hindered the operators during a very critical period. Importantly, the testbed has also given the new software team the confidence to roll out several changes to the software during 2015, citing the testbed assertions as confirmation that the new release should be acceptable. Of course the testbed will never guarantee that a new release is bug-free, but the confidence is certainly increased.

The resulting framework has proved that one of the most complex pieces of software in the LHC can be tested offline, and should serve as an inspiration to develop other test frameworks for other critical systems at CERN.

ACKNOWLEDGMENT

The team would like to thank Jörg Wenninger from the Operations group, Vito Baggiolini from the Controls group and Markus Zerlauth from the Machine Protection and Electrical Integrity Group for their considerable contribution of time and advice during the project.

REFERENCES

[1] Lyndon Evans and Philip Bryant, "LHC machine", JINST, 3 (2008), p. S08001.

[2] L. K. Jensen et al, "Software Architecture for the LHC Beam - Based Feedback System at CERN", ICALEPCS'13.

[3] M. Arruat et al., "Front-End Software Architecture", ICALEPCS'07.

[4] R. Steinhagen "Real-Time Beam Control at the LHC", PAC'11.

[5] M. Gasior et al., "Advancements in the Base-Band-Tune and Chromaticity Instrumentation and Diagnostics Systems during LHC's First Year of Operation", DIPAC'11.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

[6] D. Domínguez et al., "An FPGA Based Multiprocessing CPU for Beam Synchronous Timing in CERN's SPS and LHC", ICALEPCS'03.
[7] G. Kruk et al., "LHC Software Architecture [LSA] -- Evolution Toward LHC Beam Commissioning", ICALEPCS'07.
[8] K. Fuchsberger et al., "Status of JMad, the Java-API for MADX", IPAC'11.
[9] L. Ponce et al., "Operational Experience with the LHC Software Interlock System", ICALEPCS'13.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ENHANCING THE DETECTOR CONTROL SYSTEM OF THE CMS EXPERIMENT WITH OBJECT ORIENTED MODELLING

R. Jiménez Estupiñán, A. Andronidis, T. Bawej, O. Chaze, C. Deldicque, M. Dobson, A. Dupont, D. Gigi, F. Glege, J. Hegeman, M. Janulis, L. Masetti, F. Meijers, E. Meschi, S. Morovic, C. Nunez-Barranco-Fernandez, L. Orsini, A. Petrucci, A. Racz, P. Roberts, H. Sakulin, C. Schwick, B. Stieger, S. Zaza (CERN, Geneva, Switzerland), U. Behrens (DESY, Hamburg, Germany), O. Holme (ETH Zurich, Switzerland), J. Andre, R. K. Mommsen, V. O'Dell (Fermilab, Batavia, Illinois, USA), Petr Zejdl (Fermilab, Batavia, Illinois; CERN, Geneva), G. Darlea, G. Gomez-Ceballos, C. Paus, K. Sumorok, J. Veverka (MIT, Cambridge, Massachusetts, USA), S. Erhan (UCLA, Los Angeles, California, USA), J. Branson, S. Cittolin, A. Holzner, M. Pieri (UCSD, La Jolla, California, USA)

Abstract

WinCC Open Architecture (WinCC OA) is used at CERN as the solution for many control system developments. This product models the process variables in structures known as *datapoints* and offers a custom procedural scripting language, called Control Language (CTRL). CTRL is also the language to program functionality of the native user interfaces (UI) and is used by the WinCC OA based CERN control system frameworks. CTRL does not support object oriented (OO) modelling by default. A lower level OO application programming interface (API) is provided, but requires significantly more expertise and development effort than CTRL. The Detector Control System group of the CMS experiment has developed CMSfwClass, a programming toolkit which adds OO behaviour to the *datapoints* and CTRL. CMSfwClass reduces the semantic gap between high level software design and the application domain. It increases maintainability, encapsulation, reusability and abstraction. This paper presents the details of the implementation as well as the benefits and use cases of CMSfwClass.

INTRODUCTION

The Detector Control System (DCS) applications of the CMS experiment are written with WinCC OA, mostly using the native CTRL language with the process variables being modelled in tree-like data structures, called *datapoints*. *Datapoints* are the default persistence layer and they are used by the runtime database to hold the process variables. CTRL language does not include a type definition syntax to declare and manipulate new structures. This can only be done by means of functions from the standard library or through the WinCC OA graphical editing interface. Low encapsulation and coupling between CTRL language and WinCC OA *datapoints* make data manipulation complex, compared to other languages.

The design of software, from the software engineering point of view, must be an independent process, unaware of the technology specifics [1]. During the formalization of a software design, abstract domain problems need to be transferred into algorithms and organized in data models. For that reason, the resources available in the programming platform, such as data structure definition mechanisms or programming language features, have a direct impact on the project implementation. When the semantic distance between the modelling language (e.g. UML) and the programming language is large, then the time and complexity of the translation process increases. Also other aspects of the software may be affected, like the readability of the code and maintainability of the entire software.

The CMS DCS team has attempted to close this gap by creating a development toolkit to add OO behaviour as well as code and data encapsulation down to the datapoint level. CMSfwClass toolkit revises the original WinCC OA and JCOP framework [2] device modelling concepts, enabling more abstract and powerful software architectural designs.

CMSFWCLASS TOOLKIT

The toolkit is composed of two different layers. The first layer is the back-end to add object orientation into WinCC OA. The second toolkit layer is a graphical user interface to provide easy and comprehensive access to the object oriented abstraction layer.

Figure 1: CMSfwClass toolkit modules.

Though the OO back-end has been designed to work in any WinCC OA project, the *GUI* has a certain level of integration with the CERN JCOP framework. Classes can be registered and accessed as JCOP device definitions. The *GUI* also uses CERN made libraries for syntax checking capabilities.

CMSfwClass OO Back-end Features

The OO back-end of CMSfwClass provides most of the commonly used features of modern OO programming and some custom object management features:

- Single inheritance and method overriding.
- Interface definition.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- Subtyping and interface polymorphism.
- In-memory objects (without *datapoints*).
- Object serialization in files.

The toolkit provides a default base class (CMSfwObject). This class serves as a baseline for other classes and provides several useful methods to handle objects at any level in the class hierarchy. CMSfwClass implements some custom features to express other design aspects of the model, such as specialization of objects at runtime, or composition and aggregation relations between objects. These features facilitate object management; for instance cascaded object deletion.

CMSfwClass toolkit GUI features

The CMSfwClass GUI was conceived as a computer-aided software engineering tool. The user interface guides developers during the process of creating classes and objects. The user drives the development process through different panels where attributes and methods can be introduced, while consistency and programming best-practices are assured by CMSfwClass.

The most relevant features of the GUI are the following:

- Code generation.
- Guided development process.
- Syntax checker.
- Object management.

The user interface provides code generation capabilities and live messages to remind the user how to proceed during the construction of a class. When a new class is created based on a super class, the constructor already includes code for inheriting the behaviour of the super classes. Then developers can override the default behaviour. When a new attribute is added to a class, *datapoint* structures are automatically altered to include the new element. The toolkit GUI offers the developer the possibility of creating accessor methods (getters and setters) for all class attributes. To insert method implementation details, the WinCC OA CTRL language editor is opened automatically for the scope of the method.

Apart from enforcing many of the development guidelines in the auto-generated code, CMSfwClass toolkit drives the users through different panels and pop-up messages to avoid several classic programming mistakes. For example, when adding a new attribute of class type (by composition), the toolkit warns the user about the need of updating the constructor and destructor methods.

Since CTRL language is an interpreted language, there is no compiler at our disposal. Basic syntax checking is provided in a library created at CERN. The library gives information about common defects of the code such as missing variable declarations, missing return statements and other problems that would otherwise only be discovered at runtime. It also contains some other useful functionality to extract function signature information and function location in the analysed files.

Another important aspect of the GUI is the object browser and operation interface which give a comprehensive visualization of the data structures. From the class browser, users can navigate across the class hierarchy, edit class definitions, access object collections and execute the methods of any object.

DEVICE MODELLING USING CMSFWCLASS

WinCC OA *datapoints* allow engineers to model pieces of hardware and logical entities in a hierarchy of basic data structures. The JCOP framework additionally provides utilities to register, configure and handle *datapoints* as devices [3]. CMSfwClass toolkit goes one step further and adds full encapsulation by putting together the device data structure and its behaviour in a single file. A class definition file also contains a description of how the model interacts with other classes and libraries. Classes are written in CTRL language and include the following sections:

- Header: Libraries and constant values.
- List of attributes: primitive types or class types.
- Interactions: parent class, implemented interfaces.
- Class methods.

With CMSfwClass we can model hardware entities and connections between them using classes. To do this, we need to classify components by their common features, to make a proper division of concerns

A possible device classification is in Figure 2, where the model starts with a generic channel class definition inheriting from the baseline class CMSfwObject. An object of this class implements the basic behaviour of a channel. For example, actions to switch the device ON/OFF or a method to access the status of the channel. The original behaviour of a channel baseline class can be overridden to compile with different specifications for high and low voltage channels.

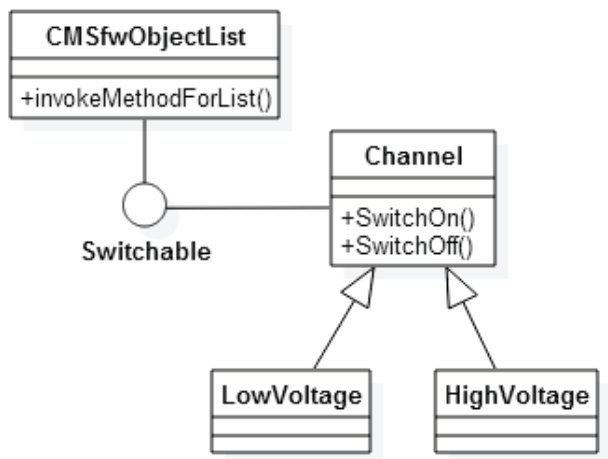


Figure 2: hardware model example.

This particular organization of entities implemented in objects can be useful when handling large, heterogeneous item collections. We can treat them in the same way since they inherit from the same base class “channel” and they implement the interface “Switchable”. We can easily go across the collection and perform the “SwitchON”

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

operation, abstracting from the detailed implementation required for every channel.

ARCHITECTURAL DESIGN USING CMSFWCLASS

In addition to modelling devices, developers can use CMSfwClass to model abstract domain problems. The CMS DCS team decided to refactor a small application called CMSfwScheduler, to get a more maintainable and clean implementation of the program. A comparison of the two implementations, with and without CMSfwClass, is shown in Table 1.

Table 1: Code Metrics

	Procedural	CMSfwClass
Code files	4	2
Core code lines	545	369
Auto-generated	0	329
Total	545	698

In this example, we see there is 47 % of code that has been automatically generated by CMSfwClass to perform OO consistent operations. As a result, by implementing the same program with CMSfwClass, the developer wrote 32% fewer lines of code. This comparison does not count the time spent on implementing data structures, defining naming conventions to handle data or creating user interfaces to access the data. These features are available by default in CMSfwClass and also speed-up the development process.

When control system software reaches a certain complexity, it is much easier to translate to code using OO than a procedural approach. The implicit mechanisms of OO modelling are meaningful during the software design but also in the code. CMSfwClass empowers the code by doing complex operations in fewer lines. Design patterns can also be applied in this context, providing tested, proven development paradigms.

TOOLKIT IMPLEMENTATION DETAILS

One of the goals of CMSfwClass is to help create a proper separation of concerns when modelling software. For that reason, the engine itself complies with modern software engineering principles such as modularity, encapsulation and information hiding. The toolkit is composed of different modules.

Model Data Hierarchy

The first time a class file is used in the toolkit it has to be registered. The registration of the class file creates the necessary internal structures to operate with the class, establishing the relation with other available classes and binding the library to a particular *datapoint* type. For every class in the hierarchy there will be a *datapoint* type using the class name and grouping the specific attributes for that class. For every object there will be one *datapoint* per implemented class. The tool transparently handles a parent-child representation of the classes to determine where to find a particular attribute in the hierarchy. Thus CMSfwClass maintains a clear separation of concerns between attributes and objects of different classes.

OO Syntax in CTRL Language

The implementation of OO features is subject to certain limitations of the CTRL language. Methods and object names have to fulfil the following naming convention and rules:

- A method always uses the first parameter to transfer the object name inside the function scope.
- References to objects and class names are stored in variables of type string.
- A method is uniquely identified by its signature, using the class and method name with underscore in between: *<class name>_<method name>*
- Overriding a method implies changing the signature of the function, using the class name where the method is implemented.
  - *<class-A>_<method name>*
  - *<class-B>_<method name>*
- Object names are unique.
- An object can implement many classes, and its attributes are distributed in many *datapoints* (one per class) using the following convention:
  - *CMSfwClass/<class-A>/<object>*
  - *CMSfwClass/<class-B>/<object>*
- Objects can be referenced by any of its *datapoint* names.

Subtyping and Interface Polymorphism

Object information and behaviour can be hidden using subtyping and interface polymorphism. CMSfwClass considers that the relation between a class and its super class is an inclusive specialization of the superclass (subtyping). Therefore, methods written in the super class can operate on objects of the subclass. This feature enables treatment of objects of a specialized class as if they were instances of any of its super classes. Indeed, since CMSfwClass distributes the class attributes through different *datapoints* (one structure per implemented class), we can use any of its *datapoint* names to reference the same object. In CMSfwClass this feature is called *object shapes*.

Interfaces are also used to abstract and hide object details. Interfaces can be used to define what objects must offer in different parts of the software and to limit contexts only to objects implementing a certain interface.

Single Inheritance and Method Overriding

CMSfwClass implements single inheritance. This means that a class can only extend the functionality of a single super class. This mechanism provides children classes not only with all the attributes declared in the extended class and above, but also with all the behaviours described in the class hierarchy.

There are two different ways of invoking a method. Developers can explicitly invoke a method using its

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

original function name, or they can delegate to CMSfwClass to perform a dynamic method invocation (using wrapper functions). Dynamic method invocation verifies the method's accessibility and looks for the most specialized version of the method. The class an object is instantiated determines the method implementation to be executed. However, the shape of the object determines the accessibility of its attributes and methods. Since there might be more than one method implementation, CMSfwClass performs a search across the hierarchy from the instantiated object class to the less specialized class.

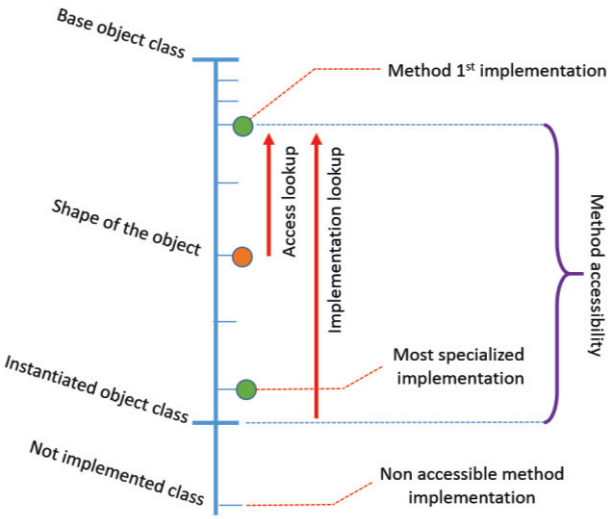


Figure 3: Method lookup

Interface Definition

CMSfwClass provides a mechanism to define sets of method signatures by class interfaces. When a class implements an interface, CMSfwClass toolkit enforces the creation of the methods defined in the interface. This mechanism adds a level of homogenization, flexibility and modularity from the point of view of software engineering. A class implementing an interface provides a first implementation of the methods. The following sub-classes extending from the parent class comply with the interface definition and therefore also implement the interface.

In-memory Objects

While in other languages, data structures are allocated in memory by default, *datapoints* are not. CMSfwClass introduces the possibility of instantiating objects in memory, eliminating unnecessary I/O operations to the persistence layer. This feature has a positive impact in terms of performance for volatile data, but adds extra complexity used for persistent data. An object can be instantiated directly in memory, or can be copied from *datapoints* to memory and vice versa. All class elements are created in a global memory variable. CMSfwClass provides a set of functions to manipulate and add event-driven messaging functionality to in-memory objects.

Object Serialization in Files

The file format to export and import *datapoint* structures in WinCC OA can be complex and hard to manipulate from a text editor. CMSfwClass includes a tool to export object collections into XML formatted files. This can be particularly useful for backing up and recovery tasks.

CMSFWCLASS STANDARD LIBRARY

As mentioned before, CMSfwClass includes a set of basic classes to help developers in the construction of the software. They form the standard CMSfwClass library, and provide with an OO version of common programming features such as:

- Polymorphic object lists and sets.
- Runnable interface for object threading.
- Event-driven object messaging.
- Design pattern solutions for object visualization.

CONCLUSION

The software described in this paper has been built, tested and used in the DCS context for the CMS experiment. The CMS DCS central team identified multiple use cases for CMSfwClass before and after releasing the first version of the toolkit. It has been proven to be an efficient environment for creating complex and high abstraction software architectures with a short development time.

Apart from the implicit benefits of using an OO oriented programming approach, CMSfwClass toolkit helps in the construction of more robust software. The real-time semantic checks and the enforced best practices in the code generation are propagated throughout all software layers.

The toolkit has helped CMS DCS team to balance the amount of time spent in different tasks of the software construction process, spending more time in designing quality architectures rather than writing code.

REFERENCES

[1] Rebecca Wirfs-Brock; Alan McKean, *Object Design: Roles, Responsibilities, and Collaborations.* (Addison-Wesley, 2003).

[2] O. Holme et al. "The JCOP Framework," ICALEPCS'05, Geneva, Switzerland, October 2005, WE2.1-6O.

[3] L. Del Caño et al. "Extending the capabilities of SCADA – Device modelling for the LHC experiments", ICALEPCS'03, Gyeongju, Korea, October 2003, TU212.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

LASER BEAM PROFILING AND FURTHER IMPROVEMENTS TO THE FHI FEL

H. Junkes, W. Schöllkopf, M. Wesemann, FHI, Berlin, Germany

Abstract

The free-electron laser facility at the Fritz-Haber-Institut (FHI FEL) startet regular operation in November 2013. Currently, the FEL provides continuously tunable, powerful pulsed laser radiation at any wavelength between 3.6 and 50 µm covering the mid-infrared (MIR) wavelength range. The first 1½ years of operation have opened up new possibilities for research in gas phase spectroscopy of molecules and clusters, non-linear optics, and surface chemistry at the FHI. The EPICS software framework was chosen to build the control system of this facility. Actions have been taken for laser beam profiling and characterization as well as improvements to the infrastructure like the archiver system. This paper presents the next upgrade step and lessons learned during the first time of operation.

DESIGN OF THE FHI FEL

The FHI FEL includes two oscillator FELs; a MIR branch for wavelengths up to about 50 µm and a far-infrared (FIR) branch for wavelengths from, possibly, about 40 to 400 µm. The MIR FEL has been commissioned and is fully operational whereas the FIR FEL has been projected as a future upgrade. A normal-conducting linear accelerator provides electrons of up to 50 MeV energy with a beam transport system feeding either of the two FEL branches or the electron diagnostics beamline as illustrated in Fig. 1 [1].

Electron Accelerator

The accelerator system was designed and build by Advanced Energy Systems, Inc.. It combines a thermionic electron gun, a sub-harmonic buncher cavity, and two S-band (2.99 GHz) standing-wave copper linacs. The first of the two linacs accelerates the electron bunches to a constant energy of 20 MeV. The second linac accelerates or decelerates the electrons to a final energy between 15 and 50 MeV. The accelerated electron bunches from bunch trains (macro-bunches) containing thousands of micro-bunches at a repetition rate of 1 GHz. The micro-bunch length can be compressed down to a minimum of 1 ps rms by a chicane between the linacs. The macro-bunches are repeated at 5 or 10 Hz [2].

Undulator and Cavity

In the MIR FEL a planar hybrid-magnet undulator is located within the IR cavity. The undulator is 2 m long containing 50 periods of 40 mm. It employs permanent magnets made out of NdFeB. At a minimum gap of nominally 16.5 mm, a maximum root-mean-square undulator parameter K_{rms} of more than 1.6 is reached. This, in combination with the minimum electron energy of 15 MeV, corresponds to a theoretical maximum wavelength of more than 50 µm for the MIR system.

The 5.4 m long MIR FEL cavity is formed by an end mirror and an out-coupling mirror. These are gold-plated copper mirrors of concave spherical shape. The waist of the cavity mode is located at the undulator center. A motorized in-vacuum mirror changer permits to select one out of 5 out-coupling mirrors with different outcoupling-hole diameters of 0.75, 1.0, 1.5, 2.5, and 3.5 mm for optimized performance at different wavelength regions. In addition, the cavity end mirror is mounted on a precision translation stage allowing for fine adjustment of the cavity length with 1 µm repeatability [2].

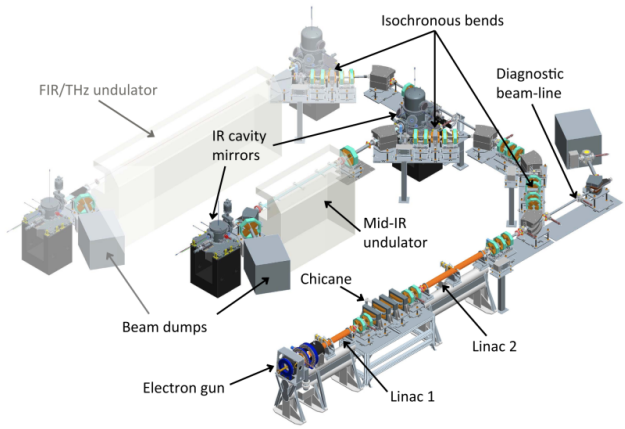


Figure 1: Overview of the FHI FEL installation.

LASER BEAM PROFILING

The IR pulses coupled-out from the FEL cavity pass through the evacuated IR beamline and propagate a distance of 18 m from the FEL vault to the diagnostic station located in the lab building. There the spectrum of the IR radiation is measured by an in vacuum Czerny-Turner grating spectrometer allowing for online monitoring of the FEL spectrum for each individual macro pulse. In addition, various commercial IR detectors are in use to determine the intensity of the FEL-pulses at different levels of sensitivity and temporal resolutions [2].

Fast IR Detector

A very fast IR detector (VIGO PEM-10.6) with sub 250 ps time resolution and an active area of 2 mm x 2 mm is used to detect individual micro-pulses. Figure 2 shows a measurement with this detector used in combination with a fast (4 GHz) oscilloscope without additional electrical amplification. At the standard micro-pulse repetition rate of 1 GHz individual micro-pulses are clearly resolved with a modulation depth of more than 50%. As can be seen in Fig. 2 this detector allows observing intensity changes on the single micro-pulse level.

Pyroelectric Linear Array

A pyroelectric linear array detector with 128 elements and a total width of 1/2" (DIAS Infrared 128LT) is

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

mounted to the spectrometer and allows for monitoring the FEL spectrum in-situ for each individual macro-pulse (Fig. 3).

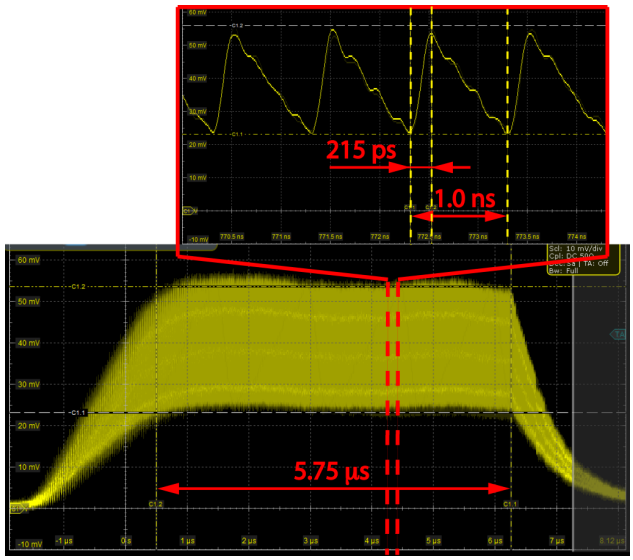


Figure 2: Detection of an FEL macro pulse at 6 µm wavelength with partial resolution of micro pulses.

The interface to the array is realized with a rtd DM6430HR 16bit, 100kHz Analog I/O PC104 module connected to a syslogic NETIPC/6. Debian Linux is used on the CPU board booting from CF card. The EPICS Input Output Controller (IOC) running on this board implements device support for 'pyroArray'. A waveform record holds the profile of the last laser shot.

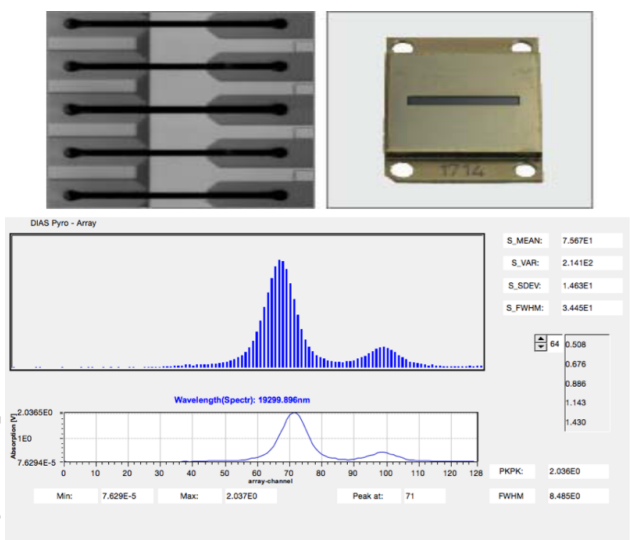


Figure 3: Pyroelectric Linear Array, OPI.

Standard Commercial Detectors

A 46 mm diameter, large area powermeter (Ophir PE50BB) is used on a daily basis to determine macro-pulse energies. Several small area (4.5 mm x 4.5 mm) pyroelectric detectors (Eltec 420M7-0) combined with in-house made amplifiers are in use to monitor the temporal shape of the FEL macro-pulses.

Pyroelectric Cameras

With pyroelectric cameras images of the laser beam profile can be obtained. Displayed in 2D or 3D views, the operator can immediately recognize beam characteristics indicating laser performance. This instantly shows detrimental laser variations. This allows for timely correction and real-time tuning of laser parameters. We use the following two camera systems :

- Spricon Pyrocam IV, 1.06 – 3000 µm, 320 x 320 elements, 80 µm x 80 µm element size, pulse rate up to 1 kHz, external trigger.
- DIAS PyroView 160L compact+, 8 – 14 µm, 160 x 120 elements, fixed measurement frequencies (70 Hz, 35 Hz, 17.5 Hz, …).

Both cameras are read out and controlled by the EPICS areadetector framework. The aravis lib is used, the camera interfaces follows the GenICam standard.

The Spiricon camera fulfils our needs but is expensive. We looked about for alternative systems. The DIAS camera works just in a limited range from 8 to 14 µm and uses fixed internal trigger frequences. Nevertheless we could integrate the DIAS PyroView by triggering the FEL with the internal clock of the camera at 4.375 Hz.

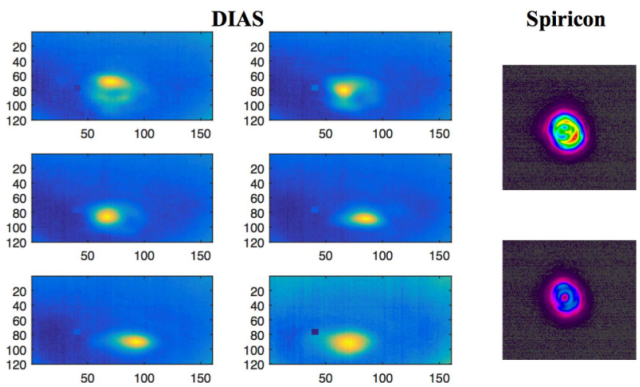


Figure 4: Images taken with areadetector.

Autocorrelation Measurement

In addition, a sensitive liquid-nitrogen cooled mercury-cadmium-telluride (MCT) detector (Judson J15D24) is used for detection of low-intensity signals such as, for instance, higher harmonic signals. Even for the fast IR detector (Fig. 2) the time resolution is by far insufficient to reveal the length and shape of the individual ps-long micro-pulses. For this reason we have installed an autocorrelation setup, shown schematically in Fig. 4, to characterize the micro-pulse shape. FEL pulses are split by a 50:50 ZnSe beam splitter (Edmund Optics). The length of one of the beam paths can be varied by a precision motorized translation stage to adjust the relative temporal delay between the two partial beams. Both beams are separately focused by 6" focal-length mirrors onto the same spot on a CdTe crystal (MaTeck GmbH) with an angle of ≈ 30° between the beams [2].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

At temporal overlap of the pulses from both paths, nonlinear effects in the CdTe crystal lead to the generation of second harmonic (SH) radiation which is emitted in the direction exactly in between the reflected fundamental beams. After spatial filtering to block the fundamental beams, the SH signal is detected with the liquid-nitrogen cooled MCT detector.

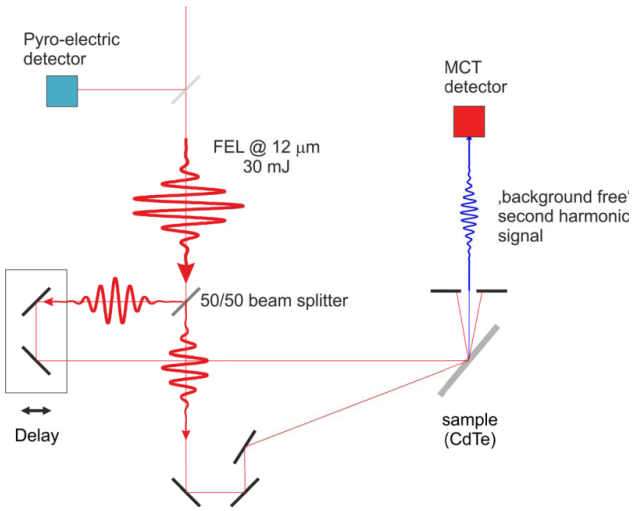


Figure 5: Autocorrelation setup, second harmonic signal.

ARCHIVER APPLIANCE

The previously used EPICS ChannelArchiver was replaced by the ArchiverAppliance developed by Murali Shankar at SLAC. We run two server systems (DELL PowerEdge R730/0599V5) in a cluster configuration. These systems are equipped with 128 GByte main memory, 4 SSDs and two 10 Gbit/s ethernet interfaces. The operating system used is Ubuntu 14.04.3 LTS. The two systems (aa0 and aa1) operate different networks. Within the FEL-LAN (10.0.0.0/23) aa1 is sampling the requested PVs, aa0 archives the data in the FHI-LAN (141.14.128.0/20). See Fig. 6. The previously used ChannelArchiver was only connected to the FEL-LAN. We could easily import the ChannelArchiver XML configuration files into aa1. The FEL-LAN is still the "PV-hotspot". We are planning further experiments at the FHI to operate with EPICS and also want to archive the data obtained there. The access to the archived data at any appliance is ensured by a proxy function implemented in the ArchiverAppliance.

The ArchiverAppliance can use multiple stages of storage. For the short term storage (STS) we use 64 GByte of the main memory (ramdisk) with the granularity of an hour. The mid term storage (MTS) with the granularity of one day is realized with SSD storage. These two storage areas are local to the server systems. For the long term storage (LTS) both systems are using one storage gateway (Crossroad StrongBox) with the granularity "forever". The connection to the StrongBox is realized with 10 Gbit/s ethernet, NFS is used to mount the LTS.

The StrongBox is a networked-attached storage (NAS) appliance. It includes a object-based Amazon S3 interface with an underlaying RESTful API. The system supports file (NFS/CIFS) and object (S3) storage. By fusing disk for nearline storage with Linear Tape File System (LTFS) tape technology for archiving, the system delivers the performance of disk with the economics of tape in one solution. One can expand the nearline capacity by adding external FC/SAS disk arrays. A tape library (IBM TS3200) with two LTO 6 tape drives (SAS) and 48 data cartridge slots was chosen and connected to the StrongBox. This gives us a capacity of more than 240 TByte. The system can be easily expanded with additional/other tape libraries.

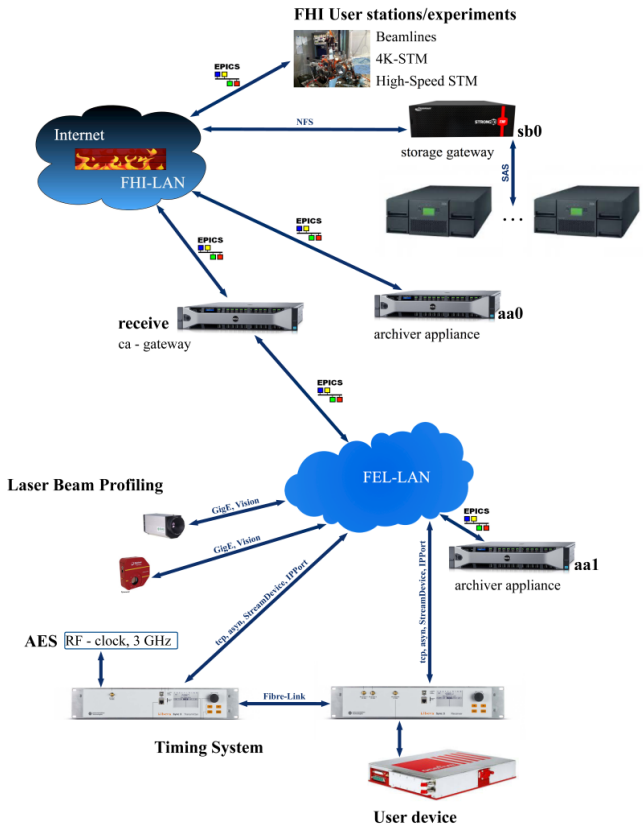


Figure 6: ArchiverAppliance setup.

MOBILE APPLICATION

To allow the operator to move freely in the institute during machine operation an iPad application was developed. The operator in charge can thus intervene at any time in the process and operate the machine (start trigger, stop trigger, adjust LLRF amplitudes and phases). A few important OPIs have been reimplemented (Fig. 7) using the Xcode development framework. These parts are now written in ObjectiveC. The communication takes places via https to a gateway running a standard Web-Server (apache, modPHP). This gateway reads the PVs out of the EPICS-IOCs using channel access (perl) and caches some of the PVs (trend data) into a SQL database on the gateway.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

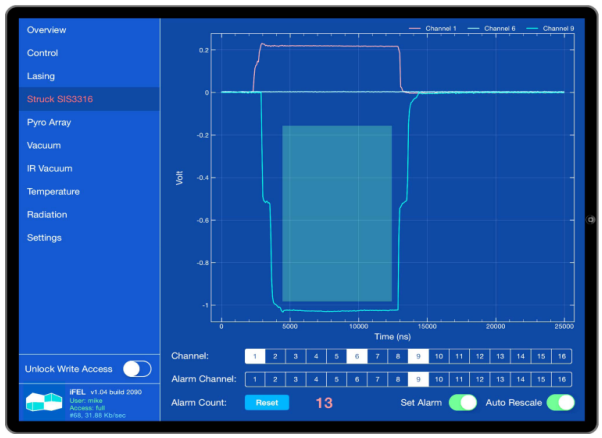


Figure 7: Screenshot iPad App.

TIMING SYSTEM

To enable pump probe experiments (see following section) the clock signal of the accelerator must be delivered jitter free to the user stations. A Libera Sync 3 system was chosen for transmitting the high-performance RF-oscillator to such an beamline end-station. The system assures clock signal distribution with down to femtosecond-level added jitter. The libera system is integrated in the EPICS control system using StreamDevice with asyn over IP. See Fig. 6 [3].

FUTURE MACHINE UPGRADE

Over the last century, linear optical spectroscopy, especially when being performed in the IR, contributed substantially to our understanding of molecular structures. In addition, the usage of ultrashort laser sources has allowed to go further by studying not only structure, but also the dynamics of energy flow between different modes in molecular and condensed phase systems, providing a wealth of additional information. Traditionally, these dynamical processes are studied using pump-probe techniques where two time-delayed optical pulses are applied to the sample, and the dynamics and interactions are derived from the sample response as function of time delay between the two pulses and there frequency or fluence, respectively. Typical single-color experiments allow monitoring the population relaxation of vibrational modes; however, two-color approaches give access to mode-couplings and thus may significantly enhance the attainable microscopic understanding. In particular, both MIR-near-infrared (NIR) and MIR-MIR two-color experiments are interesting, addressing couplings between vibrations and electrons and between different vibrational modes, respectively. Ultimately, it would be intriguing to make use of the high pulse energy delivered by the FEL to also “pump” (drive) molecular processes, reactions or to create new transient states by controlling multiple vibrational motions in a concerted fashion.

Furthermore, it would also be highly interesting to extend the wavelength range accessible by the FEL to longer wavelengths to study lower-frequency vibrations at THz frequencies. The modes of interest here are vibrations involving heavier atoms, for example in clusters, collective motions in bio-molecules and correlated atomic motions in complex solid materials. As these experiments address optical thin samples or require a high field strength they can be only made possible using an FEL.

We propose an extension of the FHI FEL which would enable these new classes of experiments and make the FHI FEL a worldwide unique instrument. We hereby follow two independent directions, to allow for MIR-NIR and MIR-MIR two-color operation, respectively, with the IR pulse wavelength tunable beyond the current limit of 50 µm. MIR-NIR spectroscopy will be enabled by installation of a femtosecond tabletop laser producing pulses at 1 µm synchronized with the FEL. MIR-MIR two-color operation could be achieved by installation of a second, longer wavelength undulator. Specifically, we envision this new undulator to be operational simultaneously with the existing one. This would be possible by implementing a unique design, in which an additional RF cavity is used to feed alternating electron bunches into both undulators, respectively. This is illustrated in Fig. 8. Such an approach is novel and has not been implemented yet at any other FEL facility [4].

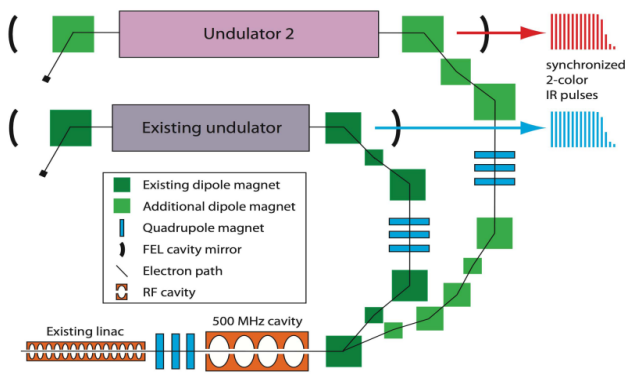


Figure 8: Future upgrade, two-color operation.

ACKNOWLEDGEMENT

Important contribution regarding the machine are provided by Sandy Gewinner (laser engineer) member of the FEL facility staff.

REFERENCES

[1] W. Schöllkopf *et al.*, In: Proc. Of SPIE 9512, 9512L (2015)

[2] W. Schöllkopf *et al.*, The New IR FEL Facility at the Fritz-Haber-Institute in Berlin. In: Proceedings of FEL 2014, Basel, Switzerland, 2014, 629–634, ISBN 978-3-95450-133-5

[3] A. Paarmann, private communication

[4] FHI internal, W. Schöllkopf, “Recent Developments of the FHI Free-Electron Laser Facility”, FHI Report 2015, 18th Meeting of the Fachbeirat, pp.38

Copyright © 2015 CC-BY-3.0 and by the respective authors

REAL-TIME ETHERCAT DRIVER FOR EPICS AND EMBEDDED LINUX AT PAUL SCHERRER INSTITUTE (PSI)

Dragutin Maier-Manojlovic, Paul Scherrer Institute (PSI), Villigen, Switzerland

Abstract

EtherCAT [1] bus and interface are widely used for external module and device control in accelerator environments at PSI, ranging from undulator communication, over basic I/O control to Machine Protection System for the new SwissFEL accelerator. A new combined EPICS/Linux driver has been developed at PSI, to allow for simple and mostly automatic setup of various EtherCAT configurations.

The new driver is capable of automatic scanning of the existing device and module layout, followed by self-configuration and finally autonomous operation of the EtherCAT bus real-time loop. If additional configuration is needed, the driver offers both user- and kernel-space APIs, as well as the command line interface for fast configuration or reading/writing the module entries.

The EtherCAT modules and their data objects (entries) are completely exposed by the driver, with each entry corresponding to a virtual file in the Linux *procfs* file system. This way, any user application can read or write the EtherCAT entries in a simple manner, even without using any of the supplied APIs. Finally, the driver offers EPICS [2] interface with automatic template generation from the scanned EtherCAT configuration. In this paper we describe the structure and techniques used to create the EtherCAT software support package at PSI.

INTRODUCTION

To support external device data acquisition and equipment control both for existing research facilities, such as Swiss Light Source (SLS), and for facilities being built at the time this text was written, like Swiss Free Electron Laser (SwissFEL) [3], an EtherCAT software interface was needed at PSI. Unfortunately, none of the existing commercial and non-commercial solutions we have reviewed and tested was able to cover and satisfy all of the requirements for the EtherCAT support.

General requirements were divided in two broad categories – the first was the full support for EPICS control system and the complete range of standard EPICS record types currently available, with the possibility for flexible addressing of EtherCAT modules and entries. The second requirement was to have a system that can provide the EtherCAT interface for common applications, both applications running locally, normally on the Ioxos IFC 1210 VME Board (equipped with the PowerPC P2020 CPU) and remotely, on a standard desktop PC or mobile device running any operating system capable of supporting network-based file systems, such as Linux, UNIX, Windows, MacOS, FreeBSD and others.

CONCEPTS

Providing support for such a wide range of applications in a single package presented a problem, since not every requirement or possible scenario for usage could have been satisfied with a single piece of software.

EPICS control system support requires its own type of dedicated device support driver. Unlike its kernel counterparts, EPICS driver has to run in Linux userspace, since EPICS system itself is a userspace application. Aside from EPICS, the system has to support other types of applications, both local and remote.

Local applications can be both userspace and kernelspace applications, which in turns mean at least two separate local APIs had to be created. Remote applications, on the other hand needed a generalized way to access EtherCAT data regardless of the operating system used.

EtherCAT Data Addressing

To describe an address of a given EtherCAT data entry, several variables have to be included: EtherCAT Master number (since there can be multiple masters running on the same host), Domain Nr. (domain is an arbitrary, user-defined collections of PDO (Process Data Object) entries sharing the same domain buffer memory and TCP exchange frame rate), Slave Nr. (Slave is another name for an EtherCAT Module), Synchronization Manager Nr. (SyncManager or SMs group objects by their exchange direction (input/output) or other criteria), Process Data Object Nr. (PDOs group entries by some arbitrary purpose defined by the Module producer) or Process Data Object Entries (PDO Entries or PDOE hold the actual data).

The user should be able to easily describe which data entry (or entries) should be addressed, in a consistent yet simple manner. To solve this problem, we have devised a new addressing schema for EtherCAT data:

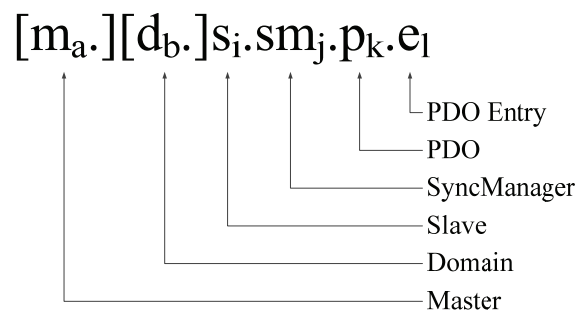


Figure 1: New schema for EtherCAT data addressing.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Master (m_a) and domain (d_b) can be omitted when a=0 or b=0, i.e. for the default master and domain. Similarly, the PDO entry (e_l), PDO (p_k) or SyncManager (sm_j) can be omitted as well (in that order), when the user wants to address multiple entries included in a larger parent container, instead of a single entry. For example, *s1.sm4.p0.e1* would be the address of the "entry 1 of PDO 0 of SyncManager 4 of slave 1", whereas the string *s1.sm4* would mean "all entries contained in all PDOs of SyncManager 4 of slave 1".

To increase flexibility, we have extended the addressing to include various modifiers, hence not rigidly connecting an EPICS record to a single entry of the same type. The possible modifiers or addressing modes (as of v2.0.6) are:

- [.o<offset>] - forced offset (in bytes), allows shifting of the starting address of the PDO entry in buffer
- [.b<bitnr>] – forced bit extraction, allows extraction of single bits from any larger PDO entry data types
- [.r<domregnr>] – domain register addressing, replaces address modifiers s, sm, p and e, using relative entry addressing inside a domain instead
- [.lr<entryrelnr>] – local register addressing, replaces any (group) of the address modifiers s, sm and p, allowing for local relative addressing of all entries inside a slave, inside a SyncManager, or inside a PDO regardless of their actual parent container or containers
- [.l<length>] – length modifier, in bytes. Used primarily to define the length of stringin/stringout EPICS records
- [t<type>] or [t=<type>] provides means for forced typecasting or type override, changing the default type of the data entry when applied. Many types are provided, from *int/uint* (8-, 16-, 32-bits), *float*, *double*, *BCD*, etc.

It is worth noting that the addressing modes or modifiers listed above can be freely mixed as needed – for example, the address s1.sm4.p0.e1.b4 would mean "extract the bit 4 of the 32-bit entry s1.sm4.p0.e1", and the address s1.sm4.p0.e1.o2.b4 would mean "extract the bit 4 of the 32-bit entry s1.sm4.p0.e1, but shifted by 2 bytes (.o2)", which would effectively extract the bit 20 (2*8+4=20). Similarly, the address r45.o2 with the modifier t=float would mean "domain register 45, shifted by 2 bytes, typecasted to float".

EPICS SUPPORT

Since PSI almost exclusively uses EPICS control system for its accelerators, integrating EPICS support was a top priority. For the system employing EtherCAT components, the EPICS Core is running on the Ioxos IFC 1210 Boards [4], equipped with two separate Ethernet interfaces, a PowerPC P2020 CPU and the VME Bus backplane. The operating system installed is a Linux with the appropriate PREMPT-RT patch.

Since EPICS has its own interface for device drivers, a special EPICS userspace driver had to be developed, using high priority real-time threads for the control loop. Without the real-time capabilities provided by the PREMPT-RT Linux, timing and execution of the control loop would not be reliable and hence not real-time capable.

For the EPICS support, the timing control loop is maintained by the EPICS userspace device driver. EPICS records are registering the entries they are "interested in" at the IOC boot time, and the driver then carries the exchange between the records and the EtherCAT control loop.

Additionally, EPICS support has to provide both "normal" reading and writing, not synchronized with the EtherCAT control loop, and I/O Interrupt mode, to allow each Ethernet packet to trigger an interrupt, at which point the new values can be read and written to the buffer. This was accomplished with double-buffering technique between EPICS and the driver.

Another problem to solve was the fact that EtherCAT modules are not always required to accept the write values – a write request may, for example, fail for a number of reasons – and that means that the Ethernet TCP packet on a return trip may contain write values which differ from the content of the write buffer.

This means that not only the refreshed read values, but also the write values has to be transferred back to the write buffer at the end of every cycle. Yet, the newly received write values, unlike new read values, cannot be simply copied over the old values in the buffer, since that would effectively overwrite the new write request values which were already accepted since the last cycle. To solve this, a multithreaded double-buffering with the write-mask for write requests was implemented (see Figure 2).

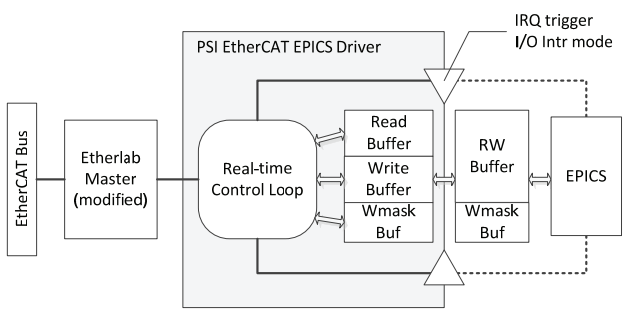


Figure 2: EPICS driver structure.

All EPICS records use a single driver type (DTYP) called *ecat2*. Scan rates for records can be set to any valid EPICS scan rate, including *Passive* and *I/O Intr*.

All of the extended addressing modes and modifiers, including typecasting, can be used in EPICS as well. Almost all combinations are allowed - even in EPICS, with its rigidly defined record types, it is possible to achieve high level of flexibility.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

For example, it is possible to extract a single bit from a *mbbi* record without additional *calc* records, or to have a complete bit field extracted from a record of any length. Type override and other modifiers can be used for all record types, including array-type records (*aai/aao*). Length modifier (.l<length>) is used to define the length of the string for string-type recods (*stringin/stringout*).

Additionally, several special types of records are provided. These records signify status of the EtherCAT Master, aggregate status of the slaves (modules), EtherCAT network link status and status of each slave (module) – preoperational, operational, error, etc.

GENERAL SUPPORT

To support the local and remote applications wishing to connect to and use the EtherCAT hardware, the second part of the driver package was created. We have developed three different subsystems to allow application developers a highly flexible way to access the entries and other data:

- Kernelspace API
- Userspace API
- Procfs-based tree(s)

All of the above are included in the Linux-based PSI EtherCAT kernel device driver and supporting tools and libraries. The driver provides support automatic scanning of the EtherCAT bus and automatic configuring of domains and found entries, and manual configuration for some or all modules as well.

Additionally, the driver automatically constructs and maintains procfs trees throughout its operation, and takes care of triple-buffering and write-masking process needed for data exchange with the client applications. Description of the each of the access modes is presented below.

Kernelspace API

Kernelspace API (kAPI, Figure 3) is a set of functions providing the easy access to driver control loop parameters and EtherCAT data entries.

The driver provides an internal real-time control loop for buffering and exchange of TCP packets over Ethernet, Timing of the control loop is based on the host high resolution timers, but can be driven by an external source as well, such as a timing system input.

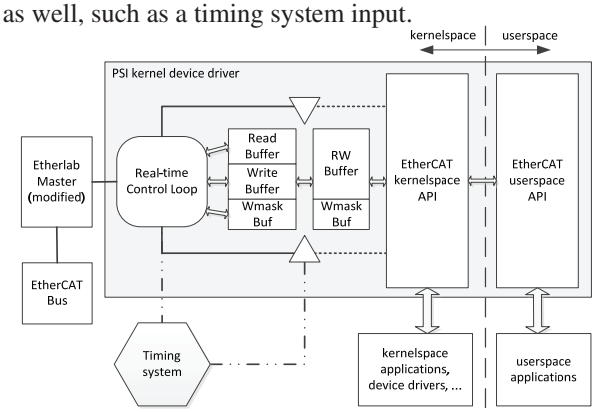


Figure 3: PSI EtherCAT driver structure.

Multiple kernelspace applications and/or drivers can use the API, however, the RT-priorities of various application kernel threads has to be set carefully, in order to allow both the driver and the rest of the kernel to run properly.

Userspace API

Userspace API (uAPI, Figure 3) is a set of functions providing (almost) the same functionality found in the kAPI. The functions library can be used statically or dynamically with userspace applications as needed.

The only difference is that there is no possibility for external timing input for the control loop (since the control loop is located in the kernelspace part of the driver), only timing triggering for data acquisition (or delivery) can be used.

Procfs Trees

To allow local and remote applications to access the EtherCAT data, but without the need for an API or a dedicated remote server and client, we have developed the concept of procfs trees. Procfs trees are a series of directories and "files" constructed on-the-fly by the drivers in the Linux host proc file system.

Each directory represents some kind of a parent container, such as a slave, a syncmanager, a PDO, a domain or a master (Figure 4). Each file in these directories represents either a direct representation of a EtherCAT PDO entry, or a utility file representing the data about the system or about the containers present.

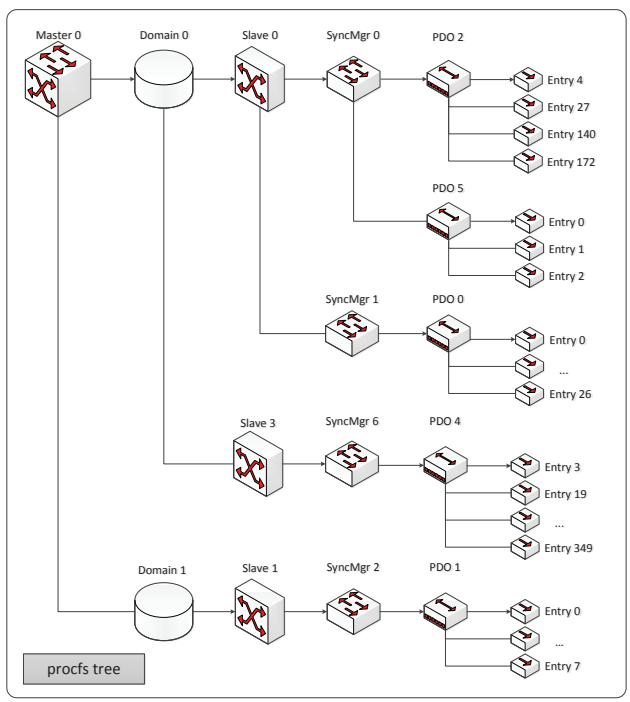


Figure 4: procfs tree general structure.

There is also a special *cmd* entry provided in the *procfs tree*, allowing a CLI or application to interactively talk to the driver, sending commands (for example, add entry,

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

add PDO, add slave entries, list data, etc.). Also, entry data can be read or changed using the CLI as well.

The complete procfs tree directory/file structure is updated automatically each time something changes in the system – during bus scans, when an entry is added to a domain and similar actions.

The files in the structure can be read from or written to – this allows any application programmer to easily access the EtherCAT data without having to meddle in the API programming at all.

EXTENSIONS AND UTILITIES

The PSI EtherCAT support package offers several extensions and tools. The most important ones are described below.

Slave-to-Slave Communication

It is often the case that the data on the EtherCAT bus has to be transferred from one EtherCAT device or module to another, preferably in real-time. For this kind of communication, two different directions of transfer can be observed, *upstream* and *downstream*.

Upstream slave-to-slave communication is transfer of data from a module further away on the EtherCAT bus from the master to a module closer to the master. Downstream communication is the transfer from a closer module to one further "down the stream" from the master. The stream in this case represents the path an EtherCAT TCP packet is travelling, and its direction remains constant as long as there are no physical changes in the bus configuration and modules present.

From real time point of view, this communication is highly deterministic, yet not identical – downstream communication (send on one module, receive on another) can, theoretically, be done in the same bus cycle, hence costing exactly zero bus cycles to execute. Upstream communication, due to the way TCP packets are handled by the EtherCAT, will take exactly one bus cycle to complete.

We have decided to implement the slave-to-slave communication (*sts*) with constant cost of completion, in this case, exactly one bus cycle for both upstream and downstream communication requests.

In EPICS, sts-communication transaction requests can be inserted as follows:

ecat2sts <source> <destination>

For example:
ecat2sts r8 r0
ecat2sts r2.b0 r0.b6
ecat2sts s2.sm0.p1.e0 s1.sm0.p1.e0
ecat2sts s3.sm3.p0.e10.b3 s4.sm2.p1.e0.b7

As can be seen in examples above, any valid addressing mode and/or modifier can be used for source and destination. API access is done by calling a function to register a transaction request, but the addressing remains the same.

Support for Programmable Modules

The PSI EtherCAT drivers and utilities also support setting up and live programming of programmable EtherCAT modules and devices, such as, for example, EtherCAT network bridges (EL6692, EL6695), motor controllers, and so on.

From EPICS, any module can be programmed by using *ecat2cfgslave* set of commands, for example:

ecat2cfgslave sm <arguments…> - configures one Sync Manager for the given slave.
ecat2cfgslave sm_clear_pdos <arguments…> - clears (i.e. deletes) all PDOs for a given Sync Manager (SM)
ecat2cfgslave sm_add_pdo <arguments…> - adds a PDO with index pdoindex to a Sync Manager.
ecat2cfgslave pdo_clear_entries <arguments…> - clears (i.e. deletes) all PDO entries associated with the given PDO.
ecat2cfgslave pdo_add_entry <arguments…> - creates a new PDO entry and associates it with the given PDO

Network bridges even have their own, simplified commands for programming entries:

ecat2cfgEL6692 <netbridge_nr> in/out <numberofbits>

CONCLUSION

In this paper, we have presented the PSI EtherCAT software support package and described its components. The system is already successfully used at PSI in the last several months.

As is usual with such systems, it is to be expected that changes will be made to this package in the future to accommodate needs and new requirements of expanding number of users of the system, the existing features will be extended and streamlined and the new features and components will be added.

REFERENCES

[1] Beckhoff.de website: http://www.beckhoff.de/
[2] EPICS, Experimental Physics and Industrial Control System, website: http://www.aps.anl.gov/epics/
[3] PSI.ch website: http://www.psi.ch/media/swissfel
[4] Ioxos Technologies, *IFC 1210 – P2020 Intelligent FPGA Controller*, website: http://www.ioxos.ch/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PERSONNEL PROTECTION SYSTEM UPGRADE FOR THE LCLS ELECTRON BEAM LINAC*

M. Cyterski#, E. Chin,
SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA

Abstract

As facilities age and evolve, constant effort is needed in upgrading control system infrastructure. This applies to all aspects of an accelerator facility. Portions of the Personnel Protection System of the Linac Coherent Light Source are still relying on a 50 year old relay based Safety System. This presents a substantial risk to the facility's ability to reliably perform its mission. An upgrade is underway to modernize these systems using Siemens S7-300 Safety PLCs and Pilz PNOZMulti programmable controllers. The upgrade will be rolled out over multiple years requiring the implementation to be fully compatible with adjacent legacy system while setting the foundation for the new generation system. The solution is a modularized safety system which can be deployed in a short time (1 month) while being flexible enough to adapt to the evolving needs over the next 20 years. Once fully deployed, the upgraded PPS System will provide not only greater availability to users, but also a higher level of Personnel Safety than previously provided.

BACKGROUND

The 3 mile Linac has been in operation at SLAC National Accelerator Lab for more than 50 years. In that time many upgrades have been made to support successful operation of the facility into the future. In 2014, a multiyear project was begun to upgrade the Personnel Protection System for the section of Linac used for operation of the Linac Coherent Light Source (LCLS).

The Personnel Protection System (PPS) at SLAC is composed of a system of interlocks whose function is to prevent personnel from being exposed to radiation within the accelerator enclosures.[1] In the Linac this is done by controlling the ability of the Klystrons to produce RF for the accelerator waveguides. The SLAC Linac is composed of 30-100meter long sections. Pairs of these sectors are configured as 15 stand-alone PPS zone.

The system currently used is composed of telephone relays and has operated safely for many years. In order to continue to support the high level of facility uptime demanded by LCLS, a modern replacement is needed. The LCLS Linac is composed of the final 10 sectors. The new system will provide a substantial increase in diagnostic information available to the operations staff as well as new functional modes. These modes will reduce facility downtime following personnel access to the accelerator housing.

*Work is supported by the U.S. Department of Energy, Office of Science under Contract DE-AC02-76SF00515.
#cyterski@slac.stanford.edu

Completing an upgrade of this size on an operational facility presents an additional level of difficulty. Annual maintenance down time is on the order of two months, with half of this time needed for recertification of all PPS installations. This only leaves four weeks to complete the removal, installation, and commissioning of a new system.

REQUIREMENTS

In addition to the standard requirements of a Personnel Safety System, this upgrade must meet two additional project specifics requirements. [2][3]

The first is the need for a personnel controlled access. Currently any access into the linear accelerator requires the operations staff to manually go into the tunnel and verify that no one is inside before closing the facility for beam operation. Depending on the number of sector pairs accessed this can take a substantial amount of time. Controlled Access will allow for trained personnel to enter the accelerator housing using a safety token and return the token once complete.[4] This eliminates the need for an operations search.

Secondly, due to the installation window time constraints, the upgrade must be deployed in phases. Each phase will be staged and modularly assembled while the facility is in operation and then deployed during a maintenance downtime.

DESIGN

The new system is composed of 3 separate PPS zones instead of the 5 sector pairs it's replacing. Each zone is deployed in a different annual maintenance down time. The three stages are listed in Table 1.

Table 1: Installation Stages

Zone	Controlled Access	Installation
Sectors 24 & 25	No	September 2014
Sectors 21-23	Sectors 21-25	September 2015
Sectors 26-30	Sectors 21-30	TBD 2016

Personnel Access to the accelerator regions is handled by three distinct access states at SLAC. No Access is the condition in which beam operation is allowed as long as the interlocks are met. Permitted Access allows free access to individuals to the PPS zone. This always requires an operations search of the region prior to machine recovery. Controlled Access allows for access without the need to search the PPS zone.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

There are two methods to access the Linac housing in sectors 21 through 30, a vertical ladder in each sector which requires fall protection equipment to be used and a stairwell located at Sector 24. Because of the need for additional equipment and training to use the vertical ladders, only the sector 24 stairwell is instrumented as a PPS Entry location. This means that a controlled access scheme requires the ability to move between the three zones without violating individual zone security and requiring an operations search of the area. The ladder located at each sector can still be used as entry point under permitted access conditions. Each stage is a stand-alone installation; the number of signals shared between zones is minimized to facilitate the staged deployment of the system. Controlled access was not fully implemented until the second stage at which point it provided an operational benefit. Controlled Access allows for the boundaries between the three zone to be opened without violating zone security as long as each zone is in controlled access and is secure

For each zone the PLC architecture used to implement the system is based on three PLCs as shown in Figure 1. Two Pilz PNOZMulti controllers are implemented as the safety interlock controllers. These handle all the safety critical sensors,

Figure 1: PLC architecture.

Emergency Off hardware and permits to radiation hazards. These communicate with a Siemens S7-315F processor over Profinet. This PLC is responsible for all non-safety critical functionality. This includes interfacing with the Experimental Physics and Industrial Control System (EPICS), controlling indicators throughout the region, and other non-safety critical functionality. EPICS is used to control and monitor all aspects of the PPS systems at SLAC. The individual status of all inputs as well as the ability to release safety tokens and grant access is done through EPICS. As an additional level of safety all PPS installations include a hardwired enable which is used at the PLC installation to gate all commands received over EPICS. If this signal is not present then the system ignores the control system request.

INSTALLATION

In order to meet the installation schedule, a large amount of staging and pre-assembly work was required.

New terminal cabinets and copper trunking were mounted and routed to the rack locations during LCLS Operation. These trunks were terminated in the rack once the rack hardware was upgraded. Because existing racks were repurposed, it wasn't possible to pre-wire a new rack. Instead an assembly jig was constructed to allow the rack hardware to be prewired and then installed as a single unit in the repurposed rack. The upgraded rack assembly can be seen in Figure 2. This single rack installation replaces 4 racks in the case of Stage 1 and 6 racks in the case of Stage 2. At the completion of Stage 3 it is expected that 2 racks will replace 10 racks. Once the prewired assembly is installed in the rack, the copper trunks are terminated.

Figure 2: New Installation in repurposed racks.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The existing zone wiring was removed and the conduit, inspected. New wire was pulled in the existing conduit and connected to the previously mounted terminal boxes. Field hardware was replaced at each entry way and throughout the tunnels. The sector 24 entry way was upgraded to support its new function as a controlled access point. Sector 24 entry before the upgrade and after can be seen in Figure 3.

Figure 3: Sector 24 entry way.

The legacy system uses redundant constant current loops which run the length of the Linac in order to sum the state of the individual subzones. Each zone is capable of breaking the continuity of these loops. There is a global PPS system which looks at these loops as well as other inputs and ultimately issues the hazard permits to the RF. The interface to this system is unchanged; the loops pass through dry-contact outputs on the respective Pilz Safety Processors. This allows for the upgraded and legacy installation to operate side-by-side without any intermediate solutions.

CONCLUSION

While completing the first two stages of this upgrade, a strong dependence on pre-planning and execution was necessary to deploy a large scale upgrade in a comparatively short amount of time. This project would not have been achievable without the strategies employed. When the third stage is executed, the project will benefit from the lessons learned to date and successful completion of a substantially larger geographical upgrade.

REFERENCES

[1] *Radiation Safety System Technical*, Basis Document SLAC-I-720-0A05Z-002-R004, SLAC National Accelerator Lab (2010).

[2] Linac Sectors 24-25 PPS Requirements Document CD-SS-PPS-02-11-18R1, Internal Document, SLAC National Accelerator Lab (July 2014).

[3] Linac Sectors 21-23 System Requirements Document CD-SS-PPS-02-11-21R1, Internal Document, SLAC National Accelerator Lab (June 2015).

[4] Linac Sectors 21-29 Controlled Access System Requirements Document CD-SS-PPS-02-11-22R1, Internal Document, SLAC National Accelerator Lab (June 2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

UPGRADE OF THE CONTROL AND INTERLOCK SYSTEMS FOR THE MAGNET POWER SUPPLIES IN THE T2K PRIMARY BEAMLINE

Kazuo Nakayoshi, Ken Sakashita and Yoshiaki Fujii
High Energy Accelerator Research Organization (KEK), Tsukuba, Ibaraki, Japan

Abstract

T2K is a long-baseline neutrino oscillation experiment at J-PARC in Japan. A high intensity neutrino/antineutrino beam is generated and propagates 295km to Super-Kamiokande. The high intensity proton beam which reached 350 kW in May 2015, is extracted from the Main Ring synchrotron and guided through a primary proton beamline to a graphite target using normal-conducting (NC) magnets and super-conducting combined-function magnets. In October 2014, we replaced all of the power supplies (PSs) for the NC magnets with newly developed PSs. We also developed a new control system based on the Experimental Physics and Industrial Control System (EPICS) and PLC, putting emphasis on the safe operation of power supplies, and integrated it into the existing interlock system. Consequently the latency time for the interlock system was improved. We report the actual implementation and operation results of these developments.

INTRODUCTION

The T2K (Tokai-to-Kamioka) experiment [1] is a long-baseline neutrino oscillation experiment at J-PARC (Japan Proton Accelerator Research Complex) in Japan. A high intensity neutrinos/anti-neutrino beam is produced, and propagates 295 km from J-PARC to Super-Kamiokande.

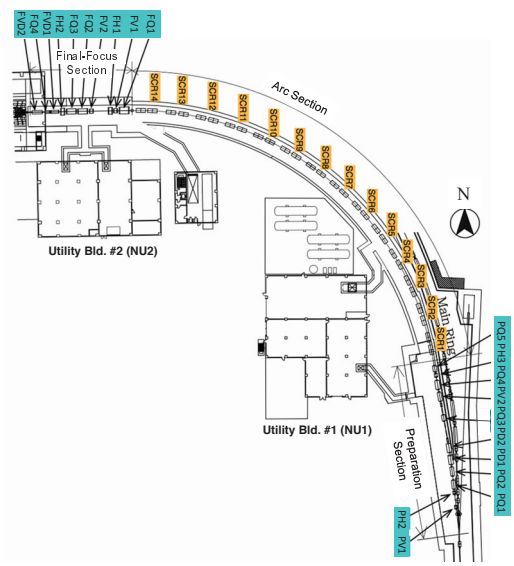


Figure 1: T2K primary beamline.

The T2K neutrino experimental facility is composed of primary/secondary beamlines and a near detector (ND280). The high intensity proton beam which reached 350 kW in May 2015, is extracted from the Main Ring synchrotron (MR) and guided through the neutrino primary beamline to a graphite target. Figure 1 shows the T2K primary beamline, consisting of the preparation (PREP), arc and the final focusing (FF) sections. Fourteen doublets of super-conducting combined function magnets [2] are located in the ARC section and bend the beam towards Kamioka. In the PREP section, the extracted proton beam is tuned with a series of 11 normal-conducting (NC) magnets. Ten NC magnets in the FF section guide and focus the beam onto the target. The power supplies (PSs) for the PREP section are located at the utility building-1 (NU1) and those for the FF section are located at the utility building-2 (NU2).

Some critical primary beamline issues are:

- A single failed beam shot, caused by a trip of the NC PSs, for example, results in serious damage of beamline equipments.
- The NC magnet power supplies were made mostly in 80's and required increasing effort for maintenance.

In order to solve these problems, we developed new power supplies for the NC magnets with a power supply company, and replaced all of the PSs in the summer of 2014.

NEW POWER SUPPLIES

The requirements for the new PSs for the NC magnets were (1) improvement of the safety interlock, (2) precise and stable operation, (3) improvement of maintenance, (4) upgrade of the control system and (5) downsizing. We report on each requirement and its actual implementation.

Table 1 shows the types of PS, corresponding to each magnet, its rated current, and its unit count. Figure 3 shows pictures of the new PSs.

Table 1: New power supplies for the normal conducting magnets in the T2K primary beamline.

Magnet type	DC OUT (A) / (V)	Converter type	Current stability(A)	# of units
Dipole	1500 / 100	chopper	0.1	4
Quadrupole	1000 / 100	chopper	0.1	9
Steering I	± 400 / ± 40	chopper	0.05	1
Steering II	± 200 / ± 20	switching	0.05	2
Steering III	± 100 / ± 10	switching	0.05	5

The output current stability is 0.1 A for the dipole and quadrupole magnet PSs and 0.05 A for the steering magnet PSs. The stability is superior to the previous PSs and the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

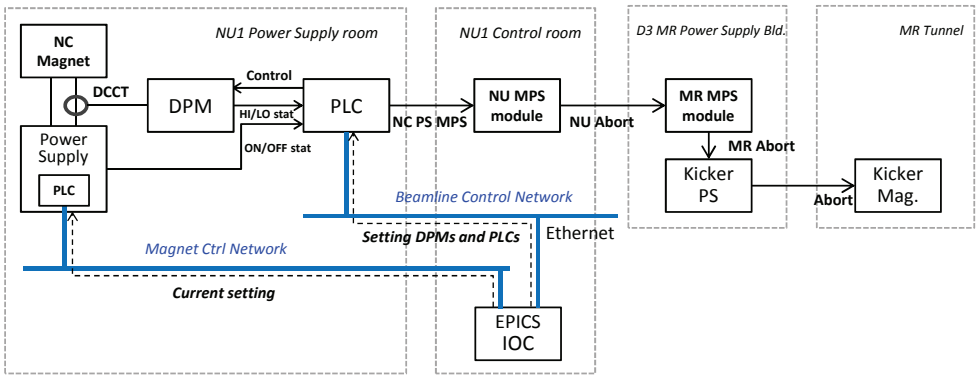


Figure 2: Schematic of the control system for the new power supplies for the normal-conducting magnets.

Figure 3: Pictures of the new power supplies for the normal-conducting magnets. The left photo shows a power supply for a dipole magnet and the right shows four steering magnet power supplies installed in a rack.

output current ripple is reduced. There are two DCCTs in each PS; one is for feedback control and the other is for current monitoring used in an interlock system. Each PS has a LCD touch panel on the front for local operation. For safer operation, control of the PS is password-protected. We succeeded in saving space (about -67%) by installing four steering magnet PSs in a single rack (see Figure 3).

IMPROVEMENT OF THE CONTROL SYSTEM

The control system for the old PSs was not designed to support EPICS [3]. Therefore, we chose to use a relational database as an interface between the old control system and EPICS. A relational database (MySQL) that includes the status of each magnet was prepared. We could access the magnet data via MySQL tables. For the new PSs, we changed the framework of the control system to a less complex configuration: we got rid of our unique control system and adopted standard Programmable Logic Controllers(PLCs) with EPICS. We built two EPICS IOCs, one for PSs at NU1 and another for those at NU2. Figure 2 shows a schematic of the control system for the new PSs at NU1. The IOC communicates with a PLC [4] built into each PS over Ethernet. We separated the new PS's control network from the existing beamline control network in order to separate it from EPICS broadcasts. The new NC PS's output current is monitored by the existing interlock system for current fluctuations as described in the next section.

IMPROVEMENT OF THE INTERLOCK LATENCY TIME

We developed a current-fluctuation interlock system for old NC PSs using digital panel meters (DPMs) in 2012 [5]. The DPMs continuously sample and digitize the DCCT output voltage of the PSs and determine whether it has stayed within a preset range. HI/LO status outputs of the DPM are connected to PLC input-modules. The input-modules aggregate the HI/LO status from all DPMs and an output-module outputs MPS (Machine Protection System) signals. The MPS is an interlock to stop the MR beam and protect beamline equipment from the high intensity beam or a cascade of equipment failures. The DPM is able to change the number of input voltages used in a moving average to remove the fast component of electrical noise.

Latency Time of the Old PSs

We measured the latency time of the DPM and PLC for the old PSs while changing the number of voltages used in the moving average. Figure 4 shows the results of the measurement. The latency of the PLC was measured to be 14 msec, which was nearly constant, as expected. The total latency of the DPM increases in proportion to the number of every used in the moving average. The ripple on the output current of the old PSs was large. Therefore we increased the number of voltages used in the to use as a digital filter. For 100 samples used in the moving average latency time of ~100 ms was expected at the DPM. In order to reduce the latency time, a smaller number of samples was required.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

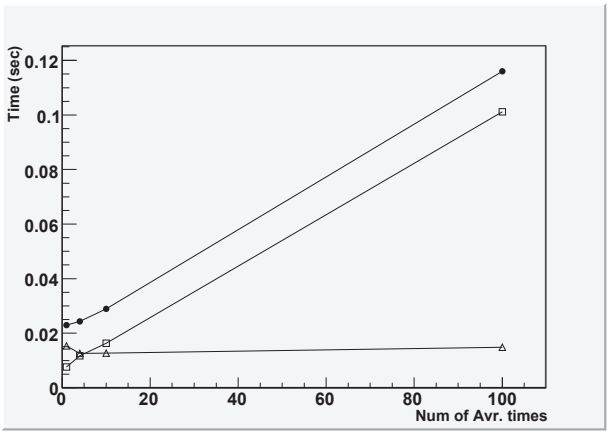


Figure 4: The interlock latency time vs. number of averaged voltage samples times on a digital panel meter (DPM). The open triangle shows the latency time of the PLC. The open square shows the latency time of the DPM. The closed circle shows the sum of the PLC and DPM latency times.

Latency Time of New PSs

In the new PS, we use one of two DCCTs in each PS for the interlock system. The ripple on the output current of the new PSs was small so that only one sample was required in the DMP moving average for each new PS. We measured the interlock latency time of the DPM and PLC while changing the threshold current value of the DPM for the PS for PD1, one of the dipole magnets.

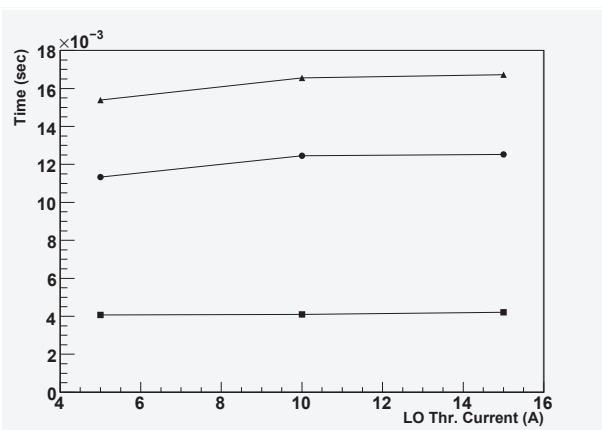


Figure 5: The interlock latency time vs. LO threshold current on a digital panel meter (DPM). The closed square shows the latency time of the DPM. The closed circle shows the latency time of the PLC. The closed triangle shows the sum of the PLC and DPM latency times.

Figure 5 shows the results of the measurement. The results show that the latency time of the PLC is almost the same as the previous measurement. The latency time of the DPM was about 4 ms due to no averaging required at the DPM. The total latency time of the DPM and PLC is about 16 ms with a ±5 A threshold window, the settings used during beam operation. An additional latency time from the PLC to the fast extraction kicker magnet in the MR of $\tilde{0}$.07ms was measured, but time is negligibly small compared to the latency of the DPM and PLC. We drastically reduced the latency time of the interlock system of the NC PSs, reducing the risk for damage of beamline equipment by high intensity beam.

SUMMARY

We have developed new NC PSs in collaboration with a power supply company, and upgraded the control system for safety and stable operation of a high intensity proton beam. We integrated the new NC PSs into the existing interlock system which checks for NC PS current fluctuation. The latency time of the interlock system was drastically reduced. We developed an EPICS-based control system for the new NC PSs which is simpler and less complicated than old one.

ACKNOWLEDGMENT

The authors would like to acknowledge the members of the J-PARC neutrino facility.

REFERENCES

[1] K. Abe, *et. al.*, "The T2K experiment", Nucl. Instr. and Meth. **A659**, Issue 1, 11 Dec. 2011, pp106-135.

[2] T. Ogitsu, *et. al.*, "Superconducting combined function magnet system for J-PARC neutrino experiment," Applied Superconductivity, IEEE Trans. , vol.15, no.2, pp.1175,1180, (2005).

[3] Experimental Physics and Industrial Control System(EPICS) home page. `http://www.aps.anl.gov/epics/`

[4] J. Odagiri, et al. ,"EPICS Device/Driver Support Modules for Network-based Intelligent Controllers", ICALEPCS2003, Gyeongju, Korea (2003).

[5] K. Nakayoshi, K. Sakashita, Y. Fujii, "Improvements in the T2K Primary Beamline Control System", ICALEPCS2013, San Francisco, US (2013).

[6] Y. Suzuki, "Beam line control and database", Proceedings of PCaPAC 2005, Hayama, Japan (2005) .

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INSTALLATION OF A HOT-SWAPPABLE SPARE INJECTOR LASER SYSTEM FOR THE SLAC LINAC COHERENT LIGHT SOURCE*

S. Alverson#, G. Brown, F.-J. Decker, S. Gilevich, S. Vetter, C. Zimmer
SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA

Abstract

LCLS is a facility for generation of very short duration, highly intense x-ray pulses which requires an extremely reliable photocathode electron source. In order to maintain high uptime (>95%) for the experimenters, operations rely on a maintenance program for active laser components as well as on built-in redundancy in case of failure. To accomplish this, a duplicate laser system was installed, allowing for quick swap between the active system and the spare in the event of a malfunction or for planned maintenance. As an added bonus, this redundant system provides additional possibilities for science as both laser systems can also be run to the cathode simultaneously to create multiple particle bunches. Diagnostics were put in place to maintain both spatial and temporal overlap and allow for the fast switching between systems by operations personnel while still remaining within the safety envelope. This was done for both the primary UV drive laser as well as the secondary IR "heater" laser. This paper describes the installation challenges and design architecture for this backup laser system.

BACKGROUND

The Linac Coherent Light Source (LCLS) at SLAC National Accelerator Laboratory is a high intensity, extremely tuneable X-Ray Free Electron Laser Facility (XFEL). The light this facility uses for experiments is generated by a 1 km electron accelerator with a Photocathode source [1]. Light from a high power 253 nm wavelength (UV) laser (hereon referred to as the Drive Laser) is pulsed at up to 120 Hz on a copper cathode to emit electrons which are then accelerated by RF through a series of cavities.

In order to counteract unintended micro-bunching of the electrons in each pulse, they are also sent through a small chicane with an alternating series of permanent magnets (undulator) where they interact with a second 760 nm (IR) laser called the Heater Laser. To save the cost of constructing a completely separate system for the Heater Laser, it is actually formed from the unconverted light of the Drive Laser during the Tripling process used to create the UV light.

Loss of either of these laser systems due to an equipment failure would result in the unavailability of beam for the end users performing their experiment that week. Two exact duplicate laser systems (named Coherent 1 and Coherent 2) were developed so that there would always a spare available should something break or need to be serviced on the active line. Unfortunately, due to the high maintenance required to keep these lasers tuned up appropriately and the high sensitivity of the XFEL produced downstream to small changes in the characteristics of the injector laser systems, it could take many hours for a qualified laser specialist to switch to the spare, align it on the cathode, and tune it to match the old laser. This issue was compounded outside of the normal Day Shift hours when the specialist would have to drive in to site from home to perform the work.

REMOTE SWITCHING

In order to reduce the high downtime cost incurred by failures or scheduled maintenance tasks on the active laser system, it was desirable to implement a method by which Control Room staff could remotely switch between the lasers quickly. To do this, new mirrors and shutters were put in place in the common line of the Drive Laser and Heater Laser respectively before they entered the transport pipe down into the accelerator injector vault (Figure 1).

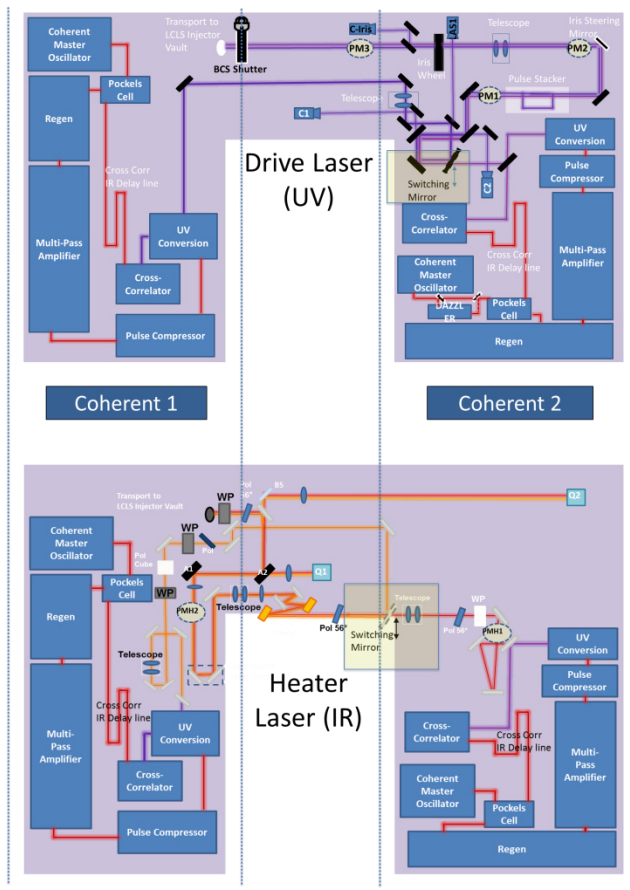


Figure 1: Optics table layout of active and spare injector laser systems.

* LCLS is supported by the U.S. Department of Energy, Office of Science under Contract DE-AC02-76SF00515.
#alverson@slac.stanford.edu

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The mirror mounts would have control along three axes – transverse tilt control Left/Right and Up/Down for the mirror itself, and longitudinal In/Out on a linear stage to insert and retract the mirror entirely from the laser path. Additionally, two solenoid driven shutters for Drive Laser and two for the Heater Laser would be used to block the unused beams or block both beams while the mirrors are inserting or retracting. These all need to be controlled remotely over the EPICS Channel Access network with automated code to assist in fast switching.

IMPLEMENTATION

For simplicity and to help with maintenance costs we chose to stick with the same or compatible hardware devices already used elsewhere in the laser system wherever possible (Figure 2).

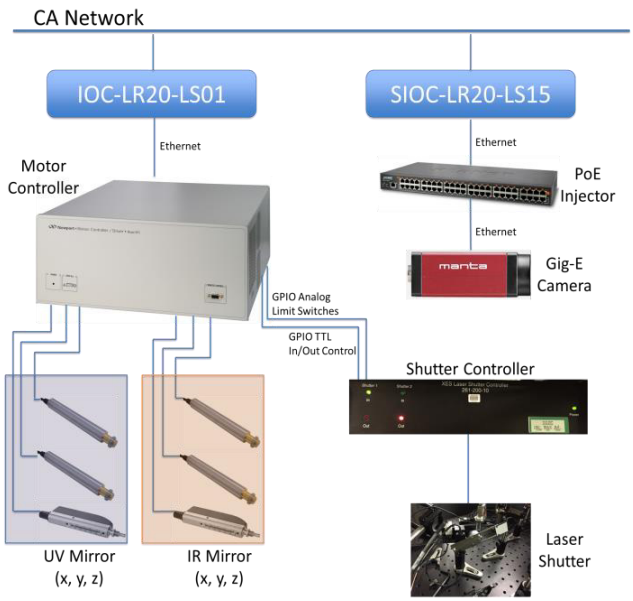


Figure 2: Hardware architecture for installation.

Motion Control

For motor control in the LCLS Laser System we typically use the Newport XPS-Q8 smart motor controller in conjunction with compatible stepper or servo motors also from Newport. We kept to this trend and chose the LTA-HS stepper motor for the In/Out control and used TRA6CC steppers for the transverse steering control.

Shutters

The shutters we use are SLAC built devices consisting of a black high absorption (non-reflective) material on a solenoid driven arm. The fully open and closed positions have photo-interrupter limit switches. Control is managed by a SLAC built controller chassis which provides local status and power for the shutters and can in turn be controlled via TTL output and position switch read back signals. Since we were already using the Newport XPS controller for the mirror motors, it was a simple decision to use the built-in GPIO ports to handle control and read back of the shutters as well.

Cameras

In order to monitor the spatial profile and alignment of the spare laser while not in use, a camera was installed which the Laser Operators can monitor daily and ensure that the spare is in good condition. We used an AVT Manta G033B Gig-E camera with Power-Over-Ethernet (PoE) and the cover glass removed by the vendor to enable use with UV.

A Soft IOC is run on a Dell PowerEdge R620 server where all image processing is done. As raw image data from the camera can require a lot of bandwidth, the cameras are run directly to the server on a local DHCP managed IP address before being processed down and set out over the Channel Access network. A Planet HPOE-2400G PoE Injector is used between the camera and the server to provide power to the camera.

TWO-BUNCH CONSIDERATIONS

In addition to the primary goal of fast remote switching between the two laser systems, it was also desired by the physicists to allow for both lasers to be run simultaneously separated slightly in time. To accomplish this, the switching mirrors are replaced with beam splitters which are left in place and the shutters alone select which beam (or both) is allowed down into the injector vault. The time delay can be adjusted either by varying the EVR trigger delay one of the laser's RF phase-lock-loop (PLL) or by using the existing motorized delay stage in the pulse stacker.

Safety

The Beam Containment System (BCS) safety system uses toroids to measure the electron beam current at various points along the accelerator and trips off the machine if it measures losses exceeding the limits allowed by the Radiation Safety group. These toroid comparators use gated ADCs and to ensure that they do not miss the beam, photo-diodes in the laser are required to measure the laser pulses within a timing gate shared with the toroid comparator system. If the laser falls outside of this 40 ns wide gate, the beam is shut off.

With the addition of a second laser pulse it was necessary to ensure that this photo-diode system could see both pulses and act to make sure that they both fell within the timing gate. As long as both pulses were contained within the 40 ns gate we found that the toroids would successfully average over them both to give an acceptable measurement of the total beam losses.

SOFTWARE

All hardware for the project is controllable through EPICS R3-14-12 with an EDM gui interface (Figure 3). Position setpoints both for the In/Out positions and steering are saved for each laser and can be changed by Laser Operators to compensate for alignment differences between each laser line.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

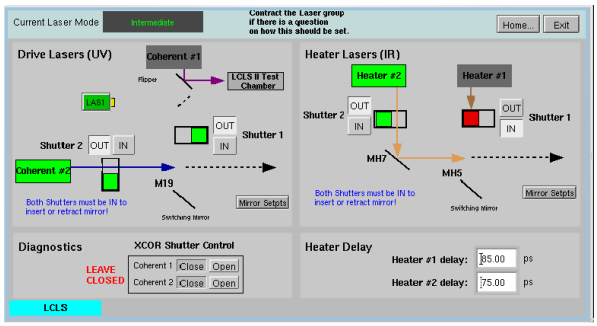


Figure 3: EDM switching interface.

An additional PyQt4 application was also written in cooperation with the control room Operators to provide an automated interface to load and change parameters when switching between lasers (Figure 4).

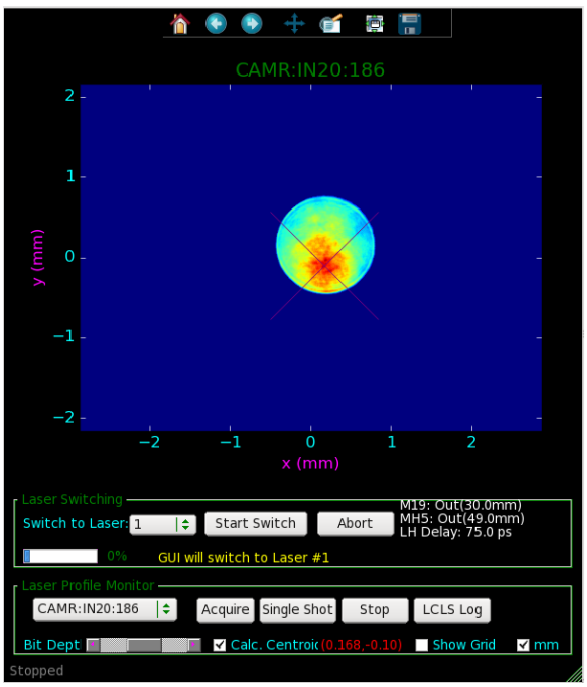


Figure 4: PyQt4 Switching Interface. Camera image shows the laser on the virtual cathode after the switching mirrors to ensure that beam is aligned properly.

CONCLUSION

With the old laser setup, switching lasers in the event of a failure would often take several hours. With the new remote switching controls in place however, the Control Room Operators have successfully managed to perform the switch on many occasions in less than 15 minutes. This drastically reduced the downtime incurred by laser equipment failure as well as allowed for more opportunities for the Laser Operators to request that the lasers be swapped so that they may perform scheduled maintenance which in turn has improved the long term performance of the injector laser system as a whole.

We have also now had several experimental shifts running with the new two-bunch operation mode [2] and have successfully seen both laser pulses within the BCS timing gate (Figure 5).

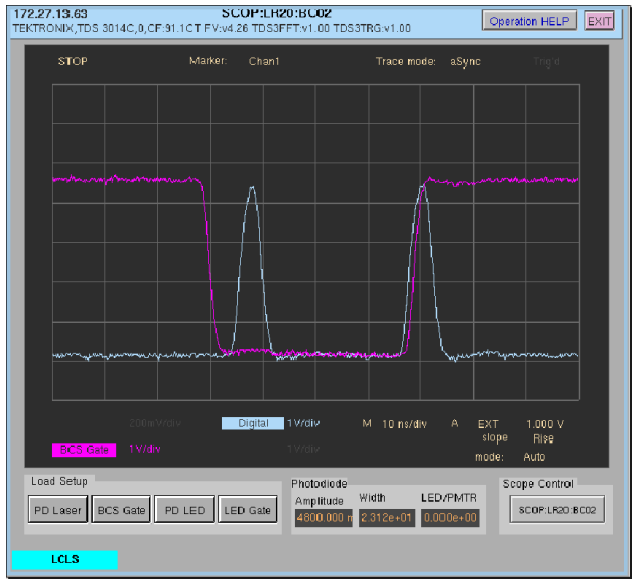


Figure 5: BCS Timing Gate on oscilloscope. Magenta trace shows the toroid timing gate and the blue trace shows the two laser pulses.

Having proved that the Toroid Comparator system is able to integrate across both beams simultaneously and trip the beam appropriately in the event of losses, we were approved to use this technique to create XFEL for experiments. Taking the light emitted in the Undulator Hall from the two bunches through the into the FEE diagnostic area showed that we were indeed able to lase on both bunches separately by ionizing gas and reading out the scattered particles on PMT detectors (Figure 6).

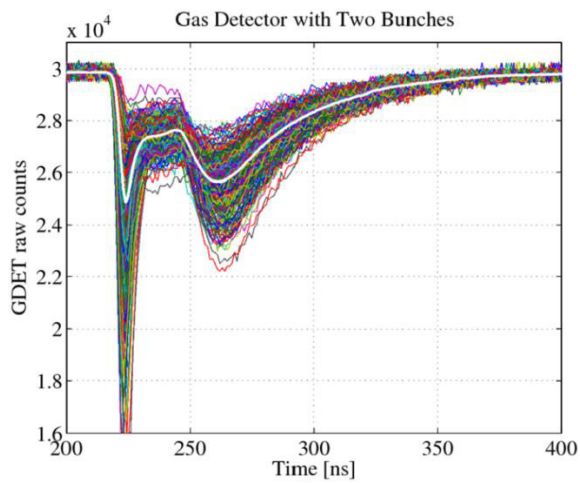


Figure 6: Gas Detector raw waveforms (above) show a 2.5:5 ratio after the laser heater was timed for the first bunch. The spike in the front is an instrumental reaction to coherent synchrotron radiation. Therefore the integrated GDET signal typically uses the counts from 250 to 400 ns.

The two-bunch XFEL was also taken to one of the user hutches where they were imaged hitting a single injected water droplet simulating the use case for experimentation (Figure 7).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

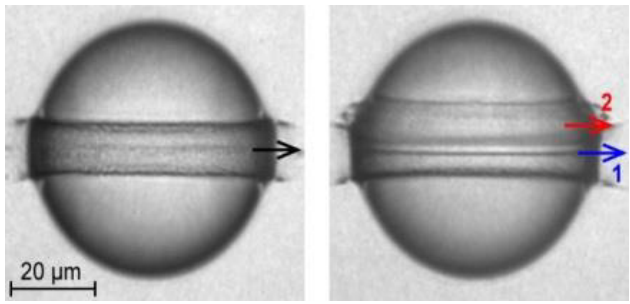


Figure 7: XFEL beams imaged by camera in CXI experimental hutch hitting water droplets. Single bunch beam is seen on the left and two-bunch on the right. The arrows indicate the XFEL beams.

ACKNOWLEDGMENT

Work supported by the U.S. Department of Energy under contract number DE-AC02-76SF00515.

REFERENCES

[1] D. Dowell, et al., "LCLS Injector Drive Laser", PAC'07, June 2007 SLAC-PUB-12948, http://www.slac.stanford.edu/cgi-wrap/getdoc/slac-pub-12948.pdf

[2] A. Marinelli, et al., "High-intensity double-pulse X-ray free-electron laser", Nature Communications, March 2015, http://www.nature.com/ncomms/2015/150306/ncomms7369/full/ncomms7369.html?WT.ec_id=NCOMMS-20150311

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

NEW DEVELOPMENTS ON EPICS DRIVERS, CLIENTS AND TOOLS AT SESAME

I. Saleh, Y.S. Dabain, A. Ismail, SESAME, Allan, Jordan

Abstract

SESAME is a 2.5 GeV synchrotron light source under construction in Allan, Jordan. The control system of SESAME is based on EPICS and CSS.

Various developments in EPICS drivers, clients, software tools and hardware have been done. This paper will present some of the main achievements: new linux-x86 EPICS drivers and soft IOCS developed for the Micro-Research Finland event timing system replacing the VME/VxWorks-based drivers; new EPICS drivers and clients developed for the Basler GigE cameras; an IOC deployment and management driver developed to monitor the numerous virtual machines running the soft IOCs, and to ease deployment of updates to these IOCs; an automated EPICS checking tool developed to aid in the review, validation and application of the in-house rules for all record databases; a new EPICS record type (mbbi2) developed to provide alarm features missing from the multibit binary records found in the base distribution of EPICS; a test of feasibility for replacing serial terminal servers with low-cost computers.

INTRODUCTION

SESAME consists of a 22 MeV Microtron, an 800 MeV Booster Synchrotron and a 2.5 GeV Storage Ring. Control System Implementation uses (EPICS) base R3.14.12. Servers are implemented as EPICS Input/ output Controllers (IOCs). Clients are implemented using a custom build of Control System Studio (CSS) based on V.3.16. Siemens S7 PLC controllers are used for the machine interlocks. An Allen Bradley PLC controller is used for the Personal Safety System (PSS). VME hardware is used for the timing system. Development and administration platforms use Scientific Linux 6.4. A Git version control is used to track development and documentation. All clients, servers, and controllers are connected to an isolated machine network. There are twelve virtual servers reserved to run the IOCs, archive system, alarm system and Git repositories.

The control systems have been implemented for the Microtron, Transfer Line 1 (TL1) and Booster. The Booster's control system is divided into seven subsystems: vacuum, power, RF, diagnostics, cooling, timing and Personal Safety System (PSS). Each control subsystem consists of one or more clients, servers, and controllers [1]. This paper will focus on the developments on EPICS drivers, clients and tools at SESAME.

TIMING SYSTEM DRIVER

The timing system for the Booster consists of one event generator VME-EVG230 and one event receiver VME-EVR230, both of which are connected to the EPICS network over TCP/IP. New Linux-x86 EPICS drivers were developed for the timing modules from Micro-Research Finland.

The VME-EVG230/VME-EVR230 are traditionally controlled over the VME bus. Both modules can also be programmed over the available Ethernet port however. Building an EPICS driver for controlling the timing modules over Ethernet instead of the VME bus has the following benefits:

- Drops the dead weight: The VME crate, the VME CPU card, the RTOS that runs on the CPU card (along with any required licenses), and the debug terminal that connects to the CPU card are no longer needed.
- Lowers the cost of implementation: A direct consequence of the point above.
- Confines the required development skills to Linux/x86 platforms. Knowledge in VME-bus, VxWorks, or any other RTOS/OS is not required.
- Maintains coherency in the control infrastructure. This point may be specific to SESAME only since all of the IOC's at SESAME run on Linux/x86 platforms.

The device layer uses the traditional asynchronous processing model of EPICS support modules. The driver layer uses the remote programming protocol of the timing modules to control them. This protocol uses UDP. The drivers add feedback, retransmissions, and timeouts to create reliable communication over UDP.

Both drivers are LGPL'd and can be found as public domain on Github [2]. The client of the timing system is implemented in Control System Studio (CSS) and is shown in (Fig. 1).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

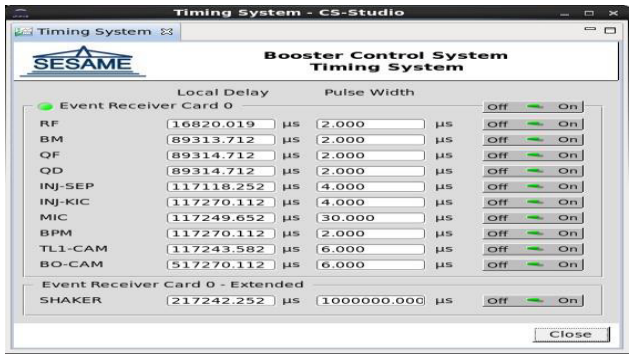


Figure 1: CSS Timing OPI.

BASLER GIGE CAMERA DRIVER

New Linux/x86 EPICS driver was developed for the GigE camera series from Basler. The device layer uses the traditional asynchronous processing model of EPICS support modules. The driver layer uses the Pylon libraries from Basler to control the cameras over Ethernet. Once configured, the driver starts a separate server thread for each camera. Each server thread listens for commands such as image capture or gain and exposure settings. A set of mutexes and signals are used to synchronize operation between the various threads.

BASLER GIGE CAMERA CLIENT

To connect to the Basler GigE camera driver, a custom client was built at SESAME utilizing the EPICS channel access libraries. The client is responsible for providing a graphical user interface to render the camera's video stream possibly doing some image post-processing like colour mapping and to control the camera image settings.

The camera client uses the EPICS channel access libraries to monitor and control the various records provided by the driver. A waveform record contains the grayscale image data and various analog input/output records provide access to camera's gain, exposure, ROI and trigger source. Once the camera client is started it connects to the driver and enables the camera. The camera is usually disabled if unused to save network bandwidth. Once the camera is enabled, the waveform record containing the grayscale image starts updating as the camera is triggered. On each update, a call-back is called in the client to copy the image data to one of two buffers used to render the image. One of the buffers is being read from while the other is written to. And once the image buffer is completely written to and ready, a switch is done so that the next render uses the new image data. The double-buffering is used so that no tearing happens in the rendering of the image.

The client is written in C using OpenGL, SDL, AntTweakBar and of course EPICS channel access libraries.

OpenGL is an API to do high performance 2D and 3D rendering on computers. In the client, it is used to render the image frames maintaining a high framerate while consuming low CPU cycles and RAM memory. SDL is a cross-platform library used to provide access to keyboard, mouse, and windowing system and graphics hardware via OpenGL. AntTweakBar is a library that works with SDL and OpenGL to provide a light graphical user interface to interactively tweak parameters and settings in an application. It is used to control and show the camera image settings like region of interest, framerate and trigger source.

Both driver and client are LGPL'd and can be found as public domain on Github [3]. (Fig. 2) shows the camera client presents colour mapping option.

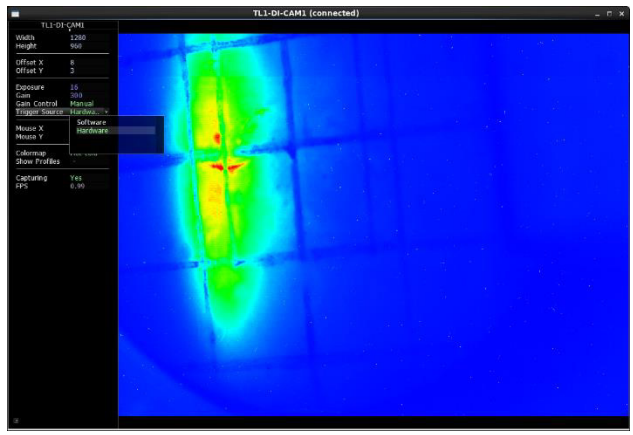


Figure 2: Camera Client.

IOC MANAGER

The control infrastructure at SESAME uses 12 virtual machines to run the different IOC's and 8 physical machines to run the clients and for development purposes.

The IOC manager is a Linux/x86 EPICS driver developed for the purpose of managing all the machines that are part of the control infrastructure at SESAME.

Specifically, the IOC manager provides the following services:

- Hostname and user of the IOC
- Date, time, and uptime of the IOC
- Drive, memory, and CPU utilization
- Control over which IOC's to run
- Enabling/disabling of the IOC's
- Build system

The date/time information give diagnostics on whether or not the IOC is synchronized with the NTP server. The drive, memory, and CPU utilization are indicators on the health of each IOC. They are connected to the alarm handler at SESAME. Control over which IOC's to run enables us to run different IOC's on different machines, and lets us know which IOC's are running where. Enabling/disabling the IOC's allow us to manually shutdown, turn on, or reset the IOC's. Finally, the build system of the manager synchronizes the IOC with the main Git repository, builds the source, and restarts the IOC's.

Each IOC is run inside a separate terminal multiplexer session. This allows us to a remotely attach/detach to a session whenever we need to access the IOC itself.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The driver layer uses traditional synchronous EPICS processing model of EPICS support modules. The device layer uses standard POSIX API to access machine information. Bash scripts are used to synchronize with the main Git repository. The client of the IOC manager is implemented in CSS and is shown in (Fig. 3).

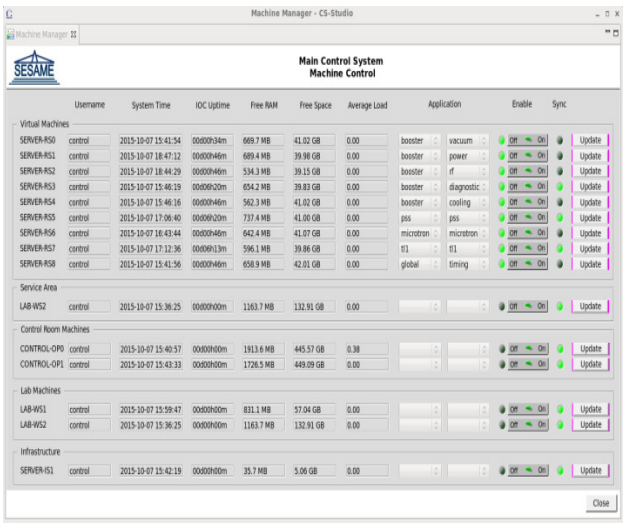


Figure 3: IOC Manager Client.

DB-CHECK TOOL

The db-check tool is a command line tool built at SESAME that automatically analyzes, validates and aids in applying continually evolving in-house rules for EPICS records databases. It was primarily built to help in reviewing, unifying and maintaining the numerous EPICS databases present.

Features provided by the db-check tool include: listing all records by name, or by type; querying a record to show all of its details and Process Variables (PVs); producing reports of records usage in each system, and what fields are set in those records; producing up-to-date alarm and archiver XML configuration files; and checking some in-house rules that PVs should follow.

Examples of the PV rules applied include: always having a DESC field present in all records: always having a ZNAM, ONAM, ZSV and OSV for BO records; always having the name of PVs to start with a one of a predefined set of prefixes (eg. is, get, set, reset, enable or disable)..

The way the db-check works is by parsing the DB and DBD files using a PEG grammar. The tool is built in D programming language and using pegged library for the parser [4]. The D programming language was chosen because it provides modelling power, efficiency and high performance in a coherent set of well-thought features, and because it interfaces easily and natively to the C programming language. An important feature that specifically helped in this project is compile-time function evaluation (CTFE) and mixins. This feature is used in pegged to build the parser using the grammar definition at compile-time instead of depending on a separate build step. And it reduced the compile-run-debug cycle time.

MBBI2 RECORD

Mbbi2 is a new record type developed at SESAME to overcome some limitations present in the official mbbiDirect record type in regards to alarms. An mbbi2 record is almost exactly the same as an mbbiDirect record except for its ability to specify alarm states for individual bits.

Frequently, devices provide read access to status registers where different bits mean different things. Some of those bits hold on/off status, others hold warnings and others hold interlock status. Using an mbbiDirect record to model this in EPICS creates a problem for alarm handling as value alarms are not supported. To support this without introducing a new record type will require using multiple CALC fields. The way mbbi2 resolves this is by setting an alarm severity on zero and one for each bit. The alarm severity for the record is equal to the highest alarm severity of its bits.

The main problem of introducing a new record type is the maintenance requirements of integrating it with the different drivers and their build systems.

TERMINAL SERVER

The terminal server is an experiment to provide simple serial to Ethernet converters using low cost computers. The computers have multiple serial ports that are mapped using the accompanying software to TCP ports available over an Ethernet network. The software was built to run primarily under minimal x86 or x64 Linux systems using libevent2 for asynchronous non-blocking communication. The software, once started, reads a configuration file and accordingly connects to a serial device on the specified baud rate and starts listening for clients on a TCP port. Once a client connects, data coming from the serial device is forwarded to the TCP connection and vice versa. Some simple statistics are printed periodically to the console for simple diagnostics.

Tests for the system showed that it was working well, but it has not been used in a production settings as of yet.

CONCLUSION

The control system of SESAME is based on EPICS and CSS. Development of new drivers, clients and tools at SESAME is important to make the control systems up to date and more consistent. Standards are used for both EPICS databases and CSS client OPIs.

REFERENCES

[1] A. Ismail, I. Saleh, Y. Dabain,, “CLIENTS DEVELOPMENT OF SESAME'S CONTROL SYSTEM BASED ON CSS”, Proceedings of PCaPAC2014, Karlsruhe, Germany, 2014.
[2] https://github.com/aismail2
[3] https://github.com/sesamecs
[4] https://github.com/PhilippeSigaud/Pegged

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

CONTROL SYSTEM UPGRADE FOR SUPERKEKB INJECTOR LINAC

M. Satoh#, F. Miyahara, Y. Seimiya, K. Mikawa, T. Suwada, K. Furukawa, KEK/ SOKENDAI, Tsukuba, Japan
T. Kudou, T. Ichikawa, Y. Mizukawa, K. Hisazumi, S. Kusano, Mitsubishi Electric System & Service Co., Ltd, Tsukuba, Japan
H. Saotome, M. Takagi, Kanto Information Service (KIS), Accelerator Group, Tsuchiura, Japan

Abstract

The KEKB project has successfully ended 13 years operation in the June of 2010. The construction of SuperKEKB main ring has almost completed for aiming at the peak luminosity of 8 x 10^{35} $cm^{-2}s^{-1}$. The injector linac upgrade is also going on for increasing the intensity of bunched charge with keeping the small emittance. The key upgrade issues are the stable operation of a new positron capture system, a low emittance photo-cathode rf electron source, and the establishment of low emittance preservation. The injector linac beam commissioning started in the October of 2013. The control system performance determines the beam operation efficiency of injector linac, and it eventually has a strong impact on the experimental results of physics. In this decade, the linac control system has been gradually transferred from the in-house system to the Experimental Physics and Industrial Control System based one for the high availability of beam operation. In this paper, we present the detail of SuperKEKB injector linac control system.

INTRODUCTION

The linac beam control system is based on a standard client and server model with three layers. It comprises a client, a server, and a local controller layers. At the beginning stage of KEKB operation, the original linac control system has been developed by using the in-house software libraries based on the remote procedure call (RPC). The client user interfaces have been implemented by a command line interface based on shell script and a graphical user interface based on Tcl/Tk scripting language. Around a decade ago, the middle phase of KEKB operation, the linac control system has been upgraded to a new framework based-on the Experimental Physics and Industrial Control System (EPICS) [1] to improve the affinity between the linac and KEKB main ring control systems. These improvements make it possible to conduct the correlation analysis of linac and main ring parameters in much easy way. Such analysis can strongly help to find a source of injection rate deterioration and any other troubles.

After introducing the EPICS framework, the new client side user interfaces have been implemented by Python scripting language since its rich library modules and functionalities can accelerate the software development process. For the simultaneous top-up injection of KEKB and PF rings, an event based timing system has been installed to enable the pulse-to-pulse beam modulation [2].

OUTLINE OF CONTROL SYSTEM

Computer Environment

In the beginning phase of KEKB project, the six Compaq Alpha servers based on the Tru64 UNIX operating system were utilized as the server computers. All server programs were basically run on them. Two of them were connected to the RAID disk drive via SCSI bus interface, and they can work as the active/standby redundant NFS servers for aiming at the high availability operation. Since the Tru64 UNIX was obsoleted, all server machines have been gradually replaced by the Linux-based servers. Eventually, we made decision to use the Linux base system without cluster functionality after the evaluation of several different types of high availability cluster system based on Linux. For keeping the high availability of server machines without cluster scheme, the blade server system was installed as the high reliability server. Currently, fifteen Linux based blade servers are utilized as the server computers with CentOS 5.11 x86_64. Both of server and client side control software are running on the server machines. CentOS 6.7 and 7.1 running on two blade servers are currently tested by using VMware vCenter of the virtual computing technology.

The Tektronix X terminals and touch panel displays based on PC9801/DOS machines have been originally utilized as the operator terminals. These terminals were replaced by the Linux based PCs and Windows based PCs with the X server application software of ASTEC-X and

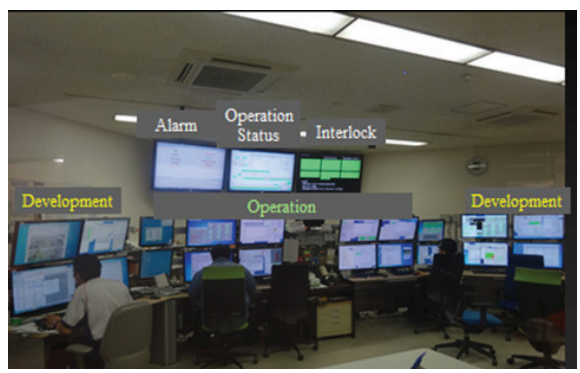


Figure 1: Photograph of the KEK electron/positron injector linac main control room.

e-mail address: masanori.satoh@kek.jp.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Reflection X. Figure 1 shows a photograph of the linac main control room. Three large liquid crystal displays (LCDs) with 55'' show the alarm, operation, and interlock statuses. Sixteen LCDs with 27'' are used for running the beam operation software and its development.

Network Environment

In the previous linac control network system, Cisco Catalyst 4506 and 3750 have been utilized as the master network core and slave switches, respectively, for enabling the active/standby failover. Each of them was independently connected to 45 edge switches of Catalyst 2950 via single or multi-mode optical fibers with 100 Mbps bandwidth. For the network connection between the edge switches and the local controllers like a programmable logic controller (PLC), the optical fiber is used for avoiding the noise generated by the klystron modulators.

Toward SuperKEKB operation, the core switch system was replaced by 6 of Cisco Catalyst 3750X. They can work as the active/active redundant system based on the virtual switching system technology. We replaced also the edge switch by Catalyst 2960S, and the network connection speed was improved up to 1 Gbps bandwidth. In addition, the wireless network was installed into the beam line tunnel for improving maintainability.

Local Controller

For the injector linac control system, many kinds of different local controllers are utilized for the injector linac component control as listed in Table 1. The ladder PLCs control the magnet power supplies, vacuum pumps, and safety related signals. About one hundred seventy CAMAC and VME based timing delay cards have been replaced by the 25 event timing cards based on VME64x bus for the pulse-to-pulse beam modulation. The reduction of number of timing modules improves the system availability. The PLCs used for the klystron modulator control have been gradually replaced by the new embedded controller of Armadillo. The EPICS Input/Output controller (IOC) can run on it. The Windows based digital oscilloscope as the BPM readout system will be replaced by a new one based on VME card soon. Its measurement precision is around 3 μm [3].

Table 1: List of the Network Attached Local Controllers used for the Linac Control System.

Devices	Accelerator components (# of components)	# of local controllers
VME64x	Event based timing system (MRF EVG-230, EVR-230RF)	25
PLC	Magnet (363)	59
	Vacuum (333)	26
	Klystron (5)	5
	Charge interlock	3
Network attached power supply	Magnet (105)	105
Linux based PLC	Profile monitor (100)	30
Embedded Linux	Klystron (66)	66
Data logger	Temperature (690)	28
Oscilloscope	BPM (90)	23
NIM modules	Timing watchdog (15)	15
Total		385

EPICS ENVIRONMENT

Overview

In the linac control system, the EPICS IOCs were originally implemented with the base R3.14.9 for wrapping the existing in-house control system. Table 2 shows the total number of IOCs for each subsystem. The EPICS process variables (PVs) are just communicating with the existing system based on RPC when the software access any PVs. Toward higher reliability, the server applications were remodeled for the direct communication between the IOCs and local controllers by EPICS base R3.14.12. All EPICS IOCs are running inside the vncserver sessions as shown in Fig. 2.

Table 2: Number of EPICS IOCs used for each Subsystem.

Subsystem	# of ICOs
Safety	2
Monitor	48
RF	57
Magnet	19
Vacuum	1
Operation	3
Timing	21
Temperature	2
Total	153

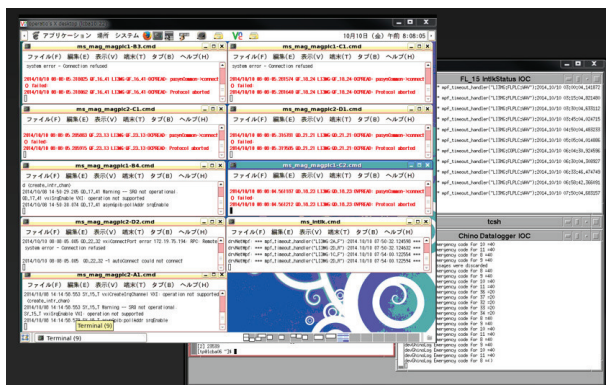


Figure 2: All EPICS IOCs are Running in the Vncservers.

Alarm

The independent in-house alarm applications were originally developed and utilized for each subsystems. For the comprehensive alarm management, we adopted the EPICS Control System Studio (CSS) [4] alarm 3.0.0 together with PostgreSQL 9.1.4 as a backend database

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

engine. Around 1000 PVs are currently registered to the CSS alarm. The graphical user interface (GUI) showing the alarm status was developed by the Python scripting language. The panel shows the both of summary and detailed information of alarm status. The more detailed historical list of alarmed PVs can be also displayed in the other windows. Since the total number of registered PVs will be increased toward SuperKEKB operation, we are evaluating the speed performance and the robustness of CSS alarm with the large number of PVs.

Archiver

In the original linac control system, we developed the simple archiver tool recording the parameters into a text file together with the dedicated viewer software. After the implementation of an EPICS based control system, the EPICS channel archiver and CSS archiver 3.2.2 together with PostgreSQL 9.1.4/9.3.3 have been utilized. The total number of PVs registered to each archiver is currently 44063. The daily disk space usages are approximately 2 GB and 4.5 GB for channel and CSS archiver, respectively.

The web based data browser is utilized for the CSS archiver though the standard java based viewer is used for the channel archiver. The web based one was developed with Flex 4.6, PHP 5.3.6, and Amfphp 1.9. It can provide the easy access to archiver data from the any client devices including mobile one. It has the functions of correlation plot, multiple vertical axes, PV name search, and autocomplete for the quick parameter analysis.

The data retrieve speed of CSS archiver is slower than that of channel archiver. For improving these issues, the pg_reorg option was added into PostgreSQL database. It can reduce the CSS archiver database size by 33%, and improve the data retrieving time to a certain extent. For the further improvement of data retrieving speed, another scheme using the NoSQL type database of Casandra as a backend is now being evaluated.

OPERATION SOFTWARE

Outline

The original linac operator interface was developed by the Tcl/Tk scripting language and command line interface. For the rapid application development, the main software development language was transferred to Python scripting language. The communication between the operation software and EPICS PVs can be established via PythonCA module developed for the KEKB project.

Operation Status Display and Electronic Logbook

The real-time linac operation status is given through the Web-based application which can be run on the multiple operating system without installing specific software. This web application was implemented by using WebSocket together with Node.js and Socket.IO library. WebSocket can reduce the CPU load on the server machines in comparison with the other technologies like Ajax or Comet since its protocol can provide a full-duplex communication channels over a single TCP connection. This application shows the information of present beam repetition, high power klystron status, radiation monitor status, and so on as shown in Fig. 3.

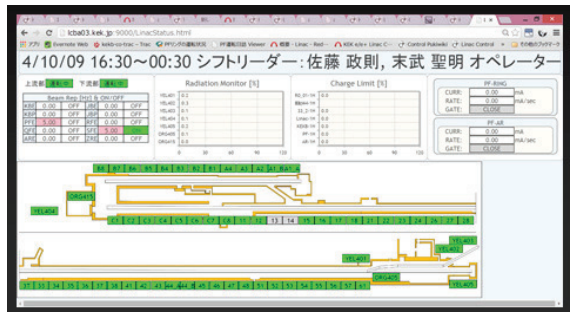


Figure 3: Web-based linac operation status.

The electronic operation logbook system (OPELOG) can enable the rapid operation information sharing between the accelerator operation team and hardware group. We developed an OPELOG system with Flex 4.6, PHP 5.3.6, Amfphp 1.9, and PostgreSQL 9.4.0. It comprises the summary and detailed information screen for each operation shift. OPELOG can automatically record the all routine operation state transition like the status change of beam repetition and klystron failure. In addition, a snapshot of operation GUI image can be easily imported into the detailed information screen. Moreover, OPELOG has the quick search function of past log by a key word combination set during a specific time period. It can be a strong help to find a solution of similar trouble occurred in the past.

CONCLUSION

Towards SuperKEKB project, the injector linac upgrade and commissioning are now on going for aiming at the 4 ring simultaneous top-up operation. Since the middle phase of KEKB operation, the linac beam control system has been gradually transferred from the in-house system based on RPC to the EPICS based one. Many operation software are developed by the Python scripting language. The data retrieving speed performance of archiver and the rapid development of advanced commissioning tools are recent main issues.

REFERENCES

[1] http://www.aps.anl.gov/epics/.
[2] K. Furukawa et al., “Pulse-to-pulse Beam Modulation and Event-based Beam Feedback Systems at KEKB Linac”, in Proceedings of IPAC’10, Kyoto, Japan, pp.1271-1273 (2010).
[2] http://ics-web.sns.ornl.gov/kasemir/archiver/.
[3] F. Miyahara et al., "HIGH POSITION RESOLUTION BPM READOUT SYSTEM WITH CALIBRATION PULSE GENERATORS FOR KEK e+/e- LINAC" in Proceedings of IBIC2015, Melbourne, Australia, September 13-17, TUPB023 (2015).
[4] http://controlsystemstudio.org/.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONTROL SYSTEM DEVELOPMENTS AT THE ELECTRON STORAGE RING DELTA

D. Schirmer*, A. Althaus, F. Bahnsen
Center for Synchrotron Radiation (DELTA), TU Dortmund, Germany

Abstract

Increasing demands, mandatory replacement of obsolete controls equipment as well as the introduction of new software and hardware technologies with short innovation cycles are some of the reasons why control systems need to be revised continuously. Thus, also at the EPICS-based DELTA control system [1], several projects have been tackled in recent years:

(1) Embedding the new CHG[1]-based short-pulse facility for VUV and THz radiation [2-4] required, for example, the integration of IP-cameras, Raspberry-Pi PCs and EtherCat/TwinCat wired I/O-devices [5, 6].

(2) The request for a staff-free control room led to the programming of new web applications using Python [7] and the Django framework [8]. This development resulted in a web-based interlock system that can be run, amongst others, on Android-based mobile devices.

(3) The virtualization infrastructure for server consolidation has been extended and migrated from XEN [9] to the kernel based KVM [10] approach.

(4) I/O-units which were connected via conventional fieldbus systems (CAN, GPIB, RS-232/485), are now gradually replaced by TCP/IP-controlled devices.

This paper describes details of these upgrades and further new developments.

INTRODUCTION

DELTA, a 1.5-GeV electron storage ring, is operated since 1999 by the TU Dortmund University as a synchrotron light source for campus-based, regional and international users. Since 2011, the facility has been extended by a short-pulse source for VUV and THz radiation making use of the CHG principle. An upgrade to EEHG[2] is in preparation [11]. Not only for these reasons the EPICS-based DELTA control system has been revised and complemented in many fields.

NEW SOFTWARE

Network Administration Tool

Since the number of control system network devices and, thus, the complexity of the network topology increased constantly, it was mandatory to develop a DELTA-specific management tool. With the help of this tool, all individual devices and their connection properties are registered centrally. It administers network interfaces, IP/MAC numbers and assignments to domains. Furthermore, it generates consistent DNS/DHCP configuration files as well as location and network plans. The program is implemented as a web application and is based on the high-level Django/Python framework [7, 8]. All data and configurations are stored in the DELTA MySQL [12] database.

Web Applications

The local radiation safety authority has given its consent to a non-permanent attendance of the control room during standard machine operation. For security reasons, however, it was necessary to develop an additional alert system, which notifies the operator to machine malfunctions via a mobile phone.

For real-time monitoring of the accelerator interlock warnings, a web-socket connection is performed, which keeps a full-duplex communication link between a client (e.g., an Android-based mobile phone) and a server continuously open. The connection uses WAMP[1] [13], a web-socket sub-protocol (with RPC[2] and publish & subscribe mechanism).

On the server side (see Fig. 1 below), *Crossbar.io* [14] is the central software component. It interacts as a message WAMP-router for different so-called service nodes. One node provides the service to read EPICS records another calculates und triggers interlock warnings (all implemented with Python). In addition, it launches web services that provide panels accessible from various URLs. There are web pages for machine status (e.g., beam current, lifetime, insertion device status), interlock notifications as well as the electronic logbook (see Fig. 2).

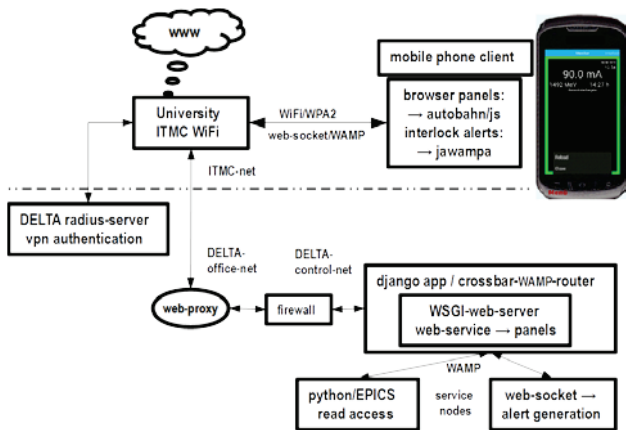


Figure 1: Software setup of the WiFi alert system.

On the client side (see Fig. 1 top), e.g., an Android-based smartphone, an *Autobahn/JS*-application [15] (an implementation of WAMP in javascript) displays the

*detlev.schirmer@tu-dortmund.de
[1]Coherent Harmonic Generation
[2]Echo -Enable Harmonic Generation

[1]Web Application Messaging Protocol
[2]Remote Procedure Call

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

server-side spawned panels, and a *jawampa*-application [16] (library to support WAMP to Java) indicates the interlock warnings, all in real-time. The communication is established via WiFi inside the DELTA building. For security reasons, all WiFi-clients must connect to a specially preconfigured ITMC[1]-WiFi controller including an authentication mechanism (radius server [17]). After a successful login, a web-proxy inside the DELTA office network relays the link through a firewall to the dedicated web server/WAMP router within the machine control network, from where EPICS access is possible. An acoustic and vibration alarm will be activated on the mobile phone in the case of a machine alert or server connection lost (see Fig. 1).

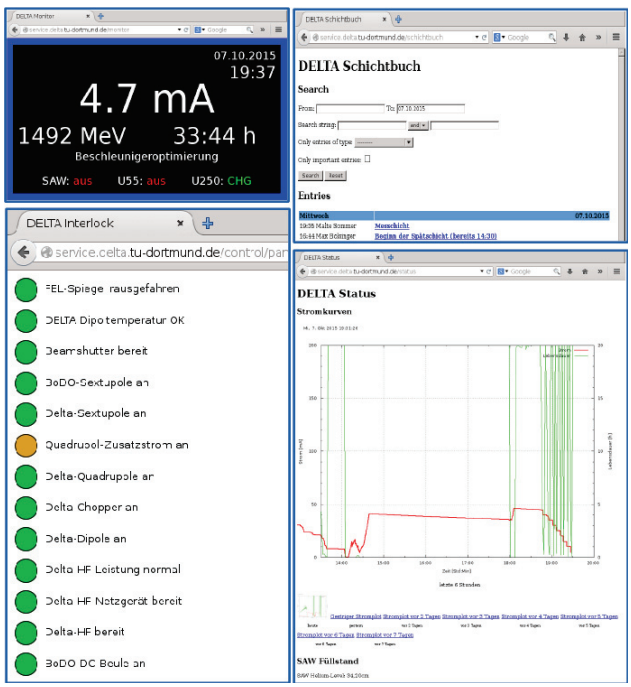


Figure 2: Web applications (panels for: DELTA monitor, logbook, interlock and status).

As another consequence, the central display server for all status monitor panels in the machine hall and the office corridors uses the same standardized web service (see URLs as already discussed above). Also PC-client access to the web-based logbook (elog) or the interlock panel is provided within the DELTA-Intranet or via VPN-access even worldwide. Dedicated permitted URLs are also accessible directly from the Internet (see Fig. 2).

Furthermore, a new acoustic message announcement system has been implemented. Alarms triggered by electron beam losses, temperature warnings, and orbit control problems as well as the injection timer are now audible in the control room and the machine hall. A PyEpics [18] script evaluates corresponding Epics records and plays appropriate speech announcements (pre-arranged voice-files).

[1]IT & Medien Centrum of the TU Dortmund

Server Consolidation

Since 2009, the XEN paravirtualization technique [9] was successfully in use. It was an ideal basis to consolidate and modernize the DELTA client/server control system network [19]. Nowadays, all server-CPUs support hardware-assisted virtualization (VT, AMD-V) and, in addition, kernel-based virtualization (KVM [10]) is by default merged into the standard Linux kernel. The migration to this kind of full virtualization was required for maintenance reasons. KVM is driven by *qemu* [20] and provides an API to control VMs with GUIs like *virt-manager* [21]. In the course of this migration, all server and virtual machines have been upgraded to the latest operating system (Linux Debian 7/8 [22]). This upgrade implies also the introduction of *systemd* [23], a new system and service manager for Linux-OSs and the successor of *sysvinit* [24]. Thus, even all EPICS soft IOC shells are now centrally managed by this powerful OS-integrated software package.

Also for the purpose of maintenance the default installation of all servers is now centrally and automatically managed by the software suite *Ansible* [25]. This software package provides a simple way to automate administration tasks and to setup a complete server infrastructure. Essentially, it is a model-driven configuration management and ad-hoc task execution tool. The in-house developed network tool stores primarily all IT devices including their connection properties in database entries. Based on this information, *Ansibel* generates, updates and transfers the corresponding configuration files and restarts all affected servers subsequently. Thus, the complete DELTA network configuration such as DHCP, DNS, VPN, NTP and user/group installation as well as login accounts (ssh keys, passwords) are automatically supervised site-wide.

INTEGRATION OF NEW HARDWARE

Today, real-time requirements are fulfilled satisfactorily by specially designed fast electronic circuits (e.g. DSP-driven booster synchrotrons [26] or FPGA-operated fast orbit feedback systems [27, 28]). But in most instances accelerator control applications do not need strong real-time features. As many hardware units provide standard Ethernet interfaces, they can be controlled over the TCP/IP network. Thus, for example, the error-prone RS232-serial to CAN-bus readout of the vacuum gauge control units has been replaced by MOXA port communication devices [29] that allow to control different kinds of serial devices directly over a TCP/IP-based Ethernet, avoiding the extensive VME-CAN-controller (VxWorks [30]) middle-layer.

For the same reason, several PLCs (e.g., electron gun control, valve and vacuum-pressure interlock) were subsequently equipped with TCP/IP communication modules. Even recent power supplies for the storage ring quadrupole magnets, also prepared for Ethernet connectors, are now driven directly over the TCP/IP control network, as far as synchronized magnet ramping

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

on a sub-second timescale is not required. The I/O-bus interface for the EPICS device support is realized by use of the well-established StreamDevice [31]/asynDriver [32].

More special hardware and software is necessary in order to drive mirrors, for example, in the evacuated laser beam lines of the short-pulse facility. Because the current system has been discontinued, a new control unit is being evaluated. Here, stepper motors [33-35] are controlled over EtherCAT [6], a vendor-initialized real-time Ethernet (see Fig. 3).

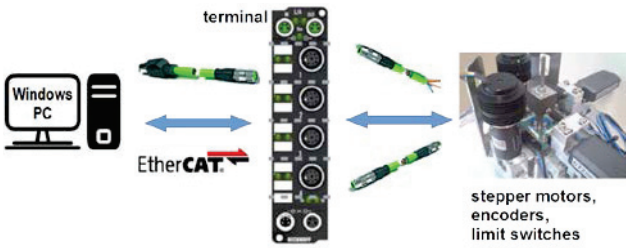


Figure 3: Hardware setup of stepper motor control [35].

The hardware configuration, such as a setup of motor parameters, is managed by the TwinCAT system manager [5], whereas the motor movement logic is programed on a PLC controller. The link to EPICS is accomplished via a vendor specific OPC[1]-server [36] and an EPICS IOC-shell with OPC driver support (OPC client) [37]. OPC is an interoperable standardized platform for industrial automation [38]. All three software levels are running on the same Windows-PC internally communicating over DCOM [39] (see Fig. 4).

Figure 4: Software setup of stepper motor control [35].

As the software encoding of the stepper motors is imprecise due to slippage and x/y motion coupling, the determination of the true mirror position must be realized by high-accuracy contact sensors and additional limit switches [33]. The analog sensor signals are connected to converter device providing a standard serial (RS232) communication interface. This port can be used again by RS232 EtherCAT terminal or by a single board controller (SBC, Raspberry-Pi) running Linux and PyEpics [33, 40]. A similar readout is possible for the digital signals of the limit switches (see Fig. 5).

[1]Open Platform Communication

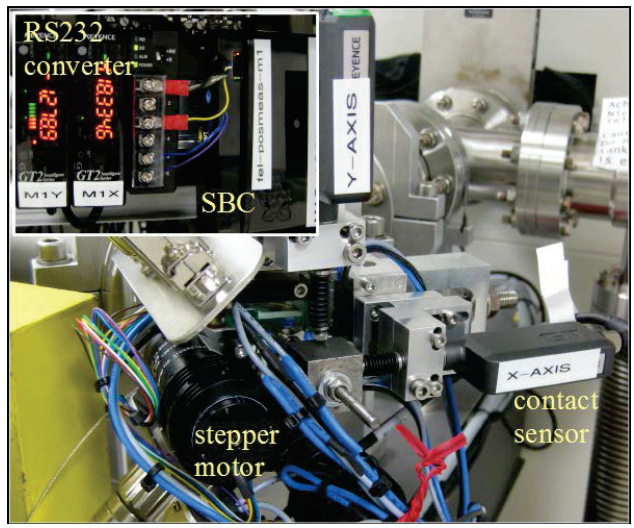


Figure 5: SBC for mirror position read back.

SINGLE BOARD COMPUTER (SBC) PROJECTS

SBCs are more and more the system of choice for "quick-and-dirty" all-round solutions. They run as central display-servers in the control room to present live images of oscilloscopes which are distributed inside the storage ring hall, or work as DHCP/DNS/VPN backup servers for the office network. Further examples are:

Test Bed for DAC Controller Cards

The DELTA booster synchrotron (BoDo) is ramped rather slowly by VME-IOC computers (from 70 MeV to 1.5 GeV and back in approx. 7 seconds). The ramp curve is user-defined and has to fulfil several sensible conditions. All booster magnet power supplies are driven by 16-bit DAC boards. The functionality of these boards is quite crucial and must be checked in advance by a simple SBC-based test bed. A python script on the SBC converts the booster ramping curve to a 16-bit digital I/O signal which is mapped via a photocoupler adapter card for voltage level matching and finally fed to the DAC-board input. The resulting analog DAC output value is measured and analyzed under several test conditions (e.g. ramp cycle frequency, curve shape, temperature variation) by a circuit analyser.

Laser Beam Alignment Diagnostics

For near-field angle measuring of the beam guidance in the laser laboratory, a SBC, equipped with a standard 5-megapixel CCD camera, is in preparation (see Fig. 6). A converging lens focuses the beam on the CCD chip. Changes of beam angle yield a shift of the focal point of the Gaussian intensity distribution. A python script quantifies this displacement and writes the measured value to EPICS records. Background radiation can be reduced via a SBC-GPIO triggered slit. This basic cost-saving setup allows angular measurement within very short distances with a resolution of 50 µrad in both transverse directions [40].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

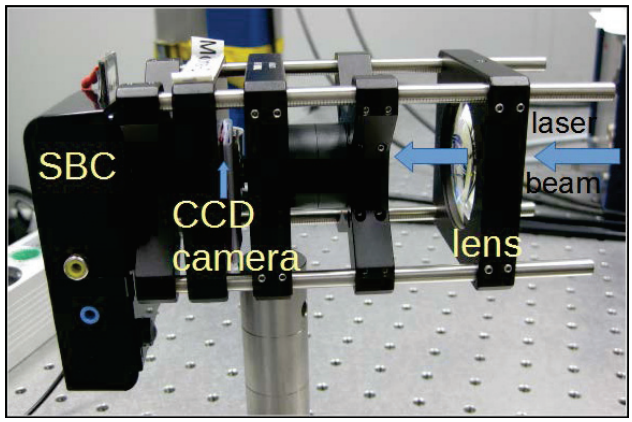


Figure 6: Test setup for laser beam diagnostics [40].

CONCLUSION AND OUTLOOK

The already existing hardware and software at the DELTA storage ring undergoes continuous modifications, mainly pushed by short industrial development cycles. The integration of new experimental setups described in this paper and others not discussed here are primarily for the self-evident purpose that accelerator and laser components can be remotely controlled or read out. To keep track of the increasing number of devices and their complex networking, the installation of new administration tools was necessary.

The future work will still put emphasis on the development of advanced web applications for machine control and monitoring. Furthermore, data-mining and post-processing has increasingly become an important task. Therefore, to optimize data analysis processes, an entire redesign of the DELTA MySQL database is essential.

ACKNOWLEDGMENT

We would like to thank all colleagues of the DELTA team for permanent support. In particular we thank Dennis Zimmermann, Carsten Mai and Boris Sawadski for constructive discussions and for providing detailed information about mirror control system, laser beam diagnostics and single-board controller applications.

REFERENCES

[1] D. Schirmer et al., Proc. ICALEPCS'07, Knoxville, 356.
[2] H. Huck et al., Proc. FEL 2011, Shanghai, 5.
[3] S. Khan et al., Synchrotron Radiation News, 26:3, 25 (2013).
[4] P. Ungelenk et al., Proc. IPAC 2014, Dresden, 1936.
[5] http://www.beckhoff.de/twincat
[6] http://www.beckhoff.de/EtherCAT
[7] https://www.python.org
[8] http://www.djangoproject.com
[9] http://www.xenproject.org
[10] http://www.linux-kvm.org
[11] S. Hilbrich et al., Proc. FEL 2014, Basel, 255.
[12] https://www.mysql.com
[13] http://wamp-proto.org
[14] http://crossbar.io
[15] http://autobahn.ws/js
[16] https://github.com/Matthias247/jawampa
[17] http://freeradius.org
[18] https://pypi.python.org/pypi/pyepics
[19] D. Schirmer et al., Proc. ICALEPCS'09, Kobe, 741.
[20] http://wiki.qemu.org
[21] https://virt-manager.org
[22] https://www.debian.org
[23] http://freedesktop.org/wiki/Software/systemd
[24] https://wiki.debian.org/Debate/initsystem/sysvinit
[25] http://www.ansible.com
[26] B. Keil, Ph.D. thesis, TU Dortmund (2003).
[27] M. Höhner, Ph.D. thesis, TU Dortmund (2015).
[28] P. Towalski, Ph.D., TU Dortmund (in preparation).
[29] http://www.moxa.com
[30] http://www.windriver.com
[31] http://epics.web.psi.ch/software/streamdevice
[32] http://www.aps.anl.gov/epics/modules/soft/asyn
[33] B. Sawadski, Masters thesis, TU Dortmund (2015).
[34] Dennis Zimmermann, private communication.
[35] J. Grewe, M. Taghavi, DELTA Int. Report, 1+2/2014
[36] http://beckhoff.com
[37] http://www-csr.bessy.de/control/SoftDist/OPCsupport
[38] http://www.opcfoundation.org
[39] https://www.microsoft.com
[40] Carsten Mai, private communication.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

UPGRADES TO CONTROL ROOM KNOBS AT SLAC NATIONAL ACCELERATOR LABORATORY *

S. Hoobler#, S. Alverson, M. Cyterski, R. Sass, SLAC National Accelerator Laboratory, Menlo Park, CA, 94025, U.S.A

Abstract

For years, accelerator operators at the SLAC National Accelerator Laboratory (SLAC) have favored hardware knobs in the control room for accelerator tuning. Hardware knobs provide a tactile, intuitive, and efficient means of adjusting devices. The evolution of separate control systems for different accelerator facilities at SLAC has resulted in multiple flavors of knob hardware and software. To improve efficiency, space usage, and ease of use, the knob systems have been upgraded and integrated.

BACKGROUND

SLC Knobs

In the 1990s a knob system was developed for the SLAC Linear Collider (SLC), using hardware knob chassis designed by Miranova Systems. SLAC developed software and a DOS knob controller to interface the knob chassis to the SLC VMS control system [1].

EPICS Knobs

The Linac Coherent Light Source control system is based on EPICS (Experimental Physics and Industrial Control System). EPICS software was written to interface the existing knob hardware to the LCLS control system. At that time, it was presumed that the VMS control system would soon be decommissioned and no effort was made to integrate the two systems.

Knob Use at SLAC

To date, the VMS control system and its knobs are still used by the Facility for Advanced Accelerator Experimental Tests (FACET) and the Next Linear Collider Test Accelerator (NLCTA). EPICS knobs are used at LCLS, the SPEAR storage ring, and NLCTA.

Until recently, the VMS and EPICS knob systems were still not integrated, making an inefficient use of space and knob hardware. To improve this, the knob software and hardware have been updated to allow individual boxes to be used by multiple control systems simultaneously.

DESIGN CONSIDERATIONS

Backward Compatibility

To stay compatible with the legacy knob hardware and software, the new system supports the existing communication protocols.

*Work supported by U.S. Department of Energy under Contract Number DE-AC02-76SF00515

System Performance

The system should maintain the performance standard of the previous systems.

- Good knob resolution.
- Quick response.
- Allow same device to be assigned to multiple boxes simultaneously without conflict.
- Ability to restore device to original value.

Enhancements

To support operation of multiple facilities from the same location, the system was enhanced to allow a single box to be used with multiple control systems simultaneously.

Maintainability

The system will initially use the legacy protocols to remain backward compatible. As older systems become decommissioned, it may be desirable to migrate to other protocols. The software should be modular to facilitate this.

Flexibility

Unlike the legacy SLC control system, the LCLS EPICS control system is distributed. There is no pre-defined list of devices that may be attached to knobs. Any EPICS process variable may be attached. The software should be flexible to handle generic cases and should also provide support for device-specific settings and facility-specific conventions.

SYSTEM IMPLEMENTATION

A knob system consists of the elements shown in Figure 1. The Knob Controller handles all communication with the hardware knobs. The hardware knob boxes and the Knob Controller together act as a server to the Knob Users. The Knob Users are the accelerator control systems which wish to attach devices to knobs. The Controller is the interface between the Users and the hardware knobs and will accept requests from any User in the system.

In this implementation, the knob hardware uses a serial protocol while all software entities use Ethernet protocols. A terminal server performs the conversion between serial and Ethernet packets. An individual knob box has four knobs, each of which can be independently controlled by a different control system. A Knob Controller and terminal server can support up to 16 knob boxes.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

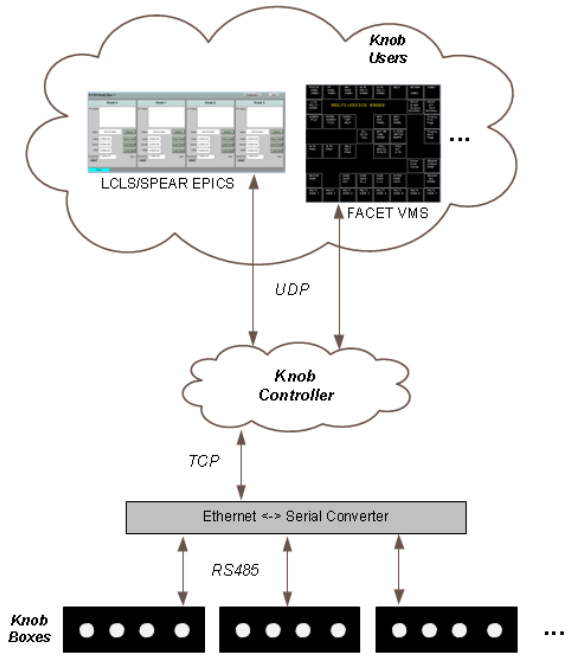


Figure 1: Knob system overview.

HARDWARE

The new knob box hardware is a small desktop chassis. It has four knobs, each with a LCD display, a disable button, and a button used to restore an attached device to its original value. Figure 2 shows a photograph of the new knob box hardware.

Figure 2: Photograph of the Knob Box Pro.

The knob box tasks are separated into two main processes, each of which is executed by a microcontroller, the 16 MHz PIC16F1937. One controller is dedicated to counting rotational ticks and determining direction from the knob optical encoders. The second handles communication with the various devices: the LCD displays, the knob encoder microcontroller, and the software Knob Controller, the latter of which uses the RS-485 serial protocol. This separation of responsibility helps prevent missed knob turn counts and improves the response time to the Controller.

In addition to the optical encoder, each knob also has two associated buttons. The first is a latching switch that toggles the knob between disabled and enabled (counts for that knob are 'zeroed' while it is disabled). The second button is a momentary switch which is coupled to a 555 timer integrated circuit. When depressed by the user for 5 seconds, the microcontroller will set the 'restore' flag in the message to the Knob Controller, causing the software to set the device back to its original value.

The box contains two rotary potentiometers to control brightness and contrast of the LCD screens. These are read back through the analog-to-digital converter on the communications microcontroller. Each box also contains an internal set of eight DIP switches which can be used for assigning an 8-bit hardware address for the RS-485 communication protocol. Figure 3 shows a block diagram of the knob box system.

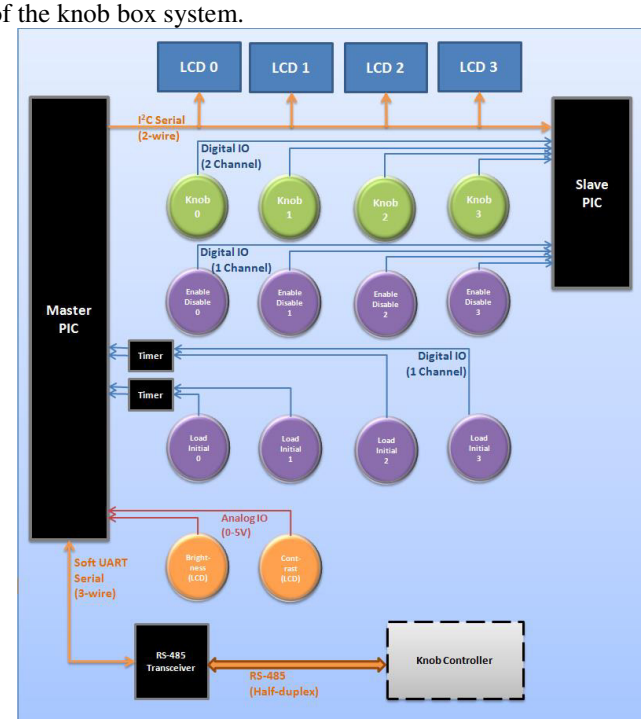


Figure 3: Knob box hardware block diagram.

SOFTWARE

The Knob Controller and EPICS Knob User are software processes running on Linux servers. They are comprised of 'C' code running in the context of EPICS IOC (Input Output Controller) processes. The legacy VMS Knob User is a combination of 'C' and Fortran software running on an Alpha.

Knob Controller

The Knob Controller handles communication with the hardware knobs. While devices are assigned to knobs, it polls the knobs for changes and sends the information to the Knob User. The Controller writes updated device information to the knob hardware display when requested by the User.

The Controller uses multiple threads to optimize performance, primarily by performing slow serial operations in parallel.

A single thread processes incoming messages from Users. It listens on a dedicated UDP port, awaiting requests. When a new attach request is received, the Controller assigns the requesting User to be the new knob

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

owner. It saves the User's information for future messages and identification. Other than attach requests, any message that does not come from a knob's owner is ignored. When a User takes ownership of a knob, the Controller notifies the previous owner.

Additionally, each knob box has a dedicated thread. It polls the attached knobs for changes, sending the results to the User. It receives and handles additional information from the box, including notification of a restore request, a recent reboot, and the box's revision information. These last features are only supported by the Knob Box Pro.

Figure 4 shows the architecture of the Knob Controller software.

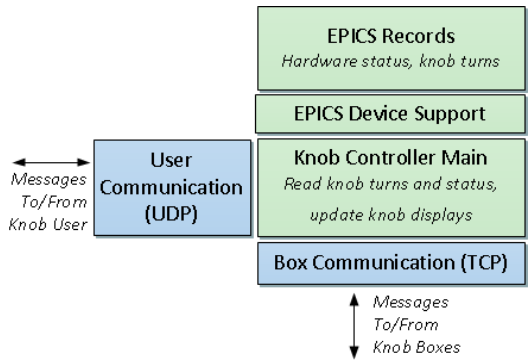


Figure 4: Knob Controller architecture. Blue sections are implemented in callbacks.

EPICS Knob User

The Knob User is specific to a control system. It provides the graphical user interface (GUI) for attaching devices to knobs. When a User receives knob turn information from the Controller, it converts knob turns to engineering units and sets the attached device to its new setting. In addition to attaching and detaching devices, the GUI provides the ability to modify and save knob sensitivity values and to save and restore device settings. Figure 5 shows the architecture of the Knob User software.

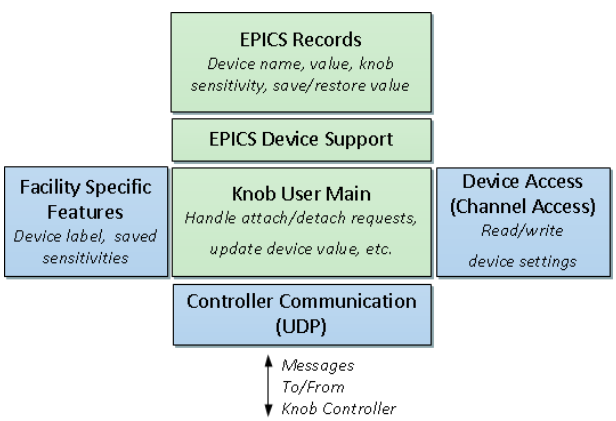


Figure 5: Knob User architecture. Blue sections are implemented in callbacks.

FUTURE IMPROVEMENTS

- Indicate on knob hardware which system owns each knob.
- Allow users to save device label to be used on knob display next time (this is already done for sensitivity).
- Allow users to save device precision to be used on knob display next time.

REFERENCES

[1] R. Johnson, R. Sass, "PC Knobs Software Design," December 1995, SLAC

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DESIGN AND COMMISSIONING RESULTS OF MICROTCA STRIPLINE BPM SYSTEM*

S. Hoobler#, R. Larsen, H. Loos, J. Olsen, S. Smith, T. Straumann, A. Young, C. Xu, SLAC National Accelerator Laboratory, Menlo Park, CA, 94025, U.S.A.
H. Kang, C. Kim, S. Lee, G. Mun, Pohang Accelerator Laboratory, Pohang, Kyungbuk, Korea

Abstract

The Linac Coherent Light Source (LCLS) is a free electron laser (FEL) facility operating at the SLAC National Accelerator Laboratory (SLAC). A stripline beam position monitor (BPM) system was developed at SLAC [1] to meet the performance requirements necessary to provide high-quality stable beams for LCLS. This design has been modified to achieve improved position resolution in a more compact form factor. Prototype installations of this system have been operating in the LCLS LINAC and tested at the Pohang Accelerator Laboratory (PAL). Production systems are deployed at the new PAL XFEL facility and at the SPEAR storage ring at the Stanford Synchrotron Radiation Lightsource at SLAC. This paper presents the design and commissioning results of this system.

BACKGROUND

At LCLS, stripline BPMs measure the transverse position of the electron bunch in the injector, Linac, and transport lines. Each BPM is instrumented with a SLAC-designed chassis that houses analog conditioning electronics, a digitizer, and a CPU. After digitization, the processed data is transmitted over a dedicated network to a VME processor that calculates pulse-by-pulse beam position. An online calibration scheme injects a tone into the system between beam pulses in order to compensate for gain variation and drift. This process occurs at the beam rate, which is 120 Hz.

For recent projects, it became desirable to use electronics in a smaller form factor while maintaining the performance standard of the LCLS design.

SYSTEM DESIGN

A new system was developed, using the MicroTCA (Micro Telecommunication Computing Architecture) crate architecture. In this system, electronics for several BPMs are housed within a single crate, with digitizers located in the front of the crate and the analog conditioning electronics in the rear. Digitized data is transferred along the PCIe backplane to the CPU for processing.

The first version of this system [2] was tested in the LCLS Linac and found to meet the performance requirements. Two of these BPMs have been deployed in a SPEAR transport line.

Later, this design was further modified to achieve improved position resolution for the PAL XFEL facility. Revised electronics, including a different operating frequency, larger bandwidth, and faster digitization rate, provide position resolution improved by a factor of two. Prototypes of this system were tested in the SLAC Linac and at the PAL Injector Test Facility (ITF). Hardware design details presented in this paper are for this improved design.

Analog Front End

The analog front end (AFE) is used to reduce the signal bandwidth and set the signal level before digitization. To achieve the desired system resolution at low bunch charge, it is important to minimize losses in the system. This is most important in the signal cables and the AFE components up to the first amplifier.

The SLAC-designed AFE contains two stages of amplifiers and attenuators. The attenuators can be adjusted from the control system to maintain reasonable signal level when the bunch charge changes. The bandpass filters select a processing frequency of 300 MHz with a bandwidth of 30 MHz. Figure 1 shows a block diagram of a single input channel of the AFE.

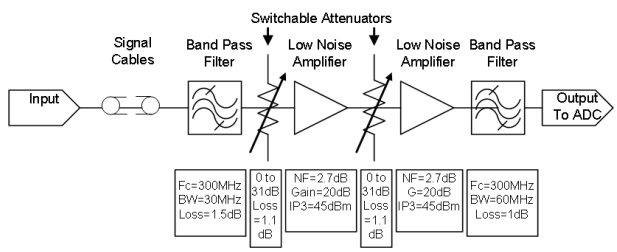


Figure 1: Analog Front End block diagram.

As in the LCLS system, the AFE performs a self-calibration process between beam pulses. A 300 MHz tone is injected into a single stripline and the coupled response is measured from the two adjacent striplines. This is performed for both the horizontal and vertical transverse planes. The calculated calibration ratios are applied to the beam position calculation to compensate for thermal drift and gain variation among channels. Figure 2 shows a block diagram of the calibration scheme.

*Work supported by U.S. Department of Energy under Contract Numbers DE-AC02-76SF00515 and WFOA13-197

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

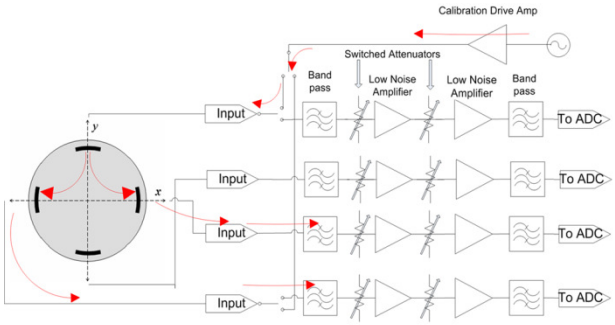


Figure 2: Calibration scheme.

Digitizer

To meet the resolution requirements, the analog-to-digital converter (ADC) must provide sufficient effective number of bits and signal-to-noise ratio. A commercial off-the-shelf product, the Struck SIS8300, was selected.

It is important to select a digitization rate that centers the signal well within the Nyquist zone. This reduces unwanted signal leaking in from neighboring zones, which decreases signal quality. This effect is also improved by good filtering and selection of analog bandwidth. In this system, the 300 MHz BPM signal is undersampled at 250 MSPS, which centers the 30 MHz-bandwidth aliased signal well in the first Nyquist zone. Figure 3 shows the 300 MHz BPM signal, the 250 MSPS digitization rate, the Nyquist zones, and the frequencies of the aliased BPM signal.

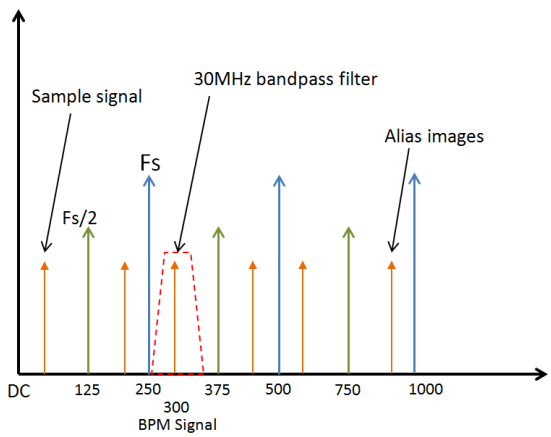


Figure 3: Nyquist zones and aliased signals.

Timing

The Micro Research Finland Oy (MRF) PMC Event Receiver Card (EVR) is used to receive global timing data and to provide hardware triggers for data acquisition.

Data Processing

The system uses embedded Linux with a real-time patch to provide deterministic processing. New software drivers were developed for the Struck digitizer and the new AFE. These new software components were integrated into the existing BPM application. Figure 4 shows the BPM software architecture.

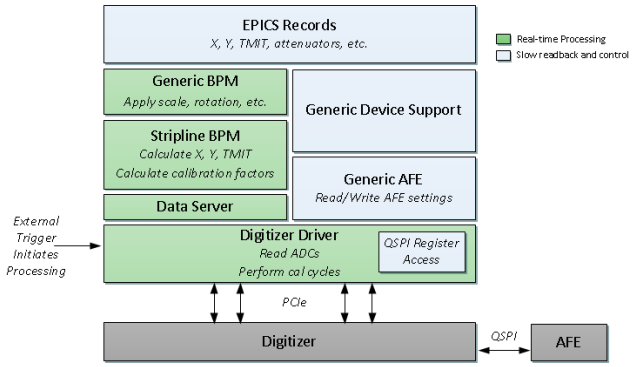


Figure 4: BPM software architecture.

The BPM data acquisition and processing is event-driven. At beam time, an external hardware trigger from the EVR initiates the beam data acquisition. Once this is complete, the software internally triggers a calibration cycle. To reduce overhead, a DMA engine is used to transfer data to the CPU. The software receives an interrupt when the transfer is complete. Figure 5 shows the process flow of two consecutive acquisition cycles.

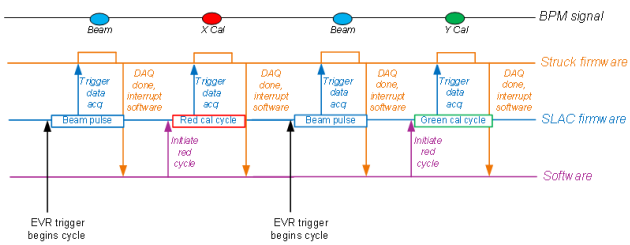


Figure 5: Data acquisition cycles.

The AFE firmware state machine executes the calibration routine and initiates digitizer data acquisition. The AFE has direct connections to the digitizer for real-time data acquisition handshaking. It also provides a QSPI interface to the digitizer for slow monitoring and control. Figure 6 shows the connections and data flow between the AFE and digitizer.

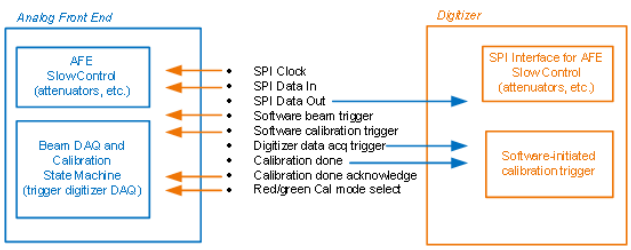


Figure 6: AFE-Digitizer interface.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INITIAL RESULTS

Prototypes of this system have been tested in the LCLS Linac and at the PAL ITF [3].

Figures 7 and 8 show the measured beam position resolution of LCLS Linac BPMs, including the prototypes. The four red data points show the prototype data. The two points on the left are early prototypes; the two on the right are later prototypes with improved resolution.

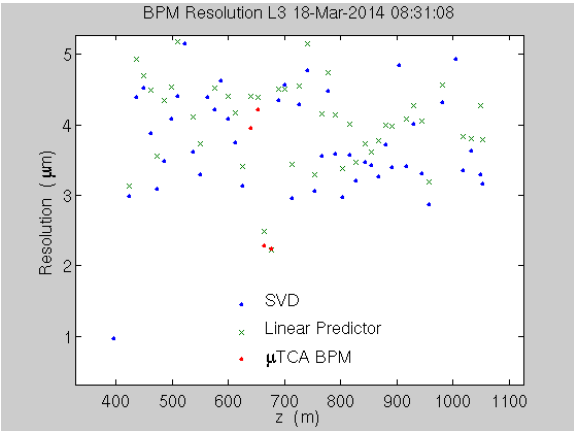


Figure 7: LCLS Linac BPMs measured beam position resolution at 160 pC bunch charge.

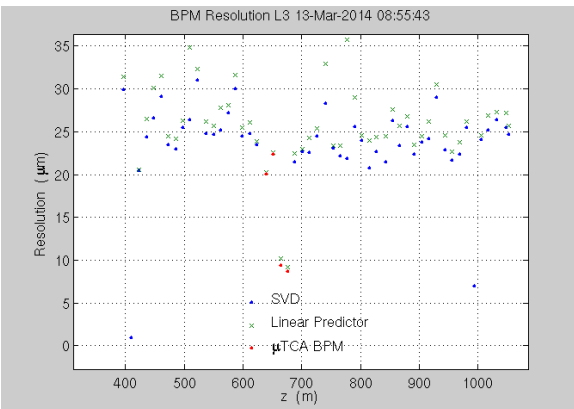


Figure 8: LCLS Linac BPMs measured beam position resolution at 20 pC bunch charge.

Figure 9 shows data taken at LCLS, plotting the position resolution as a function of bunch charge, from 20 pC to 150 pC. Within the dynamic range of the system, the position resolution should increase linearly as bunch charge decreases, until it reaches its resolution limit. Units BPM27201, BPM27301, BPM27401 are early system prototypes. The other two are legacy LCLS BPMs. The PRD limit shows the resolution requirement for the early prototype.

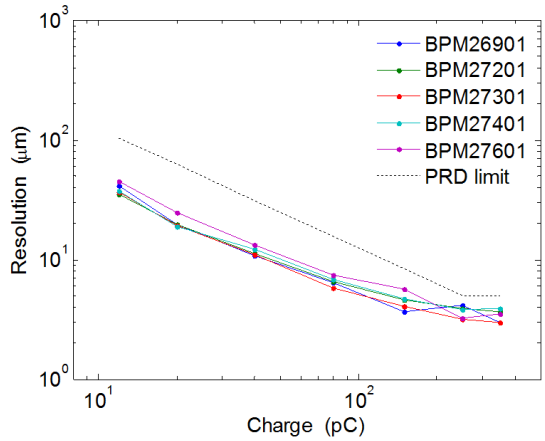


Figure 9: LCLS Linac BPMs dynamic range.

The BPMs at PAL ITF are on movers, which can be used to compare measured position as a function of actual position. The BPM system is expected to be linear within +/- 1 mm from the BPM center in the X and Y transverse planes. Figures 10 and 11 show X and Y data for a single BPM.

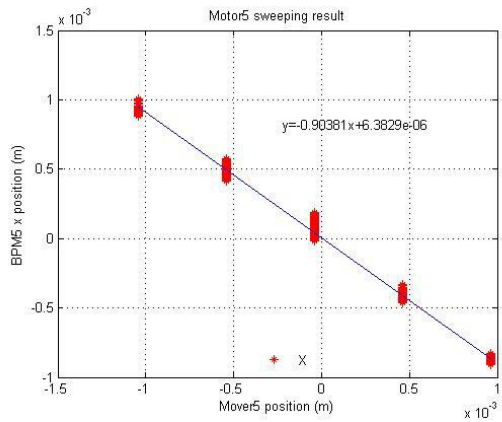


Figure 10: PAL ITF BPM measured versus actual X position.

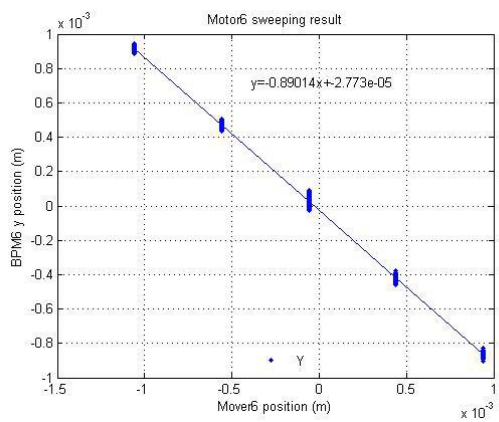


Figure 11: PAL ITF BPM measured versus actual Y position.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

FUTURE USE

The PAL XFEL facility will use this system for its 144 Stripline BPMs. This system will be commissioned with beam in early 2016.

SPEAR is currently using two of these BPMs in its transport line and plans to install more in early 2016.

REFERENCES

[1] T. Straumann et al., "LCLS Beam-Position Monitor Data Acquisition System", WPPB40, Proceedings of ICAELPCS07, Knoxville, Tennessee.

[2] C. Xu et al, "Design of an Ultra-Compact Stripline BPM Receiver using MicroTCA for LCLS-II at SLAC", WEPC23, Proceedings of IBIC2013, Oxford, UK.

[3] C. Xu et al, "Commissiong Results of MicroTCA.4 Stripline BPM System", WEPD17, Proceedings of IBIC2014, Monterey, CA, USA.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TIP: AN UMBRELLA APPLICATION FOR ALL SCADA-BASED APPLICATIONS FOR THE CERN TECHNICAL INFRASTRUCTURE

Ph. Gayet, P. Golonka, M. Gonzalez-Berges, L. Goralczyk, J. Pache, P. Sollander, F. Varela[#], CERN, Geneva, Switzerland

Abstract

The SCADA package WinCC Open Architecture (WinCC OA [1]) and the control frameworks (JCOP [2], UNICOS [3]) were successfully used to implement many critical control systems at CERN. In the recent years, the Industrial Controls and Electronics (ICE) group of the Engineering Department (EN) at CERN, supported other groups to re-implement the supervision of technical infrastructure, like Electrical distribution (EL), and Cooling and Ventilation (CV), using these tools. However, the fact that these applications are highly independent from the operation point of view, as well as their increasing number, renders operation uncomfortable since shifters are forced to continuously switch between them. In order improve the integration, EN-ICE is developing the Technical Infrastructure Portal (TIP) to provide centralized access to all WinCC OA applications, and extending their functionality including links to external databases and to a powerful localization system based on GIS. In addition the tool offers an environment for operators to develop views that aggregate data from different sources, such as cooling and electricity. This paper also describes the challenges faced during the implementation of TIP due to the large degree of heterogeneity across applications, although they are made out of common building blocks.

INTRODUCTION

For the construction phase of the LHC, a big emphasis was put in the use of industrial tools for the control systems. This was complemented with the development of frameworks that provided the necessary adaptations to the CERN environment (JCOP, UNICOS). This approach showed to be very successful and many systems were built following it in both the experiments (e.g. Detector Control Systems, Detector Safety Systems, Gas Control, etc.) and the accelerators (e.g. Cryogenics, Machine Protection, Vacuum, etc.). The result was a big gain in manpower and a high homogenization across systems.

Building on this success, an effort was started in 2009 to use the same tools for the CERN infrastructure systems. These systems cover areas like Cooling and Ventilation, Electrical Distribution or Radiation Monitoring. There is now a large number of them that have benefited from this standardization effort, and there is an ongoing process to converge to the same technologies for most of the applications, if not all. The homogenization that results from this convergence process, opens the door to further integration at the SCADA layer and better operations efficiency. This paper shows the first initiative to build a hypervisor system on top of the existing applications across several domains for this purpose.

[#]fernando.varela.rodriguez@cern.ch

ISBN 978-3-95450-148-9

OPERATION OF THE CERN TECHNICAL INFRASTRUCTURE

Operators on shift in the CERN Control Center (CCC) are in charge of monitoring the Technical Infrastructure (TI) 24 hours a day, 365 days per year. Their duties are to detect problems, make first diagnostics and decide on actions to take. The TI encompasses the entire CERN electrical distribution network from 400 kV down to 48 V, the CV installations for all accelerators, detectors and other important facilities such as the computer centre and other systems (controlled access, safety monitoring systems, etc.). Anomalies are detected through a centralised alarm display, LASER [4], to which all systems across CERN are connected. Diagnostics are made using the application specific supervision systems.

There are more than 100 standalone CV systems around the CERN sites and their corresponding supervision systems have been developed over time with different technologies and by different people. This makes difficult for the operators to treat each non conformity as there is no standard way to establish a correlation between an alarm reported in the LASER console and the corresponding synoptic view of the process experiencing the problem. This requires that the operators have a very detailed knowledge of the various systems and makes the learning curve for newcomers very steep. Moreover, each of these applications requires individual login making daily life very tedious for the operators.

TI PORTAL

Main Requirements

TIP shall centralize the access to all WinCC OA applications deployed for the CERN TI simplifying the work of the operators in the control room. TIP must provide access to all synoptic views and trend plots available in every remote application, and must use the access control schemas locally defined in each of these applications:

- a single log-in shall be sufficient to grant the operator access to every single remote application,
- the operation can be performed by both the equipment group expert and the central operators with privilege associated to their level of expertise,
- the data shall be presented to the operators and experts in exactly the same way,

Copyright © 2015 CC-BY-3.0 and by the respective authors

- the auditing functionality shall differentiate if the actions on the equipment were initiated by the equipment team or by the TI operators.

As the operation in the control room is driven by critical or summary events presented to the operators through the LASER console, TIP must grant access to every single alarm generated by the remote systems, it must also differentiate unambiguously LASER alarms from detailed process alarms.

TIP shall not disturb the remote systems and minimize the impact on their performance. Additionally, it must integrate the remote systems as they are, i.e. no changes to the remote applications shall be required to connect them to TIP.

It shall be possible to reconfigure dynamically TIP to cope with changes done in remote applications: the migration of many controls applications (e.g. CV) is a continuous process that will span over many years. At any moment, old applications can be ported to WinCC OA and enter production. Moreover, due to the fast pace evolution of some systems, like the CERN electrical network, control applications may be reconfigured at any time. TIP shall be able to adapt to these changes with minimal effort, i.e. TIP must be able to discover new applications, to detect the changes in existing ones and to configure the hooks necessary to handle the new setup centrally. Moreover, the re-configuration of TIP shall be as transparent as possible to maximize the availably of the application since it is required 24/7.

The portal application shall also allow TI operators to create their own views and combined trends where they combine data from different sources to display dependencies between systems.

TIP shall also provide means to easily integrate data published by non-WinCC OA sources as the data required for operation are gathered using different technologies.

Architecture

Figure 1 presents the architecture of TIP, the connections with other systems, the different kind of operators and illustrates its central role in integrating data from various sources. TIP is part of a distributed WinCC OA application. This native feature of the SCADA allows access from TIP to all data from the remote systems and offers seamless display of views without any extra effort. Other systems can be added to and removed from this distributed network at any time giving flexibility and future-proof design.

TIP is built using the same control frameworks and share the same GUI principles as the remote applications giving a familiar environment to the operators and drastically shortening learning curve.

TIP makes usage of a local file repository which is periodically synchronized with the file systems of the remote applications such that TIP always has an up-to-date copy of the scripting libraries and user interface files.

Another important feature is the possibility to access historical measurement data. This is achieved by connecting TIP to the central ORACLE database used by the applications.

CERN network database (LANDB), is also used by TIP to provide detailed information about devices, like displaying the geographical location of the PLCs in the CERN facilities.

Finally, the Alarm Help application is used by the TI operators to display very detailed information about LASER alarms like intervention procedures, list of affected devices, consequences, etc. TIP interacts with this application bi-directionally to provide navigation from the LASER alarm to the remote WinCC OA synoptic view of the process and vice-versa.

User Interface

It is necessary to acknowledge that many of the features of TIP presented here are inherited from the underlying frameworks and the distributed nature of WinCC OA which greatly decreased effort needed to develop the application. Figure 2 shows an example screenshot from TIP main user interface. The top area is occupied by the TIP Global Toolbar, which enables access to general functions of the application i.e overall alert screen that shows alarms from every single connected WinCC OA system.

The information in TIP is organized into different views. For example, in Figure 2, the CV process information such as status of pumps, flows, pressures, etc. are shown in the first tab, whereas the status of the readout systems, PLCs, data-servers, WinCC OA projects, are shown in the infrastructure tab. TIP features multiple points of view targeted to different users, in order to provides a window to a subset of the information handled by the application.

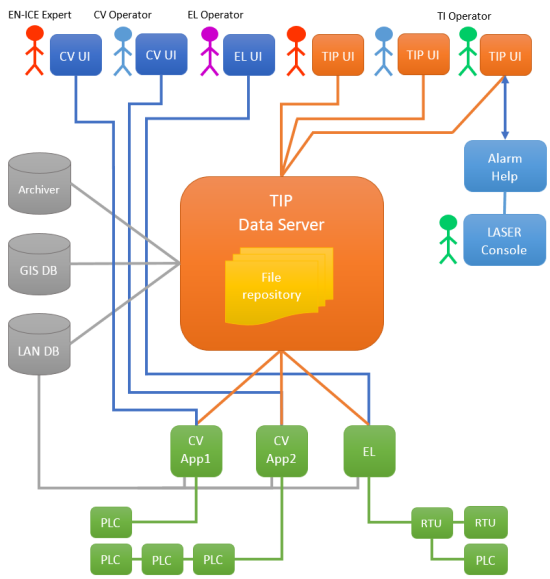


Figure 1: TIP architecture and context.

The information is organized in a tree-like structure where leave nodes in the tree represent a set of LASER alarms and the systems applications (here CV) whereas top level nodes group them geographically. Each node in

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

this tree used colours that represent the presence of active LASER alarms. The colour of the nodes is propagated all the way up enabling operators to easily localize active alarms by expanding the tree. Selecting a node in the tree causes the main area of TIP to be updated with the synoptic view associated to the node.

Figure 3 and 4 show examples of graphical interfaces, which are displayed as the operator navigates in the tree. Figure 3 shows the main operational parameters of the LHC CV systems. TIP makes usage of modern visualization techniques like spiderweb graphs and the Javascript visualization package Highcharts [5] whenever applicable to enrich the user experience. This was achieved by extending the WinCC OA HMI functionality to integrate web technologies as presented in [6].

Figure 4, shows some of the main parameters involved in the safety of the Point 1 of LHC. These user interfaces merge information from the electrical network, elevators, the LHC access control system, and the status of the Oxygen Deficiency Monitors.

GIS Integration

The TIP GIS Module combines geographical data with LASER alarm location information. The buildings shown in the map are also coloured according to the presence of active LASER alarms. This provides operators with very valuable information on the location of the problem that helps in the discussions with the intervention teams. Navigation is also possible using the GIS Module. By clicking on a red or green location it will cause the module to zoom on this area and display applications that are present in this particular location. The operator selection of a node in the impact tree also causes the GIS Module to be updated by zooming on the area where application, or group of applications, is located.

LASER Integration

The LASER Alarm Info box shows detailed information about the alarm like its description, unique identifier for equipment, localization and responsible.

Although TIP focusses on LASER alarms, using the Remote Application Toolbar, operators can interact directly with the remote application and access its Alarm Screen, pre-configured trends or check the PLC statuses.

Challenges Faced

All applications developed by EN-ICE are using common building blocks: WinCC OA as SCADA system and the JCOP and UNICOS frameworks. Moreover, within a particular application domain, e.g. CV, all applications are largely similar and homogeneous. However, fundamental differences may exist between domain, e.g. between CV and EL. In such cases, applications may implement different device and alarm models, together with distinct access control schemas.

Furthermore, the mapping between devices, alarms and associated synoptic views may also differ depending on the application domain. To ease the creation of similar portal applications, a central database containing these metadata would be desirable. Therefore an effort is required to improve the coherency across application domains.

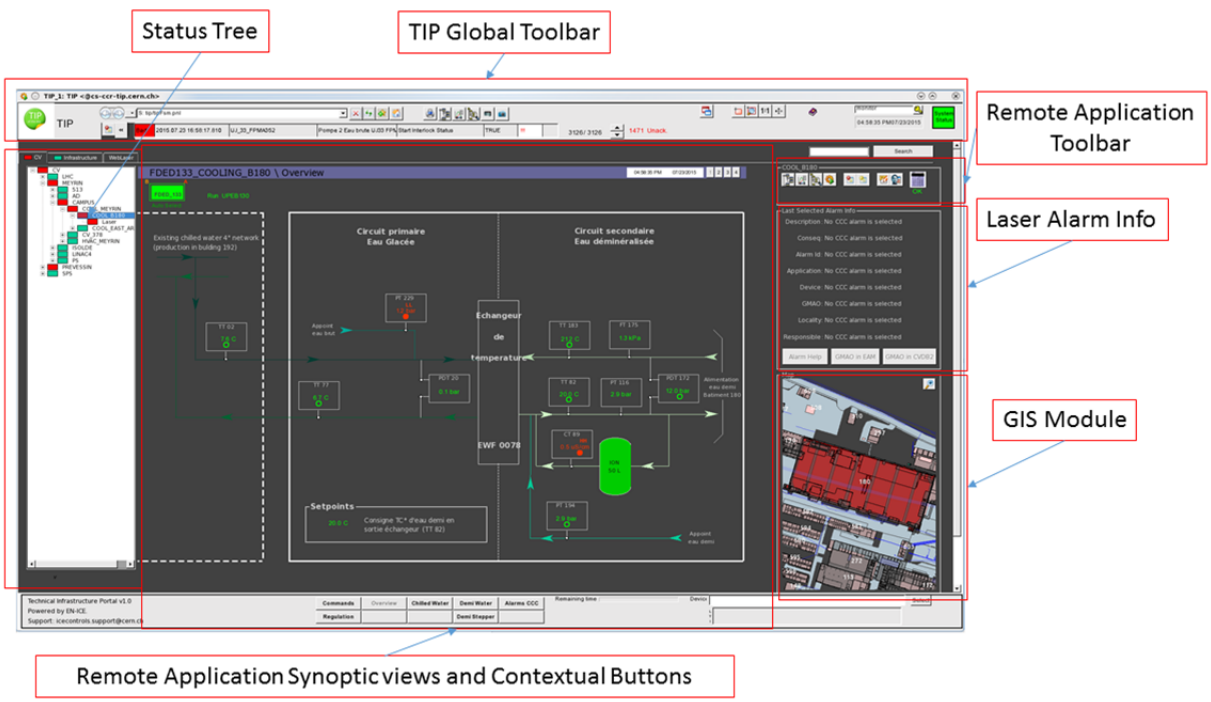


Figure 2: Main user interface of TIP.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

During the implementation of TIP, it was also necessary to extend the functionality of existing tools to be able to work on large distributed systems. These enhancements will now be integrated in the core functionality of the package.

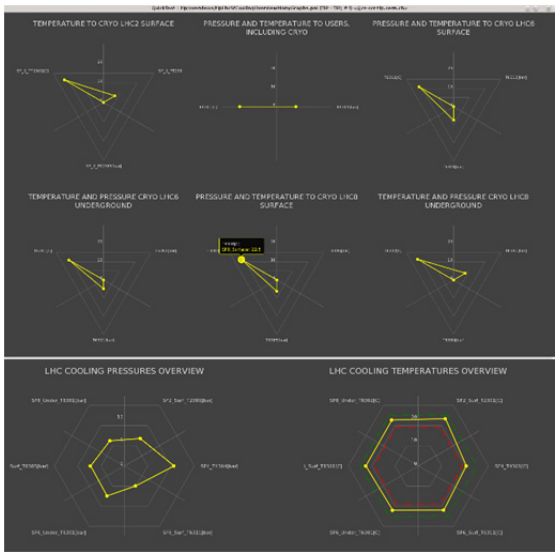


Figure 3: TIP Dashboard example.

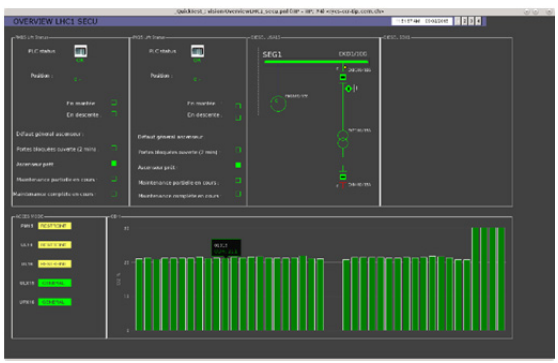


Figure 4: Compound User Interface of TIP showing information from multiple safety systems of the LHC.

STATUS AND OUTLOOK

TIP entered production in Summer 2015 and despite of the short time in use, it has already proven to be a significant improvement for the operators of the CERN TI and also for the equipment groups and the EN-ICE experts. Table 1 summarizes at the time of writing this paper, the applications that have been integrated in TIP and the number of synoptic views that can currently be accessed from TIP.

In the coming months we plan to extend the TIP to cover new domains (Radiation Monitoring system,...), and to show dependencies across different domains. In addition, TIP will be used to speed up the integration of all CERN CV systems including the legacy ones into a single tool. This is motivated by the need to provide a unique operation solution despite the long timespan of the migration process that is expected to be finished in 2025. Hence, the set of non-WinCC OA data accessed by TIP will be significantly enlarged.

Table 1: Applications under the Umbrella of TIP

Application Domain	Controls applications	Synoptic views	LASER alarms
Cooling & Ventilation	137	1846	1364
Electricity	1	211	13719
Lifts	5	0	0
Monitoring	1	335	0

To conclude, this promising start for the CERN TI, has awaken the interest of developing similar umbrella applications for many other domains like the LHC Cryogenics or Power and Interlock systems.

ACKNOWLEDGMENT

The authors of this paper would like to thank Benjamin Bradu, Jean-Charles Tournier and very especially to William Booth for all input and feedback provided during the implementation of the TIP.

REFERENCES

[1] Siemens Simatic WinCC Open Architecture website: http://etm.at/index_e.asp
[2] M. Gonzalez-Berges *et al.*, "The Joint COntrols Project Framework", CHEP'03, La Jolla, California (2003).
[3] Ph. Gayet, R. Barillere, "UNICOS a framework to build industry like control systems: Principles & Methodology", ICALEPCS'05, Geneva, Switzerland (2005).
[4] F. Calderini *et al.*, "Moving towards a common alarm service for the LHC era", ICALEPCS'03, Gyeongiu, Korea (2003).
[5] Highcharts website: http://www.highcharts.com/
[6] P. Golonka *et al.*, "FwWebViewPlus: integration of web technologies into WinCC OA based Human-Machine Interfaces at CERN", CHEP'13, Amsterdam, The Netherlands (2013).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

KECK TELESCOPE CONTROL SYSTEM UPGRADE

K. Tsubota, J. A. Mader, W. M. Keck Observatory, Kamuela, Hawaii

Abstract

The Keck telescopes, located at one of the world's premier sites for astronomy, were the first of a new generation of very large ground-based optical/infrared telescopes with the first Keck telescope beginning science operations in May of 1993, and the second in October of 1996. The components of the telescopes and control systems are more than 15 years old. The upgrade to the control systems of the telescopes consists of mechanical, electrical, software and network components with the overall goals of improving performance, increasing reliability, addressing serious obsolescence issues and providing a knowledge refresh. This paper is a continuation of one published at the 2013 conference [1] and will describe the current status of the control systems upgrade. It will detail the implementation and testing for the Keck II telescope, including successes and challenges met to date. Transitioning to night time operations will be discussed, as will implementation on the Keck I telescope. Throughout this paper the telescope control system will be referred to as TCS.

TCS OVERVIEW AND CONTEXT

Each of the Keck telescopes is over 8 stories tall, weighs in excess of 300 tons, uses an alt-az mount and has the equivalent of a 10meter mirror which is comprised of 36 individual segments. A general diagram is shown in Fig. 1. The telescope supports a wide range of instruments at various focal locations in addition to providing both a natural guide star and laser guide star adaptive optics.

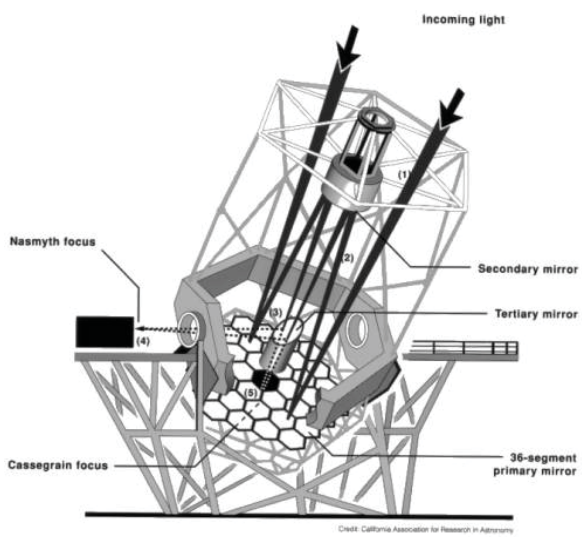


Figure 1: Keck Telescope.

TCS consists of a supervisor and set of subsystems that work together to provide Status and Control of:

- Telescope mounts
- Dome and shutter positions
- Facility rotators
- The secondary mirror

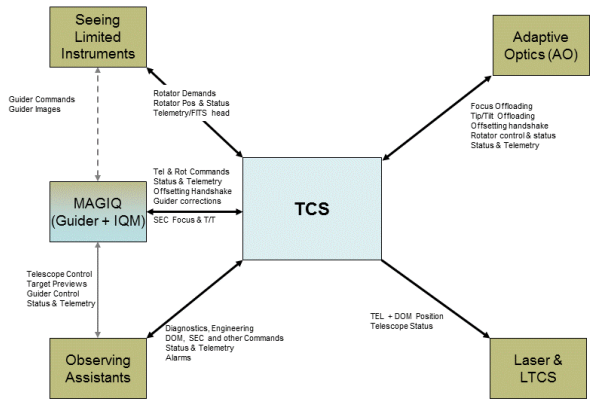


Figure 2: TCS System Context.

Figure 2 shows a high level context diagram for the Telescope Control System (TCS). It and the other Observatory principal systems interact through well-defined interfaces to accomplish the desired behaviour. Systems are tied together by the use of an Ethernet Bus. The main purpose of the TCS software is to accept the target position of a celestial object (which can be given in a variety of coordinate systems) and then calculate the mount, rotator and dome/shutter positions, so that the target is imaged perfectly at a given point in the focal plane. Furthermore, the TCS is characterized by the need to integrate a number of heterogeneous subsystems, which exhibit complex interactions. These interactions, although not hard real-time bounded, need a high level of synchronization. All this has to be done in a manner that protects the staff and equipment.

TCS operates the mechanical components of the telescope through a number of computer subsystems which control the telescope mount assembly, the dome enclosure, secondary and tertiary mirrors and facility rotators. The subsystems are responsible for the actual servo or hardware control, while the TCS coordinates their activities.

- The Telescope Axes (AXE) subsystem controls the telescope mount assembly, including the elevation and azimuth drives and other components associated with the mount operation.
- The Dome subsystem (DOM) operates the enclosure, which includes the dome carousel, dome lights and shutters, and other auxiliary equipment.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- The Secondary (SEC) subsystem controls the secondary mirror, including its tip-tilt-focus and thermal requirements. It also handles the monitoring of thermistors placed around the telescope tube which is used for focus compensation.
- The Rotator (ROT) subsystem controls the cassegrain, forward cassegrain, bent cassegrain and nasmyth facility rotators and positions the tertiary.
- The Telescope Safety System (TSS) handles the safety interlock system. It includes items such as Estop, limit switches and interlock processing and is implemented using a PLC. There are multiple interlocks including module handler deployed, crane deployed, horizon lock deployed, and so on.
- The Timing subsystem (TIM) handles time synchronization for each subsystem in addition to providing accurate triggers and time conversion routines

PROJECT OVERVIEW

Keck follows a standard development process that includes concept, preliminary and detailed design, full scale development, followed by integration and test, and commissioning. The TCS Upgrade project is at the end of the integration and test phase for the Keck II telescope and working towards commissioning. This will be followed up with integration and test of the Keck I telescope

A philosophy for the TCS upgrade project was to remain backwards compatible with the existing system. This was particularly critical for instruments and clients that were expected to communicate with TCS. As a result of this philosophy the project chose an architecture that uses proven hardware and software, is primarily COTS, leverages the open source community, is inherently backwards compatible, is viable for 10+ years and allows the subsystem controllers to be upgraded without affecting I/O. It was a conservative approach; all existing top level EPICS records can be reused, there is 100% reuse of the Keck Task Library and, if necessary, all existing UIs and tools can continue to operate.

Another key philosophy was to minimize the need for any telescope down time and to allow continuous night time observing. This meant allowing the use of both systems in parallel and to provide a failsafe fall back to the old system during commissioning. The overall approach taken to minimize telescope downtime was to implement a physical switching solution.

IMPLEMENTATION

The new redesigned telescope axis encoder systems have been successfully implemented and tested on Keck II and are currently being installed on Keck I. The EL design uses existing surfaces for mounting two read heads to a rigid yoke main triangle structure. The AZ design uses a compact, on-axis annular design with four read heads mounted to rigid and fixed structure at the center of the telescope structure. This second generation simplified encoder design has saved the project considerably in cost and effort by being easier to fabricate and install with no telescope downtime as compared to our original design. It also allows the new encoding system to run in parallel with the operational encoding system for a side by side comparison.

From the beginning TCS had a requirement to minimize operational downtime. This led to the design and implementation of a switching solution [2] to allow quick and easy switching between the operational system, referred to as Drive and Control System (DCS), and TCS. The switching solution and all significant preparation work such as cable pulls was performed well before summit I&T.

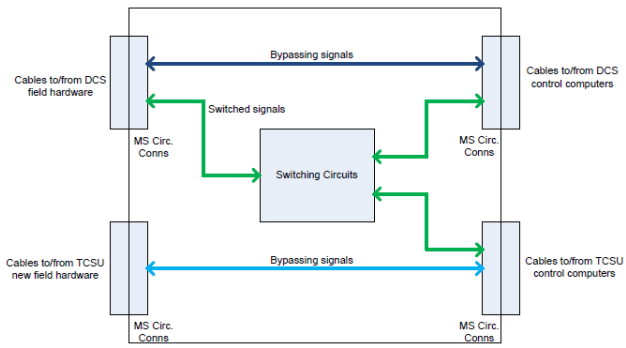


Figure 3: Switching Solution.

Figure 3 shows the basic switching implementation. The switching solution was very easy for the secondary and basically consists of cutting the existing cables and terminating them with connectors so that they can be switched between DCS and TCS with the addition of relay switches. The solution for AXE and ROT was a bit more complicated and relied on banked sets of DIN mounted relays to be switched. Switching occurs at the subsystem level. The number of switch points was reduced by utilizing the ability to simply run signals such as limit switches etc. in parallel between DCS and TCS. This not only reduced cost but also reduced implementation complexity.

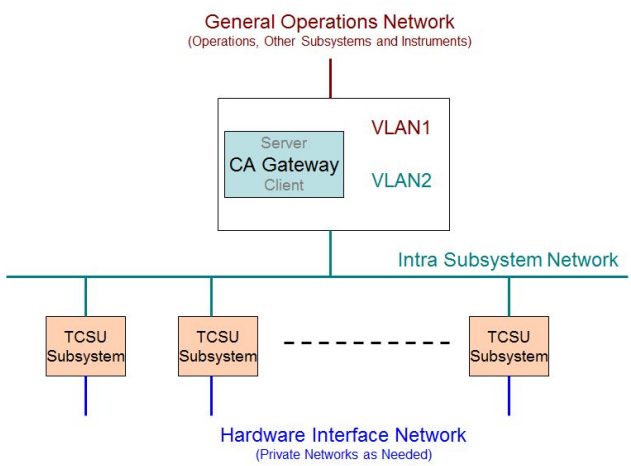


Figure 4: Network Layout.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Software also has a similar switchover concept. Fig. 4 above shows the basic network layout. TCS is implemented on a separate network and tied to the operational network by a single supervisory server with dual network interface cards (NIC). One of the NICs is on the operational network and the other on the TCS network. During standalone TCS testing all subsystem EPICS IOCs communicate through an EPICS gateway with a non-standard EPICS channel access server port running on the dual NIC server. When switching the software over to operations, the operational VxWorks control systems are halted and the TCS EPICS gateway changed to to communicate on the standard port. This allows all existing operational clients access to the new TCS control systems.

TESTING

Testing is a critical aspect of system development. Once components have been built and unit-tested, they must be tested in relation to the system as a whole. Two general principles were followed: test functionality and performance at the lowest level possible. When it could be avoided no hardware or software whose functionality and performance had not been verified was integrated. Of course it wasn't always possible to follow these principles, e.g. sometimes software must be integrated with hardware before either can be meaningfully tested.

TCS identified various levels of testing and are briefly described here.

- Developer testing
- Hardware Checkout
- Unit Testing
- Functional Unit / Subsystem Testing
- System/Acceptance Testing

Starting from developer testing and hardware checkout, each subsequent layer essentially builds on the last providing more complete and encompassing testing. Initial testing took place at our HQ lab environment where the basic hardware was installed with minimal functionality followed by summit testing after everything was installed in its final configuration.

An official deliverable of the testing phase is the documentation of the requirements acceptance testing. For each requirement the project developed a standard test acceptance form and sorted them by subsystems. In cases where one requirement spanned multiple subsystems, each subsystem had a copy of the test acceptance to complete. The major sections of the test acceptance form are the requirement to be tested, the test procedure, the test results and disposition, the test conductor name, and the date of the test. As the tests are completed and documented, they are checked off in the requirements matrix and the individual sheets are compiled into a system notebook.

SUCCESSES

The TCS Upgrade project has had a number of successes that has proven invaluable. The implementation of the hardware switching solution allows quick and easy switchovers during the day and at night. Since switchover involves three switches and a couple of cable moves and a simple procedure, anyone not familiar with the system can do it.

Implementing the switching solution on a subsystem by subsystem basis also gave more freedom in testing. It allowed TCS to selectively switchover a single subsystem as needed, especially if another subsystem was not available because of operational needs.

From the software perspective the use of a separate network and EPICS gateway implementation to isolate the software and make it public only when needed by switching the port number allows the project to run the software in parallel day and night.

Insisting on software backwards compatibility freed the project from having to change and re-validate existing client applications, of which there are many.

All these have led to successfully integrating and testing the full TCS system prior to on-sky engineering.

CHALLENGES

During the integration phase on the summit, considerable planning and coordination with operational tasks and projects as well as for personnel support was involved. There are processes in place for this but it still took a more time and effort than originally planned. Understandably, operations are the highest priority and could bump anything on the schedule for preparation of the nights observing. Also the longer it took for integration the more TCS had to deal with competing projects for telescope access.

Another challenge was getting on-sky test time. These nights are scheduled months in advance and you need to ensure you will be ready for the testing. Missing a night of testing could mean a month or more delay. Not being fully ready but going on-sky is also a waste of resources. There was one night where the project team thought they were prepared but failed to fully test all interfaces to the existing system and the project ended up dead in the water when a part our target selection tool interface failed after it had to be restarted and it was too late to give back for other projects to make better use of the time. The interface issue was eventually resolved the next day but it had already prevented the project from doing the tests that were originally planned. There is also the weather for which we don't have much control over and that the project has experienced this first hand. Just recently this past August, two scheduled partial back to back nights where weathered out preventing any TCS upgrade on-sky validation. To compensate for weather delays, this the project has requested contingency nights.

Another challenge the project ran into and is still dealing with is the tuning of our telescope axis closed loop servo performance. By design, the internal velocity loop was digitized with a FPGA which allow for much more freedom and sensitivity and driven with a 100Hz position loop. While attempting to tune the servo loop, an intermittent timing issue with our 100Hz event trigger

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

would crop up. This was eventually traced down to a termination issue in the initial signal generator setup. This was difficult to trace because whenever probes were added to monitor the signal it would change the loading in a positive way and generate a solid 100Hz signal so the problem never occurred. It took nearly a full week to get to the bottom of the issue. The bad part was that this was not the final operational configuration. The signal generator was initially setup as a way to help facilitate testing but ultimately it costed the project more time. The final configuration was to have the timing card (Bancomm 635 with a crystallized oscillator) internally generate the 100Hz event signal. Another challenge was dealing with investigating and resolving a 1Hz period oscillation in our servo performance. The FPGA velocity loop has been looking at the tachometers as its feedback. These tachometers have been noisy in the past and cleaning them have helped. However, it was decided to add a derivative term to our servo loop to supplement the use the tachometers. Another option is being implemented to replace the tachometer feedback with an encoder velocity feedback calculated in software and feeding into the FPGA in place of the tachometer feedback.

TRANSITION TO OPERATIONS

As the TCS Upgrade Project nears completion of its Keck II performance testing, a path for transitioning to operations is being actively developed. As part of this transition, Keck standards dictate that all projects undergo an Operational Readiness Review (ORR) prior to release. This is an external review of our readiness for operations by documenting that all requirements and performance specifications have been satisfied. It also verifies that all necessary as-built design documents, procedures is complete and in place for handing over to operations. The ORR also shows that all required training has been completed and that the project has proven that the system is stable and can be reliably operated by documenting the engineering and shared risk nights and results.

CURRENT STATUS

Currently all Keck II telescope subsystems (AXE, ROT, SEC, DOM, TSS, TIM) have been fully integrated and tested as designed.

The full TCS system has been successfully run on-sky at night and the pointing model based on pointing data test results using actual stars was completed and updated.

Fine tuning our telescope axis servo performance continues as well as completion of all remaining test procedures. Completion of all as-built documentation and procedures continues as well as identification of and training preparation. The Keck II TCS Upgrade Project ORR is scheduled for December 2015 with plans to fully deploy the Keck II TCS upgrade in late January 2016.

Performance

Some of the performance requirements are very tight and difficult to achieve, even if the system were to have been designed from scratch. As shown in Table 1, the current system does not meet the original settling time requirement for offset moves but has improved it over the current control system. With the TCS upgrade, the settling times for small moves in azimuth using the double path with feedforward turned on are nearing the original requirements and have met them in some cases. Double path means that the base position is separated from the offsets and the two are applied separately. In elevation, performance is close between the single path with feedforward and the double path with feedforward but both perform close to or better than the current system. With the continuing servo upgrades and tuning, TCS hopes to improve even more from the current system and get even closer to the original requirements.

Table 1: Keck II Offset Moves Settling Times

Telescope moves (arcsec)	1	5	10	50	100	1000	10000
Time Requirement (sec)	0.3	1	1	3	3	10	20
AZIMUTH							
DCS System (2011)	0.8	2.7	2.8	-	5	9.9	19.2
TCS no feedforward	1.8	2.4	2.5	3.9	4.8	-	-
TCS one path w/FF	1.1	1.2	1.5	2.7	3.3	-	-
TCS two path w/FF	0.7	0.9	1.2	2.4	5.8	-	-
ELEVATION							
DCS System (2011)	3.2	2.6	2.6	-	5.7	10.3	23.4
TCS no feedforward	2.7	4.2	5.1	6	-	-	-
TCS one path w/FF	0.9	2.6	3.1	3.7	5.2	-	-
TCS two path w/FF	2.1	2.9	2.6	3.8	5.1	-	-

MOVING TO KECK I

As TCS nears completion of fine tuning the Keck II system, the necessary hardware upgrades to Keck I is being actively installed. To date the majority of the switching solution is installed. The dome, secondary, and rotator upgrades, which are fully installed, have been tied into the switching solution. The drive control and encoder system for the telescope axis are currently being tied in with the switching solution. All Keck I hardware installation will be complete by November this year with integration testing starting immediately after. The Keck I full system integration and validation completion is expected by the end of March 2016 followed by the Keck I TCS Upgrade Project ORR and handover in May 2016.

ACKNOWLEDGEMENT

The W. M. Keck Observatory is operated as a scientific partnership among the California Institute of Technology, the University of California and the National Aeronautics and Space Administration. The Observatory was made possible by the generous financial support of the W. M. Keck Foundation.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REFERENCES

[1] J. Johnson et al., "Keck Telescope Control System Upgrade Project Status", ICALEPCS 2013 MOCOAAB05

[2] K. Tsubota et al., "Switching Solution – Upgrading a Running System", ICALEPCS 2013 THCOBB05

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

EPICS IOC BASED ON COMPUTER-ON-MODULE FOR THE LNL LABORATORY

J. Vásquez, D. Pedretti, R. Ponchia, INFN/LNL, Legnaro, Italy
M. Bellato, R. Isocrate, INFN, Padova, Italy
M. Bertocco, UNIPD, Padova, Italy

Abstract

At LNL it is being carried out an upgrade campaign of the control systems of the accelerator complex. The two main goals are standardization of hardware and software and system interoperability. EPICS has been chosen as the standard framework for developing new control systems; this will address software standardization and system interoperability. In order to achieve hardware standardization, a new EPICS IOC is under development, which will become a basic construction block for all future control systems. The COM (Computer-on-Modules) from factor has been chosen as the hardware platform for the IOC, along with the peripheral devices needed for developing all the foreseen control system at LNL. Prototypes of this IOC has been developed using ADLINK's Type 6 COM Express modules on generic carrier boards with DIO, ADC and DAC expansion boards. These prototypes have been tested under typical applications at LNL in order to validate the hardware platform choice. Experimental results show that the performance of the IOC in terms of effective resolution (ENOB and bias error), sample rates and CPU usage is suitable for satisfying the requirements of the control systems.

IOC DESCRIPTION

The IOC is intended as a standard system with the necessary interfaces in order to be able to control the instrumentation present on all the foreseen system at LNL. EPICS tools will be used for software development in a standard way. For each application, a custom-made control algorithm will be developed under the framework.

COM form factor has been chosen as the hardware platform for the IOC. In particular, the type 6 COM Express standard was selected [1].

The COM will be installed on a custom carrier board, designed at LNL. This carrier board will contain peripheral devices such as digital IO, ADCs, DACs, stepper motor controllers, and communication interfaces, among others, connected to the COM. Its design is under carefully considerations in order to cover all the common needs for the current and the foreseen future control systems at LNL.

Some interfaces are already available directly from the COM. Others will be controlled using USB ports. Many others will be controlled by a FPGA that will communicate with the COM through a PIC Express lane. Figure 1 shows a simplify block diagram of the IOC.

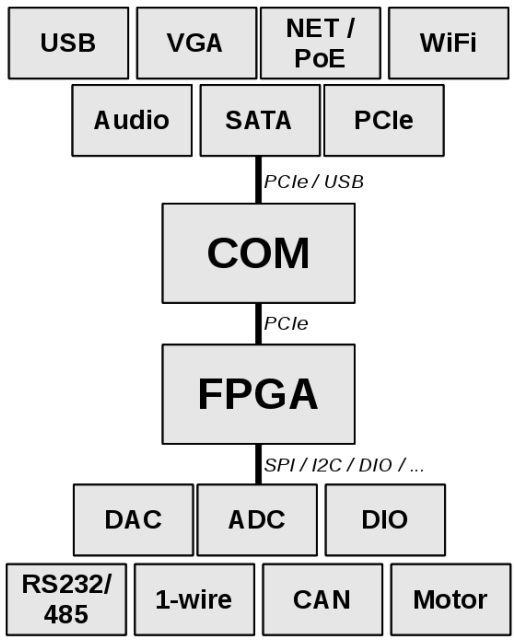


Figure 1: IOC simplify block.

IOC PROTOTYPES IMPLEMENTATION AT LNL

In order to test the validity of the COM as hardware platform for IOC developments, prototypes were developed using commercial devices. These prototypes, at the same time, allowed developing the software part of control systems.

The developed software is mostly independent from the hardware implementation, due the fact that the selected COM uses Intel CPUs allowing the use of a standard Linux environment for the implementation. In this way, once the custom IOC will be ready, the software would be easy transport to the new hardware platform.

On the other hand, prototypes were used to develop and implement diverse control systems at LNL. This allowed testing them under real operative conditions, as well as to develop critical control system without waiting for the final custom IOC to be available.

Hardware Architecture

The prototypes have been developed using Adlink [2] Type 6 COM Express module on a generic carrier board. Commercial Digital IO, ADCs and DACs PCIe boards were installed on the carrier board in order to provide IO capabilities to the IOC.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Adlink's DAQe-2214 boards were used as ADC inputs. Each board uses one x1 PCIe interface and provides 16 16-bits, 250 KSample/s, +/-10 V channels. It also provides 24 digital IO channels.

In the same way, Adlink's PCIe-6216 boards were used as DAC outputs. Each board provides 16 16-bits, +/-10 V channels, using a 1x PCIe line.

On Figure 2 it is shown a picture of a prototype with the COM on a generic carrier board with a DAQe-2214 and a PICe-6216 boars.

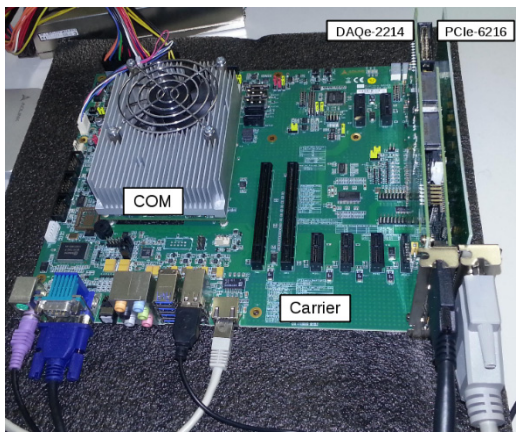


Figure 2: IOC prototype.

Software Architecture

Adlink provides linux drivers for the expansion boards used on the prototypes. The drivers provide common APIs. In order to interface the EPICS IOC device supports to Adlink drivers, asynDriver [3] was used.

The analog input acquisition is performed using a periodic scanning function on the asynDriver software. This function reads the ADC inputs and do callbacks updating the recently read values into the parameter library. In this way, the EPICS record attached to an analog input receives an interruption request each time a new value is available. Inside the scanning function loop, an adjustable delay was introduced in order to allow changing the sample rate of the inputs. The same procedure is done for digital inputs acquisition.

On the other hand, for the analog outputs, a function that writes the DAC channels was implemented on the asynDriver software. This function is called each time a new value is written to an EPICS record attached to an analog output, passing it the written value. The same procedure applies for digital outputs.

EXPERIMENTAL TEST AND RESULTS

A soft IOC was deployed on a prototype IOC in order to test its performance. The COM used on the prototype was the model “Express-IB-i3-3120ME” which has an Intel Core i3-3120ME CPU at 2.4GHz and 4Gb of RAM. The COM was installed on a generic carrier board model “Express-BASE6” with a DAQ-2214 and a PCIe-6216 boards. A picture of this prototype is shown on Figure 2.

Fedora Core 17 with kernel versions 3.3.4-5.fc17.x86_64 was installed on the IOC with the board drivers installed on the system. EPICS R3.14.12 and asynDriver R4-23 were also compiled and installed. On this system, an EPICS IOC application consisting only on analog inputs and analog outputs records linked to the physical ADCs and DACs using the driver written with asynDriver.

This IOC application was used to compute the CPU usage by the application and the resolution of the ADC channels.

Acquisition Rate and CPU Usage Performance

This test was performed in order to estimate the maximum sample rate that be achieved with this IOC, as well as the correlation between the sample rate and CPU usage by the application. The test consisted on measure the CPU usage by the EPICS application (which include the driver interface) for different samples periods.

The sample period was changed using the delay inserted on the scanning function on the asynDriver described earlier. Four different scanning rate where targeted. The first one correspond to the faster scanning rate achievable, setting the delay to zero. The latest one was set to around 10 Samples/s, which it is a usual value used as sample frequency on several applications at LNL. Finally, other two rates were selected in between these values, i.e. 200 Samples/s and 1000 Samples/s. The test was perfomed both, only acquiring a single analog input channel, and acquiring all 16 analog input channels. The results are presented on Figure 3.

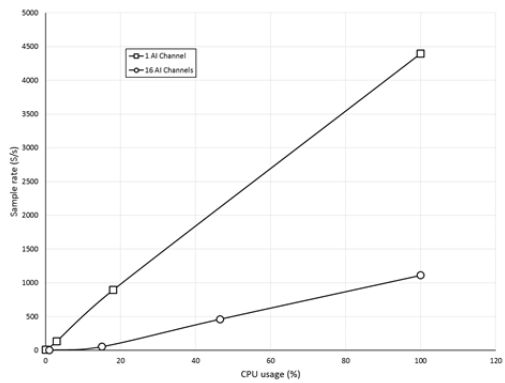


Figure 3: CPU usage versus sample frequency results.

The results show that the highest sample rate that can be reached scanning only one channel is almost 4.4 KSample/s, while when scanning all 16 channel it is around 1.1 KSample/s. However, for reaching these rates, the application uses all the available CPU, due to the fact that the acquisition loop continuously read the input channels, without any delay.

The CPU usage decreases drastically when a delay is set, lowering also of course the acquisition rates. Furthermore, it is evident that more CPU is needed for achieving higher rates on all channel respect to the case with only one channel, as it is expected.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Particularly, for application when only one channel is needed, the second obtained value is a good compromise, achieving a sample rate of almost 900 Sample/s using only 18% of CPU. For application when all channels are needed, the third case is a good choice, using only 15% of CPU reaching rates of almost 55 Samples/s. Finally, low acquisition rates of some units of Sample/s, typically used on the LNL applications, the CPU usage is negligible.

Analog-to-Digital Performance

In order to determinate the performance of the ADC inputs, tests were performed for estimating its effective number of bits (ENOB) of resolution, following the method using frequency domain described on [4].

The ADC input range was set to +/-10 V, with a 90% full-scale input sine wave signal. For each test, more than 205888 samples were taken, at a rate of 890 Sample/s. The frequency of the input signal was 400 Hz, chosen in order to be lower that the Nyquist frequency. The calculations were performed for other six lower frequencies: 200 Hz, 100 Hz, 50 Hz, 10 Hz, 5 Hz and 1 Hz.

A plot of the obtained ENOB for the different input signal frequencies is presented on Figure 4.

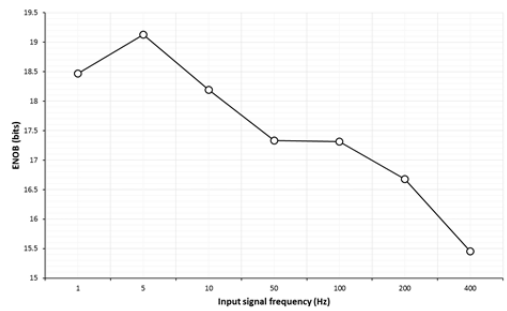


Figure 4: ENOB obtained for the analog inputs.

It can be see that for low frequency signals (lower than 150 Hz approximately) the obtained ENOB is higher than 17 bits, reaching even 19 bits. For a signal of 400 Hz, the ENOB is around 15.5 bits. The reason for accomplishing resolution greater the 16 bits (the ADC resolution) for the lower frequency signals is the effect of oversampling the input signal.

CONTROL SYSTEM IMPLEMENTATION AT LNL

Four different system were selected for the prototype implementation. The control system for these four systems contains most of elements that will be necessary on all the future implementations. Therefore, testing the prototype on all of them will demonstrate that the IOC could be implemented on any other system at LNL.

The Beam Diagnostic Data Acquisition

Beam diagnostic units are one of the most important and widely use elements on any accelerator facility. At LNL, a standard beam diagnostic unit is formed by a Faraday Cup for measuring the beam current, and a Beam Profiler for measuring the spatial distribution particles of the beam.

An IOC prototype was designed for acquiring the data from this beam diagnostic unit. On the IOC an analog input channel is used to read the output of the pre-amplifier of the Faraday Cup, while two digital outputs are used to set its gain. On the EPICS database, the read signal is converted back to the corresponding beam current.

The pre-amplifier of the beam profiler is formed by forty transconductance amplifiers with variable gain, similar to the one use on the Faraday Cup pre-amplifier. The 40 signals are multiplexed to a single analog output. On the IOC, an analog input channel is used to read the output of the pre-amplifier, while one digital output is used to generate a clock signal to control the analog multiplexer and two digital outputs are used to set the gain. In this case, a scanning function was implemented on the asynDriver driver that generates the clock and synchronously reads the analog input. At the end of a scanning sequence, the data are placed on a 40-value vector, which is passed to an EPICS waveform record.

The Electrostatic Beam Focalization and Beam Extraction

Beam focalization devices are of paramount importance on an accelerator facility, used even more extensively that the beam diagnostics. A large number of electrostatic beam focalization units are installed along the low energy beam line. These devices are driven by a series of high voltage power supplies.

On the other hand, the beam extraction system, even though it has a completely different function, its control system implementation is very similar to the beam focalization system. It consists on a single high voltage power supply that set an electric potential between the ion source and the extraction electrode.

An IOC prototype was designed for controlling the beam extraction system and an electrostatic beam focalization unit. The focalization unit comprises a steerer and a quadrupole triplet groups.

For each power supply, both the output voltage and current can be set using four analog signals. On the IOC, two ADC channels, two DAC channels and a digital output are used to control each one of the power supplies. Inside the EPICS databases, Proportional-Integral-Derivative (PID) algorithms were implemented in order to automatically control both the current and voltage, according to the set points imposed by the operator.

The Magnetic Beam Steerers

Magnetic steerers are part of the beam transport elements. They are widely used at LNL for the transport of high energy beams. A magnetic steerer unit is form by four coils, two vertical and two horizontals, installed on the beam line. They are connected on series of two, and piloted using two high current power supplies.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The IOC configuration is exactly the same respect to the electrostatic beam focalization described on the previous section.

The ECR (Electron Cyclotron Resonance) Negative Beam Source

At LNL, one of the stable ion source present is an Electron Cyclotron Resonance (ECR) source [5] [6]. The ECR ion source in controlled by a large number of instruments: nine power supplies with analog control, five power supplies and measure instrument with serial communication port (RS232), one faraday cup for beam diagnostic, and one PLC.

The IOC developed is equipped with 16 DAC channels, 32 ADC channels, 96 digital IO and 8 serial ports, in order to be able to control all the instrumentation on the system. Most of the software components were taken from the previously described prototypes, as for example, PIDs algorithms for the analog controlled power supplies, and conversion algorithms for the beam diagnostics. On the other hand, for the instrumentation with serial communication, new algorithms were produce.

CONCLUSION

Standardization of the control systems of the accelerator complex at LNL is being addressed with the development of the new EPICS IOC.

It will be based on highly integrated Computer-on-Modules, installed on a custom tailored carrier board, equipped with all the necessary IO interfaces for satisfying the requirement of all the foreseen control systems. EPICS as a framework will bring a homogeneous software architecture with added benefits as total interoperability between the systems, and essential services as data archiving and logging.

Tests showed that the ADC inputs could reach high sample rates, around 4.4 KSample/s at the cost of high CPU usage. Nevertheless, good tradeoffs are available; a good choice for the LNL applications is the rate of 900 Sample/s using only 18% of CPU. Furthermore, sample rates of tens of Samples/s could be easily reached, with negligible use of CPU. Also, ENOB between 15.4 and 19bits were obtained on the ADC inputs.

In order to validate this hardware platform as suitable for developing the control systems at LNL, prototype IOCs were developed and implemented using commercial devices. These IOCs have been tested under real applications at LNL, with the implementation of four control systems: the beam diagnostic data acquisition, the Front-End beam extraction and focalization, the magnetic beam steerer and the ECR source systems. All the installed prototypes have shown that the hardware platform is suitable for the development of the LNL control systems using EPICS.

Although the performance is not optimal, the prototype IOCs have shown that they are suitable for controlling the selected systems, satisfying the operation requirements. However, much higher efficiency is expected from the custom IOC design.

REFERENCES

[1] ADLINK COM Express website: http://www.adlinktech.com/Computer-on-Module/Com-Express/index.php
[2] Adlink Computer-on-Module website: http://www.adlinktech.com/Computer-on-Module
[3] M. Rivers, "asynDriver: Asynchronous Driver Support," Advanced Photon Source, Argonne National Laboratory, website: http://www.aps.anl.gov/epics/modules/soft/asyn/
[4] "IEEE Standard for Terminology and Test Methods for Analog-to-Digital," IEEE Standard, p. 1241 (2010).
[5] B. Wolf, Handbook of Ion Sources, CRC Press (1995).
[6] M. Cavenago, T. Kulevoy and S. Petrenko, "Operation of the LNL ECR Ion Source," in EPAC, Vienna, Austria (2000).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MEBT AND D-Plate CONTROL SYSTEM STATUS OF THE LINEAR IFMIF PROTOTYPE ACCELERATOR*

J. Calvo†, D. Jiménez-Rey, E. Molina Marinas, J. Mollá, I. Podadera. Ciemat, Madrid. Spain

Abstract

Linear IFMIF[1] [1] Prototype Accelerator (LIPAc) [2], Rokkasho, Japan, comprises a succession of devices and systems that accelerate a deuteron beam up to 9 MeV with a current of 125 mA, generating a power of 1.125 MW, and transport it up to a beam dump. The beam power becomes critical from the point of view of losses; even tiny losses must be avoided. This fact, and the complexity of the accelerator operation, requires a coherent strategy when designing, commissioning and optimizing the accelerator control system, specifically focused in the control systems of the Medium Energy Beam Transport (MEBT) and the Diagnostic Plate (DP, a movable set of diagnostics). Both systems are essential to validate the performance of the accelerator and particularly the ion source, Radio Frequency (RF) and Radio Frequency Quadrupole (RFQ) systems. This contribution will describe the recent advances in the control architectures and the EPICS based developments achieved in MEBT for the motion control of bunchers and scrapers, control of the power supplies in quadrupoles and steerers, and refrigeration and vacuum. Besides, control of fluorescence profile monitors (FPMs) in the D-Plate is displayed.

LIPAC CONTROL SYSTEM

LIPAc control system consists of the remote control, monitoring and data acquisition of all devices, systems, subsystems and operations carried out in the accelerator vault with intense radioactive environments. It uses EPICS [3] as the main set of control software tools, providing the control with distribute and real-time features. It is composed of five main elements, namely, accelerator Central Control System (CSS), Local Area Network (LAN), Personal Protection System (PPS) [4], Machine Protection System (MPS) and Timing System (TS). Within each LIPAc subsystem there is a Local Control System (LCS).

LIPAc control system development is an unique and complex work due to the special characteristics of the prototype accelerator and its beam dynamics, that are, the unprecedented high beam intensity inducing the simultaneous combination of high beam power and high space charge. Therefore, MEBT and D-plate control system development is related with general accelerator characteristics in the following way:

- LIPAc is aiming to produce a very powerful CW deuteron beam (1.1 MW @ 9 MeV). The unavoidable beam losses [5] along the accelerator lead to an important rate of neutrons and radiations that must be properly controlled. On the other hand, the power of the beam itself could, in case of accidental or abnormal operation conditions, damage or destroy highly valuable parts of the machine.
- Beam losses must be observed and studied in order to fulfil beam stability in terms of transverse position and shape. To obtain this fundamental information several diagnostics devices have been installed along the accelerator line, these devices require an specific control and a fast data acquisition.

LIPAc Local Control Systems (LCSs)

The main function of the LCSs is, obviously, to control its own subsystem, displaying their own process variables (PVs) for the rest of LCSs and enabling access and modification of these PVs by the CSS. Safety, device protection and optimization activities are also included among its functions. LCSs are composed of different elements but following three layers architecture based on EPICS, Fig. 1. The lower level consists of equipment (power supplies, diagnostics, sensors, motors) connected to the intermediate level via specialized links. The middle layer consists of EPICS front-end systems (so-called input-output controllers (IOCs)). The upper layer, part of the CCS, consists of workstations, which either play dedicated roles, offering a man-machine interface function. These two levels communicate through TCP/IP on Ethernet.

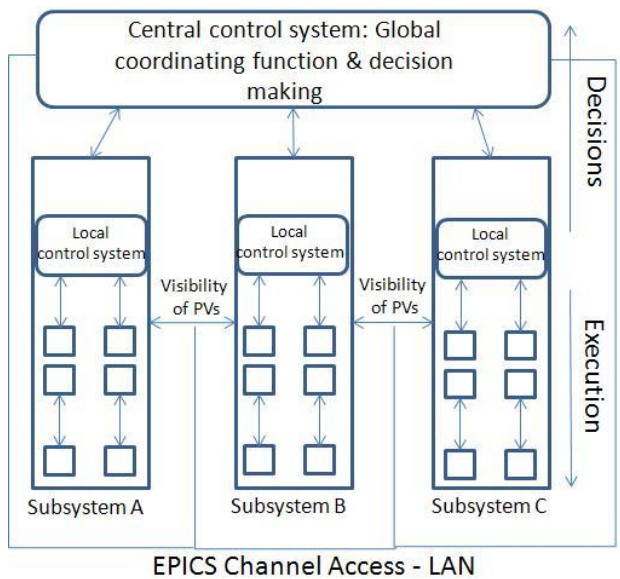


Figure 1: LIPAc general distributed process control.

* This work has been partially funded by the Spanish Ministry of Economy and Competitiveness, under projects OPTIMHAC FIS2013-40860-R and IFMIF-EVEDA II. Ref: AIC-A-2011-0654.

† julio.calvo@ciemat.es

[1] IFMIF, the International Fusion Materials Irradiation Facility, is an accelerator-based neutron source that will use Li (d, xn) reactions to generate a flux of neutrons with a broad peak at 14 MeV equivalent to the conditions of the Deuterium-Tritium reactions in a fusion power plant. IFMIF is conceived for fusion materials testing.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

MEBT LOCAL CONTROL SYSTEM

The MEBT subsystem [6] is responsible of the transport and matching of the RFQ beam into the SRF Linac.

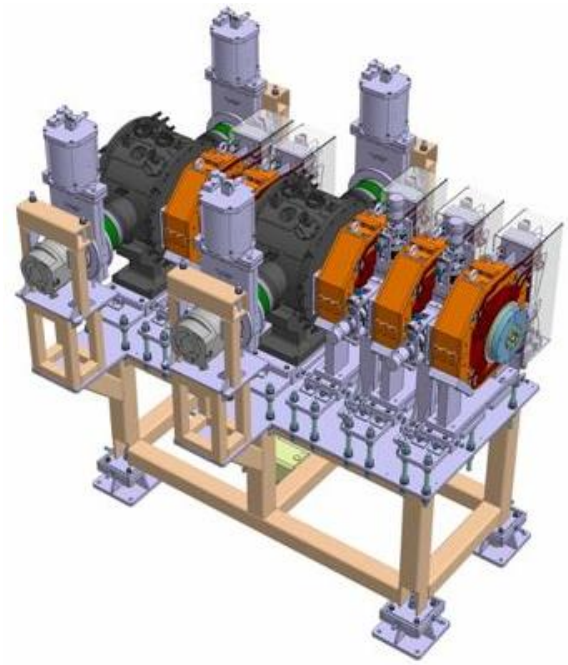


Figure 2: MEBT subsystem 3D scheme.

In order to minimize the beam losses caused by the strong space charge forces affecting the beam in this area, while keeping the sufficient freedom in beam optics optimization, a very compact scheme based in two sets of quadrupole magnets with steerers and two re-buncher cavities has been proposed, Fig. 2.

MEBT Local Control System interfaces the MEBT and the rest of the accelerator control system, the complete MEBT LCS architecture is shown in Fig. 3. Control of the main devices that make the MEBT up is explained in the next sections.

Control of the Bunchers

There are two re-buncher resonant cavities, see Fig. 4, with power couplers to supply the RF to the resonator, and a tuning system. That means temperature measurements for the water cooling and motion controls for the tuners (two stepper motors) including position readbacks (linear potentiometers) and limit switches. The stepper motor shifts a copper plunger in order to keeps resonance frequency constant during operation, potentiometers and limit switches are necessary to manage the motor. Cooling channels at the bunchers are used to keep copper temperature at 20 °C. The whole buncher control is managed by a Siemens [7] PLC S7-300 except the stepper motor control which is externally carried out by the low level RF system [8]. Communication between PLC and EPICS is achieved using S7plc EPICS driver.

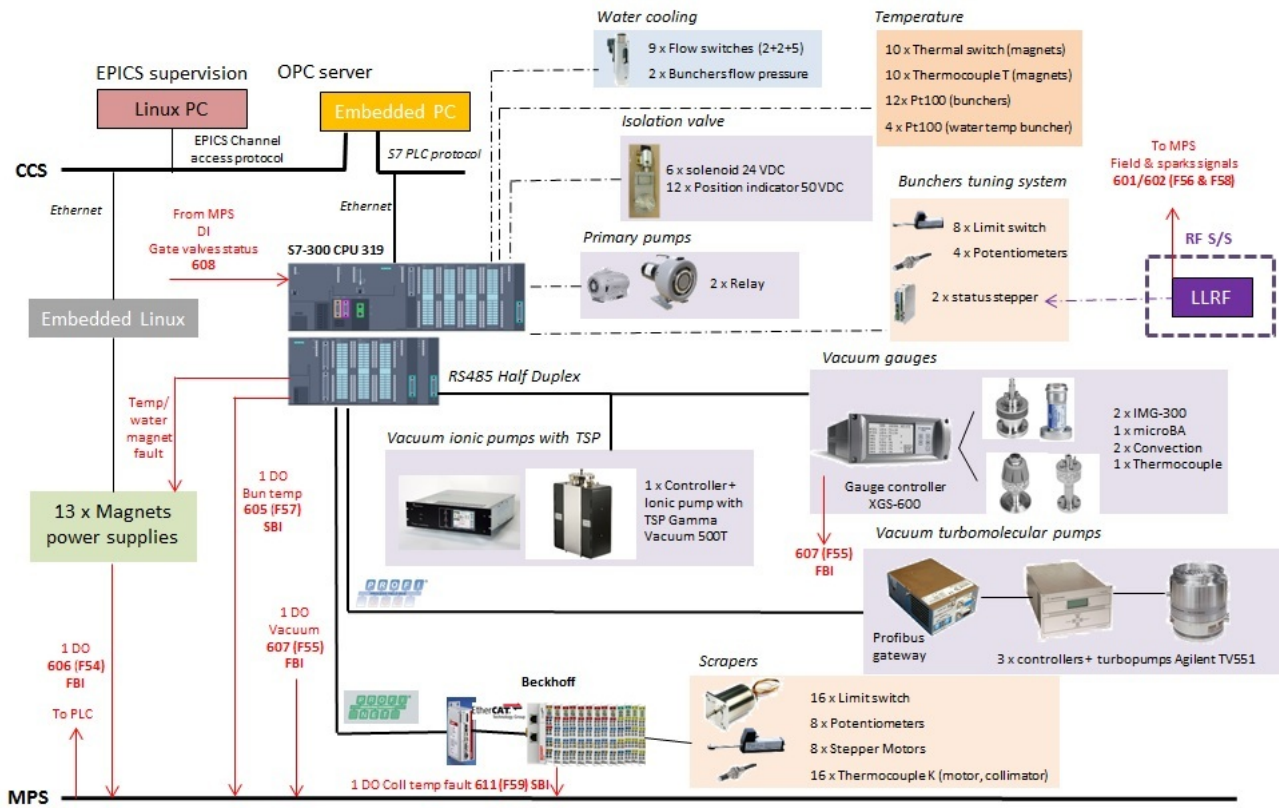


Figure 3: Global MEBT control architecture.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

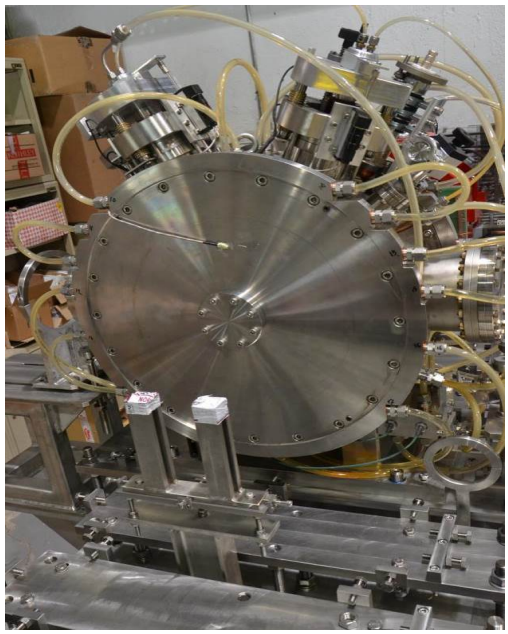


Figure 4: MEBT buncher cavity.

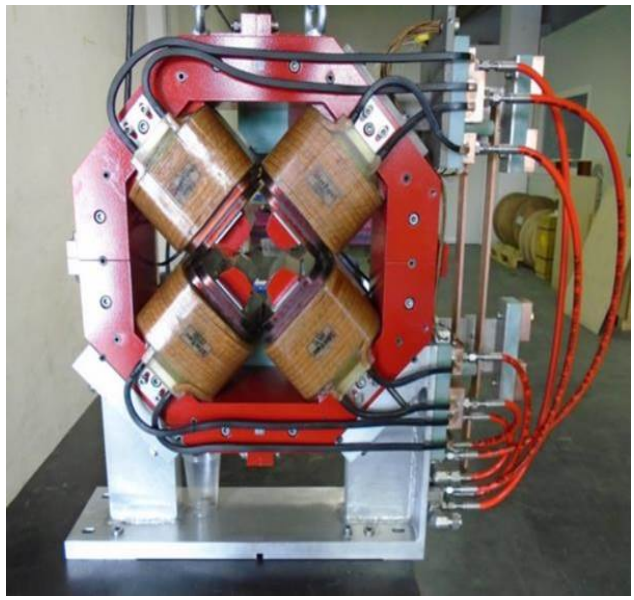


Figure 5: MEBT quadrupole magnet.

Control of the Scrapers

The two scrapers must stop the out-of-emittance beam particles before injection into the SRF Linac. Thus, they are movable items so they need motion control (eight stepper motors) including position readbacks (linear potentiometers) and limit switches. The scrapers motion control is managed by a Beckhoff [9] system. This solution system is based on CPU CX1020 model with digital input/output modules for limit switches. The communication from the Beckhoff system to EPICS is carried out using S7plc EPICS driver.

Control of the Quadrupoles

Transverse and longitudinal focusing for the beam is carried out using quadrupoles and steerers, as shown in Fig. 2. Therefore, the LCS controls settings and monitorage of the five quadrupoles power supplies and the eight steerers bipolar power supplies and temperature measurements for the water cooling. The beam size in the RFQ is small (phase advance 90 deg/m) while in the HWR-Linac the phase advance is comparatively lower (20 deg/m) where the distance is much longer between focusing sections. Consequently, the MEBT must have a very compact structure to limit the beam emittance growth. Thus, four quadrupoles, see Fig. 5, are used for transverse focusing and one more is needed to limit the beam size in the MEBT itself. The control of the five quadrupole power supplies and the eight steerers bipolar power supplies is being carried out with EPICS over Modbus protocol.

Refrigeration and Vacuum

Four pumping units and four valves are used to maintain the vacuum level, ten gauges are utilized to read it. A PLC is in charge of the control of these devices; its communication with EPICS is carried out using S7plc EPICS driver.

D-PLATE LOCAL CONTROL SYSTEM

Beam Instrumentation (BI) subsystem must warranty the successful operation of the accelerator from commissioning phases of RFQ, MEBT, Superconducting Radio Frequency (SRF) Linac and High Energy Beam Transport (HEBT), to the full power operation. Hence, its main objective is to provide all necessary information to properly transport and accelerate the beam from the source to the beam dump, then, to fully understand and measure all beam characteristics and operation optimization. Most of the diagnostics are concentrated inside of the Diagnostics Plate (DP or D-plate). DP is a movable set of diagnostics, see Fig. 6, that is placed downstream from the SRF Linac, in the HEBT. Main parameters of the beam will be measured in the DP, e.g. current, phase, beam position, transverse profiles, mean energy, emittance measurements, micro losses, energy spread, etc. Beam measurements play a critical role in LIPAc due to its uncommonly high beam current and high beam power. A global control strategy is thus necessary to clearly decide between the different measurements categories. The scope of this paper includes only the recent advance in the control of the fluorescence profile monitors (FPMs).

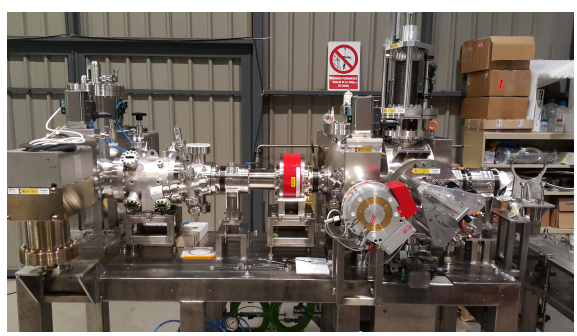


Figure 6: D-Plate support integration at Ciemat.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Control of the Fluorescence Profile Monitors (FPMs)

The objective of these devices is to develop a non-interceptive beam transverse profiler (in X and Y axis) based on the residual gas fluorescence originated by the beam-gas interaction. FPMs (two in HEBT, two in D-plate) could be used as well for the emittance measurements with quadrupole scans.

The control of the FPMs is based on a Vertilon [10] data acquisition system (PhotoniQ Model IQSP482). High voltage control for the PMT is required; therefore the acquisition system delivers 64 data points for both FPMs of the D-plate.

Control software is based on client-server architecture, see Fig. 7. Vertilon device is accessed from an EPICS IOC (client) using TCP/IP protocol, connecting with a LabVIEW application (server), which is the one that gets direct access to the device.

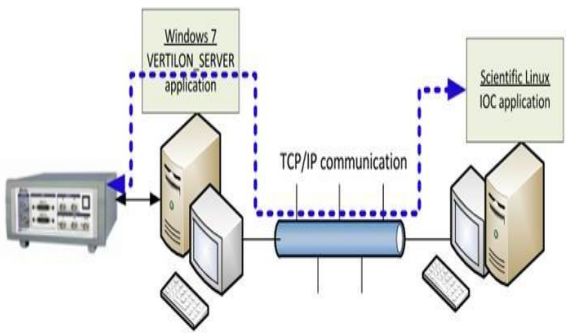


Figure 7: FPMs client-server support architecture.

CONCLUSIONS AND OUTLOOK

The objectives and requirements of LIPAc have been partially explained being the rationale for the design and development for each of the local control systems. The ability to make EPICS run in each of the devices and subsystems presented is not trivial, but the benefits provided against other centralized solutions are absolutely overwhelming. A large and complex facility like LIPAc, whose operation opens unstudied ways, cannot be conceived with independent subsystems, centralized at a single point or unable to understand each other.

LIPAc beam power provides unique characteristics that make the control system an unique and heterogeneous elaboration. For MEBT and D-Plate subsystems, theirs local control systems design has been revealed, explaining the main features and showing its architecture. Local control systems fit in a mostly distributed architecture and properly respond to the specific function for which they are designed, however, they are not isolated entities, they communicate with each other and follow the same multilayer structure.

Since there is no previous experience in operating a machine like LIPAc, all EPICS PVs are available for the central control system, which is the one and only that can manage the whole accelerator and can take decisions based on the information given from every subsystem. Making all the PVs visible for the central control system becomes the next step in order to finish the integration of the whole LIPAc control system.

ACKNOWLEDGMENT

We would like to thank the help from LIPAc EU-HT in the design of the LCSs architectures. We also want to acknowledge the suggestions from Álvaro Marqueta. A warmful thank to the Universidad Politécnica de Madrid for the valuable help in the development of the FPMs control system.

REFERENCES

[1] J. Knaster et al., "IFMIF, a fusion relevant neutron source for material irradiation current status" Journal of Nuclear Materials, vol. 453, number 1-3, pages 115-119. 2014. http://www.sciencedirect.com/science/article/pii/S0022311514004085

[2] D. Gex et al., "Engineering progress of the linear {IFMIF} prototype accelerator (LIPAc)" Fusion Engineering and Design, vol. 88, number 9-10, pages 2497-2501. 2013. http://www.sciencedirect.com/science/article/pii/S092037961300464Xf

[3] L. Dalesio et al., "EPICS architecture" ICALEPCS91:Proceedings Conference on International Accelerator and Large Experimental Physics Control Systems, pp. 278-282, 1991.

[4] Hiroki Takahashi et al., "Safety managements of the linear IFMIF/EVEDA prototype accelerator " Fusion Engineering and Design, vol. 89, number 9-10, pages 2066-2070. 2014. http://www.sciencedirect.com/science/article/pii/S0920379614001483

[5] C. Oliver et al., "Studies of Emittance Measurement by Quadrupole Variation for the IFMIF-EVEDA High Space Charge Beam" Proceedings of IPAC2011, San Sebastian, Spain, pp. 652-654, 2011.

[6] I. Podadera et al., "The Medium Energy Beam Transport Line (MEBT) of IFMIF/EVEDA LIPAC" Proceedings of IPAC2011, San Sebastian, Spain, http://accelconf.web.cern.ch/accelconf/ipac2011/papers/weps058.pdf

[7] http://http://www.siemens.com/entry/cc/en/

[8] J. Calvo et al., "EPICS based low-level radio frequency control system in LIPAc" Fusion Engineering and Design, vol. 87, number 11, pages 1872-1879. 2012. http://www.sciencedirect.com/science/article/pii/S0920379612004218

[9] http://www.beckhoff.es/

[10] http://www.vertilon.com/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REVOLUTION PROJECT: PROGRESS IN THE EVOLUTION OF SOLEIL MOTION CONTROL MODEL*

S. Zhang†, F. Blache, D. Corruble, C. Kheffafa, YM. Abiven, SOLEIL, Gif-sur-Yvette, France
S. Minolli, NEXEYA SYSTEMS, La Couronne, France

Abstract

SOLEIL is a third generation synchrotron radiation source located near Paris in France. REVOLUTION (REconsider Various contrOLler for yoUr motion) is the motion controller upgrade project currently in progress at SOLEIL. It was initiated to maintain the facility operations by addressing the risk of hardware obsolescence in motion control but at the same time making room for complex applications requirements to face new high performance challenges. In order to achieve these considerations, SOLEIL's strategy move was to go from a single controller for all applications to two motion controllers. A first Controller GALIL DMC-4183 was chosen to succeed the previous version DMC-2182. Both controllers can be integrated in the existing architecture with little hardware and software adaptation enabling full compatibility with the existing architecture. A second controller, Delta Tau Power Brick, has been selected as a HIGH PERFORMANCE solution providing advanced functionality.

The CLASSIC controller upgrade is about to be completed and the integration of Power Brick into the SOLEIL control system is ongoing. The system complexity is abstracted by embedding processing functions into low-level code and giving end-users a simple high-level interface. The work done to structure the interfacing and standardization of the controller are detailed in this paper.

CONTEXT OF REVOLUTION AT SOLEIL [1]

Motion Control Status and Upgrading Motivations

SOLEIL needs to upgrade its standardized GALIL motion controller in order to address two challenges: First, the current motion controller used in SOLEIL is dated and in risk of being discontinued and we must have an updated product before this happens. Nevertheless, this is not expected to occur in the next few years. Secondly, new motion control applications demand a higher performance and more advanced features that can not be reached by our current controller.

Strategy: Change Model

To achieve the next three goals of "Increasing performance, maintaining operational continuity, and controlling overall cost", SOLEIL decided to replace the current model of a "single and universal controller" by a model with two kinds of motion controllers:

- CLASSIC motion controller, mainly used and suitable for "simple and classic" applications. This controller is fully compatible with the existing hardware and software motion control architecture.
- HIGH PERFORMANCE controller, used in few cases, suitable for "complex and fast" applications. The hardware and software motion control architecture is designed to be close to the existing architecture for the CLASSIC controller.

CLASSIC CONTROLLER EVOLUTION

The GALIL DMC-4183 controller was chosen to succeed the current Galil DMC-2182. Its implementation needs some hardware and software developments which are currently being done to make it compatible with current architecture. On the software aspect: a new firmware was developed by GALIL which includes the new "Continuous Closed-Loop on Stepper motors" feature. It was validated by SOLEIL by an operational application done on the monochromaters of LUCIA and TEMPO beamlines. To use this feature, a new microcode (embedded code) was developed and validation tests are ongoing. For the hardware: in order to use the current ControlBox rack (same size, same pinouts), an interface board (MIG-4121) has been developed which adapt the DMC-4183 to the internal pinout of the ControlBox rack. The integration in the rack has been validated and is now operational.

HIGH PERFORMANCE CONTROLLER STANDARDIZATION

Controller Product

Delta Tau Power PMAC in the Power Brick LV-IMS format was selected for the new "HIGH PERFORMANCE" solution after a long evaluation process.

Figure 1: Power Brick LV-IMS.

Power PMAC is a general-purpose embedded computer with a built-in motion and machine-control application which offers: powerful computing capacity, multi-axis synchronization, encoder processing, virtual control through

* Work also supported by XT. Tran, M. Cerato, G. Renaud, E. Fonda and SAMBA Beamline staff, C. Engblom, Delta Tau Ldt., IMO JEAMBRUN AUTOMATION, Observatory-Sciences Ldt...
† szhang@synchrotron-soleil.fr

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

kinematic equations, non-linear trajectories, built-in software PLCs (programmable in Power PMAC script and/or C language) etc. Moreover, around Delta Tau products there is an international user community in large research facilities.

The Power Brick LV-IMS rack (see Figure 1) is now available with a SOLEIL standard compatible connectivity for motors and encoders. It comes with the configuration of a Power PMAC CPU and 8-axis integrated amplifier circuits for 2-phase and 3-phase motors direct-drive. The number of axes can also be expanded through a MACRO fiber optics network. As an option, it can be configured to support different encoder feedback protocols among a wide variety of choices.

Control Architecture

For reasons of consistency and usability we intended to keep a control architecture as close as possible to the existing one. Considering the performance benefits of the product, most process functions are embedded into low-level to abstract the system complexity for the high-level software. In order to make the controller easy to use and maintain in normal operations, we developed some standard functionalities for the hardware low-level configuration and in the embedded software.

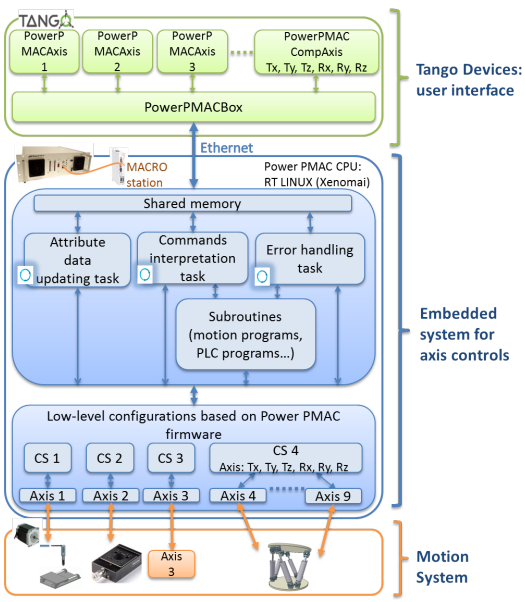


Figure 2: Power PMAC embedded system structure with an example of motion system application.

High-level Software for User Interface

In order to structure the end-user interface, the software high-level architecture shown in Figure 3 has been developed. The software consists of Tango devices for axis controls and general rack controls. The rack controls communicate over the "PowerPMAClib" and "DeltaTauLib" libraries.

The Tango device interface for the new HIGH PERFORMANCE controller is designed to be very similar to the standard CLASSIC controller. A single Device Server

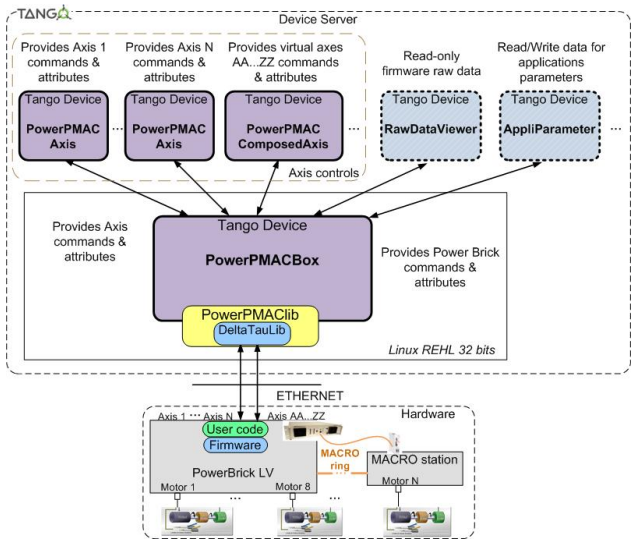


Figure 3: High-level software architecture.

"ds_PowerPMAC" including the main set functionality is composed with the following devices:

- Device **PowerPMACBox** for the controller and general data.
- Device **PowerPMACAxis** for driving physical axis.
- Device **PowerPMACComposedAxis** for driving composed virtual axes (one device/CS).

To complete the functionality supply, some devices are planned to be realized in the near future:

- Device RawDataViewer, a diagnostic tool providing read-only raw firmware data of Power PMAC for specified axis.
- Device AppliParameter, a tool providing read and write data of the controller for users to parametrize their application-dedicated variables.

The Tango devices access the Power PMAC level through "PowerPMAClib", which encapsulates "DeltaTauLib". "DeltaTauLib" is supplied by Delta Tau to provide access to the firmware and embedded "user" data via Ethernet [2]. The "PowerPMAClib" library, developed by SOLEIL, presents in its interface the access to general controller functions (e. g. connection, status, temperature, operating time, reset etc.) and the axis control functions (e. g. position, speed, stop etc.). Within axis controls, the library makes the link to the data structures stored in the shared memory of the controller for data exchanging and command settings. The information in the data structures are accessed and updated from the embedded programs in parallel from the Tango devices.

Embedded System Structure

Power PMAC uses Linux OS with the Xenomai preemptive real-time kernel which runs a single real-time application to handle tasks of different priorities [3].

In order to better manage the controller in an operational condition with different motion system applications, we implemented a generic Power PMAC project independently of the application system which is organized in 3 layers:

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

an embedded software layer for Tango axis control interface, a subroutines layer for functional sub-programs, and a hardware layer for low-level configurations (see Figure 2).

Embedded software layer: interface for Tango axis control For physical- or virtual- axis control: all commands, property settings, attribute reading/writing, and state/status requests from the Tango software go through the embedded software layer to access the firmware. Via predefined indexed data structures in the shared memory of the controller, this layer acts as an intermediate between the Tango software and Power PMAC firmware. Data structures are shown in Figure 4.

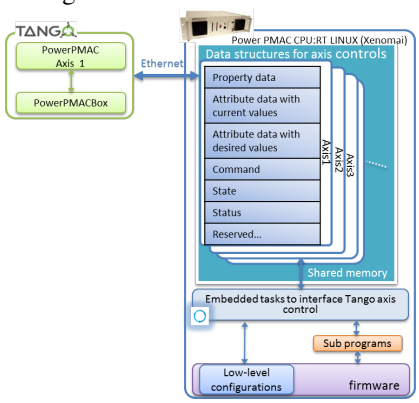


Figure 4: Predefined indexed data structures in Power PMAC shared memory for Tango axis control.

The features of this layer are:

– Interpretation of commands provided by the Tango device and that they only run if operating conditions allow.
– Attribute data updating: for exchanging high-level attributes with low-level parameters, parameters being changed only within allowed conditions.
– Error handling: setting flags in the state/status variables when certain errors/warnings occur.

These features are realized by 3 different tasks running in 2 threads with separately adjustable cycle times. One single thread is used for the tasks of interpretation and error handling. In the scope of the high-level polling frequency, this thread has higher priority than the thread used for updating data attributes.

Subroutines layer: functional sub-programs definition This layer implements some sub-programs that can be called or stopped by the software layer command request to perform a processing function or trajectory followed moves. These programs can directly use firmware functions and act on low-level hardware settings. Their execution status are continuously checked on by the interface layer for safety handling.

Some application-dedicated programs, including motion- and PLC- programs, will be defined and implemented depending on the needs. Some common programs will be part of the generic project:

– Motion programs for reference-position initialization.
– Duty cycle PLC program for motor stops within a certain period.
– Vacuum mode PLC program to itch motor current depending on the motor status (moving or stopped).
– System start-up PLC program to initialize the amplifier and reset the embedded system.

Hardware layer: low-level configurations Some typical hardware setups need to be standardized to minimize and simplify installation work. These setups have been identified of which the configuration files have been templated:

- Encoder conversion setups: Quadrature, Sin/Cos, SSI, BiSS-C...
- Motor setups:
  – 2-Phase stepper motor setup: closed Loop & micro-stepping
  – Motor setup using third-party drives with PFM signals output
  – 3-Phase brushless servomotor setup
  – DC brush motor setup

Based on the tested and validated templates, a GUI tool (see Figure 5) has been developed to automatically generate configuration files according to preselected parameters: channel number, motor type, encoder type, motor current etc. Hardware commissioning will however require a tuning process followed by manual modifications of the files. The PID regulation can be done using the Power PMAC IDE tuning tool. From then, the CS (Coordinated System)

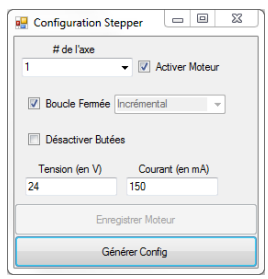


Figure 5: Hardware configuration file generator tool.

configurations will be added in the files to create the relationship between controlled axes and motors (see low-level configurations part in Figure 2). To properly benefit from the "Power PMAC CS" functionality, an individual motor is associated to a CS for physical axis controls, same as for virtual axis controls. Each CS configuration is made by a set of Kinematic forward & inverse routines.

A step-by-step procedure on low-level installation configurations and files archiving procedures for maintenance will be defined.

Integration Test Results

Unitary tests of the standardized software and hardware have been realized and validated. The integration test in the SOLEIL control system is validated with the Power Brick LV in a laboratory setting. Some details need to be finalized.

Several low-level tests have been accomplished or are ongoing on some beamline applications: DCM (double

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

crystal monochromator) on PLEIADES, DCM on SAMBA, NANOPROBE (joint project between Synchrotron SOLEIL in France and MAXIV in Sweden) [4]. The integration tests are also scheduled for these Beta-tester applications.

SAMBA beamline DCM control upgrade (see Fig. 6) Currently the system is controlled by the CLASSIC solution. This upgrade aims to achieve a low-level direct energy control and thereafter a continuous energy-scan operation, which is expected to significantly reduce the experiment duration with faster execution and reduced Ethernet communication.

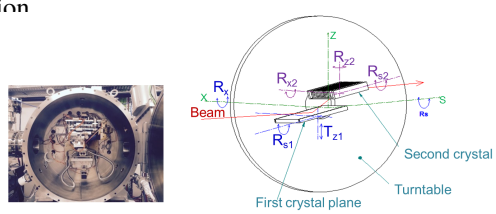


Figure 6: SAMBA beamline DCM motion system.

The control of the 7 main axes (R_x, T_{s2}, T_{z2}, C_1, C_2, R_{z2}, R_{s2}) have been installed on a Power Brick unit. It includes a DC brush and 6 steppers for which the low-level settings have been tested and validated. The kinematic equations have also been implemented to provide direct energy control, detailed as following.

Equation between E and the main axis $R_x(°)$ is shown as below, in which E is the photon energy in eV, d is the lattice spacing in Å, θ is the angle of the main axis of the monochromator R_x in °.

$$E = hc\frac{1}{\lambda} = \frac{hc}{2d}\frac{1}{\sin(\theta)} \tag{1}$$

The remaining motors need to be synchronized with the main axis to obtain the desired energy.

T_{s2} and T_{z2} are the two translations of the second crystal:

$$T_{s2} = \max(T_{s2}^{Min}, \frac{H}{2\sin(\theta)});\ T_{z2} = \frac{H}{2\cos(\theta)} \tag{2}$$

C_1 and C_2 are the positions of the two motors of the second crystal bender. As shown in the equations below, R is the radius of the crystal obtained; $\frac{1}{R}$ is the curvature. $A_{i,j}$ are coefficients:

$$C_1(\frac{1}{R}) = A_{1,0} + A_{1,1}\frac{1}{R};\ C_2(\frac{1}{R}) = A_{2,0} + A_{2,1}\frac{1}{R} \tag{3a}$$

$$\frac{1}{R} = \frac{1}{2\sin(\theta)}(\frac{1}{p} + \frac{1}{q}) \tag{3b}$$

R_{z2} and R_{s2} follow empirical correction curves. Shown in the equations below, p and q are the source to monochromator and monochromator to sample distances.

$$R_{s2} = P_n(\theta, c_{R_{s2}});\ R_{z2} = P_n(\theta, c_{R_{z2}}) \tag{4a}$$

$$p_n(\theta, c) = \sum_{i=0}^{n} c_i\theta^i\ c = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \end{pmatrix} \tag{4b}$$

Tests in low-level with the kinematic conversions have been done and integration tests with the Tango devices will be done soon. The additional functions of enabling or disabling one of the relations for test or to perform a scan, will be realized with the Tango Device "AppliParameter".

New Complementary Product

To meet the required applications when external drivers are in use, a new complimentary product Delta Tau Power Brick controller is now specified. It contains the same CPU (Power PMAC) as Power Brick LV without the built-in amplifier, providing PFM and analog signals for third-party drives. In addition, this product should be fully compatible with the CLASSIC controller in respect to cabling connectivity; an eventual control upgrade would be fairly direct and easy.

The same tool settings as well as skill-sets would be used to configure the new controller. It is expected to be integrated in the SOLEIL control system without extra development.

CONCLUSION

The new strategy of moving from a model with a single universal motion controller to a model with two specialized controllers is being implemented. Operational continuity is ensured with the new CLASSIC controller while at the same time improving some functionalities. The chosen HIGH PERFORMANCE controller, Delta Tau Power Brick, is very powerful and versatile which allows it to handle high performance systems.

The main challenge with this controller is to offer the high-performance of the Power Brick and the DMC-4183 to the end user while still keeping an easy-to-use interface. Both controllers are now ready to be used for simple functionalities and the next step will be to train the electronic group to have the necessary skill-sets to maintain these products in operation. Advanced functions, such as compensation tables and remote managing of motion programs, will be improved. The HIGH PERFOMANCE controller will bring value to different control upgrade applications in projects such as monochromator, Flyscan, Nanoprobe and Goniometer.

REFERENCES

[1] D.Corruble et al., "Revolution in Motion Control at Soleil: How to Balance Performance and Cost", MOPPC013, Proc. of ICALEPCS'2013, San Francisco, CA, USA (2013)

[2] Observatory-Sciences, Delta Tau Power PMAC Communications Library Manual, Issue 1.2, Dated March 17, 2015

[3] Power PMAC User's Manual, Delta Tau Document 050-PRPMAC-0U0, Dated Jan 6, 2015

[4] C. Engblom et al., "High Stability and Precision Positioning: Challenges to Control and Innovative Scanning X-Ray Nanoprobe", Proc. of ICALEPCS'2015, Melbourne, Australia (2015)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

IBEX - THE NEW EPICS BASED INSTRUMENT CONTROL SYSTEM AT THE ISIS PULSED NEUTRON AND MUON SOURCE

M. J. Clarke, F. A. Akeroyd, K. Baker, G. Howells, D. Keymer, K. J. Knowles,
C. Moreton-Smith, D. E. Oram, ISIS, STFC Rutherford Appleton Laboratory, Oxon, UK
M. Bell, I.A. Bush, R.F. Nelson, K. Ward, K. Woods, Tessella, Abingdon, Oxon, UK

Abstract

Instrument control at ISIS [1] is in the process of migrating from a mainly locally developed system to an EPICS [2] based system. The new control system, called IBEX, was initially used during commissioning of a new instrument prior to a long maintenance shutdown. This first usage has provided valuable feedback and significant progress has been made on enhancing the system during the facility maintenance period in preparation for the move onto production use. Areas that will be of particular interest to scientists in the future will be linking feedback from live data analysis with instrument control and also providing a simple and powerful scripting interface for facility users. In this paper we will cover the architecture and design of the new control system, our choices of technologies, how the system has evolved following initial use, and our plans for moving forward.

BACKGROUND

The ISIS pulsed neutron and muon source has been providing world-class science since the mid-1980s and currently has over thirty beamline instruments producing world-leading research. The software and hardware used to control these beamline instruments plays a significant part in the productivity of ISIS.

The current instrument control system, known as the SECI system, consists of LabVIEW [3] drivers for equipment control alongside in-house developed software for scripting, data collection and experiment management, etc. This control system has performed well for many years; however, the construction of new beamline instruments and the refurbishment of older instruments has led to significantly more complex experiments becoming possible and, as a result, are pushing beyond the boundaries for which the current control system was designed. With this in mind, and with the prospect of even more complex instruments being built in the future, it was decided to create a new control system based on EPICS.

HARDWARE ARCHITECTURE

The IBEX control system runs on a Windows 7 x64 Virtual Machine (VM) which is hosted on a rack-mounted Windows 2012 server. Each beamline instrument has its own independent server. Windows was chosen for a number of reasons, such as:

- The controls team have significant experience and expertise with running control systems on Windows
- ISIS has an IT support group that can assist with installation and maintenance
- Easier interfacing to existing site infrastructure (e.g. data archiving, system backup/replication)
- Can use existing LabVIEW drivers more easily
- Ability to do a phased migration, running either in parallel or a mixed IBEX/SECI system

The beamline instrument server and equipment are usually placed on the general ISIS network; however, network sensitive equipment can be placed on a private network. This private network is implemented using a Virtual Local Area Network (VLAN) which can be routed back to the main computer room. As the virtual machines are replicated to the computer room it is possible to rapidly restore and run an instrument from there in the event of beamline server failure.

Most beamline and sample environment equipment in use on ISIS instruments are controlled via serial (RS232) communication. For ease of access and flexibility MOXA [4] NPort serial device servers are used to expose the serial devices to the network.

SOFTWARE ARCHITECTURE

A system based on EPICS lends itself naturally to a client-server style architecture and this is how IBEX has been implemented. Figure 1 shows the basic flow of information relating to the experiment; the dashed line indicates the separation between the client and server side components. Where possible the system uses EPICS Channel Access (CA) for communication between the client and server side.

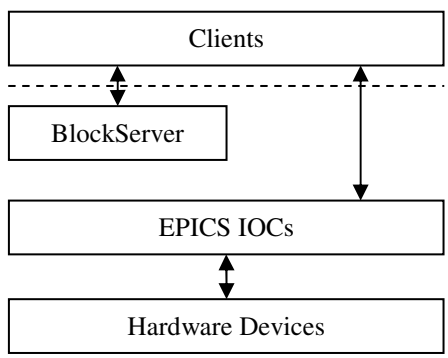


Figure 1: The flow of experiment information for a beamline instrument.

EPICS Input/Output Controllers (IOCs) are the standard way of communicating with the devices used on

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

the instruments. The procServ [5] package is used to host the IOCs as it provides telnet access to the IOC console. The conserver program [6] is used to provide remote access to these consoles for diagnostic and support purposes and the procServControl [7] package adds support for starting and stopping the IOC remotely via Channel Access.

The BlockServer is the brain of the control system; it coordinates the client and the server side to meet the needs of the type of science being performed on the beamline instrument being used. The BlockServer is responsible for managing the setting up of the system for the particular experiment being performed. The information for each required collection of devices is stored as XML files – these XML files make up what is known in ISIS as a configuration. The configurations contain the following information:

- Which IOCs to start and key IOC macro values
- Which Blocks to create
- Metadata relating to the type of experiment
- Information for configuring the IBEX graphical user interface (such as Block display grouping)

At the most basic level a Block is a scientist friendly alias for an EPICS Process Variable (PV). For example, rather than working with a complicated PV like IN:LARMOR:EUROTHERM:TEMP1 an alias can be created called SampleTemp. This simple science specific name is much more intuitive for a scientist. Blocks are usually created for key experiment parameters and the values are logged for inclusion in the NeXus data file [8] created at the end of the experiment. Block PVs are implemented using aliasing on a PV Gateway [9] which in this instance is running on the loopback interface of the server.

All important PVs on an instrument are automatically logged via the Control System Studio [10] archiver application running on the server. As Blocks are experiment based they are logged in a second instance of the archiver to allow easier adjustment and tailoring of archiving performance. The archiver pushes the log information into a MySQL database [11] on the server. The historical log can be used for monitoring and diagnosing problems and the Blocks log provides information for inclusion in the NeXus data file.

Storing information for configuring the IBEX graphical user interface (GUI) on the server side means that the GUI is instrument independent. This is very convenient for the scientists and support staff as it allows quick switching between different instruments through one GUI. Having a common GUI across all instruments also makes support and maintenance easier.

Although it is not shown on Figure 1 there is also a second PV Gateway on the server which is used to control access to the PVs. Using the CA security built into the Gateway it is possible to restrict access to the PVs on the server; this can be based on user name and/or the machine being connected from. By default, access to the IBEX server is restricted to being read-only, but read-write access is allowed from certain machines.

One of the weaknesses of the SECI system was that there was only one GUI client instance available and that was on the server. This meant that the system could not be viewed easily by multiple people at the same time. IBEX has been developed to allow multiple instances of the client to be run at the same time from any PC on the ISIS network. By default these clients will be read-only but the intention is to develop a baton handling system to allow clients to request to have sole read-write access for an appropriate amount of time.

Also not shown on Figure 1 is the messaging system which allows messages from the IOCs and BlockServer to be passed to the IBEX GUI. The messaging system is part of Control System Studio and is based on the Apache ActiveMQ implementation of the Java Messaging System (JMS) [12]. A new log server process replaces the usual IOC log program and routes IOC messages into this JMS system. A historical record of the messages is stored on the server inside the MySQL database and these historical messages can be searched via the IBEX GUI.

One of the challenges for a control system at a user facility like ISIS is incorporating new bespoke equipment quickly; for example, a user from a university may bring a prototype piece of sample environment equipment to use on an instrument. To handle these use cases there are different technologies which have been developed at ISIS for handling LabVIEW and other simple serial devices.

As the SECI system was based on LabVIEW it was desirable to be able to quickly enable existing drivers into the EPICS system. For this, a driver based on Asyn [13] was created, called lvDCOM [14], which communicates with LabVIEW via DCOM [15]. The creation of the lvDCOM driver is mostly automated and because the LabVIEW driver requires no modification it is quick and simple to create an IOC which wraps the LabVIEW driver. It is reasonably common for bespoke equipment to have a LabVIEW driver, thus lvDCOM can be used in those cases. For more complex situations it is possible to use the NetShrVar driver [16] to access National Instrument network shared variables [17] instead.

SDTest is a simple generic serial solution based on the EPICS StreamDevice [18]. With knowledge of the command strings to send to the device and how to interpret the corresponding response a StreamDevice driver can be quickly created as part of a configuration.

THE IBEX CLIENT

For IBEX it was decided to create a GUI based on Control System Studio. Control System Studio was chosen as it was the most feature complete of the modern EPICS GUI implementations. Control System Studio itself is based on the Eclipse Rich Client Platform (RCP) framework [19].

The overall design of the IBEX GUI was based on the GUI of the SECI system (see Figure 2) as the ISIS scientists and users liked this design.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

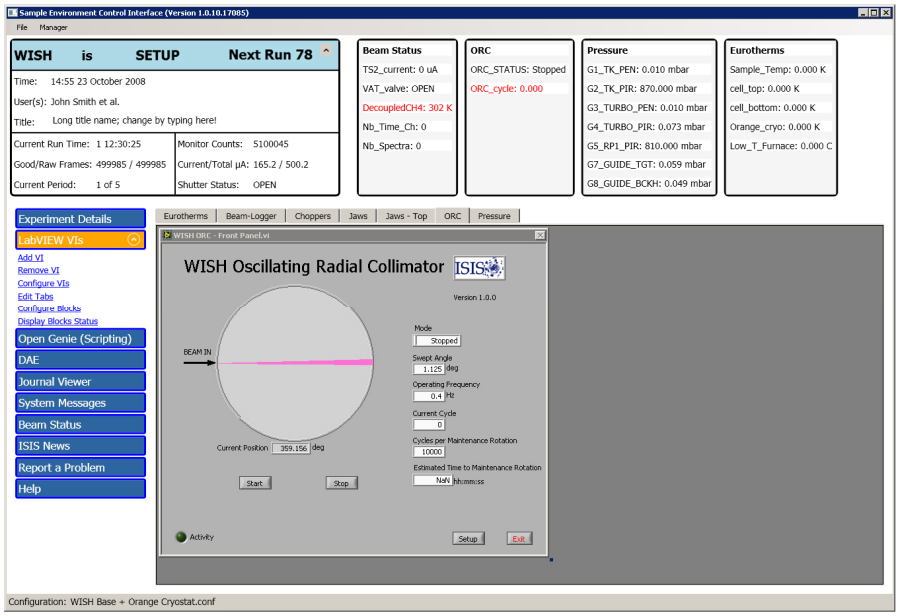


Figure 2: The GUI for the SECI system.

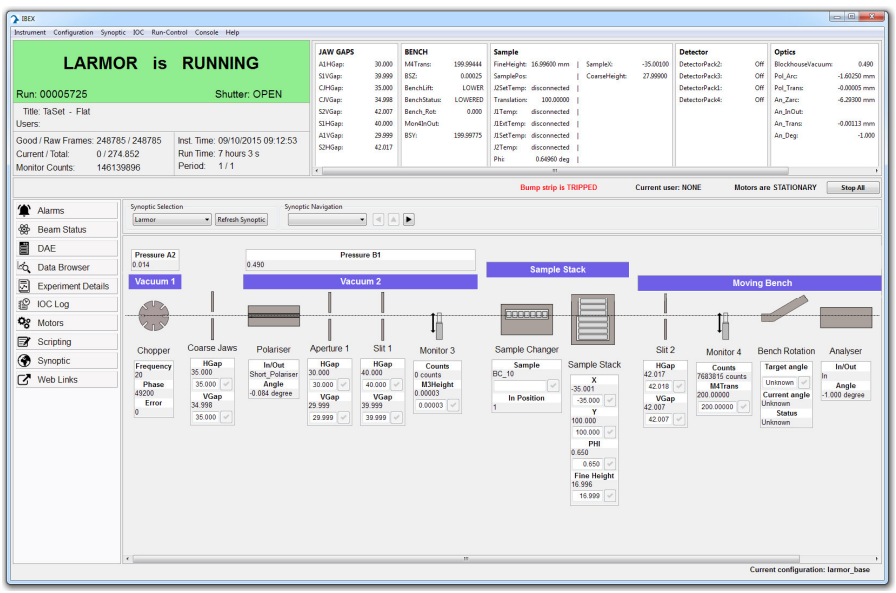


Figure 3: The new IBEX GUI showing an instrument synoptic.

The key features of the design that needed to be reproduced were:

- The data collection dashboard in the top-left corner
- The Block values displayed in groups in the top-right corner
- The navigation bar on the left-hand side

To achieve this it was necessary to create a screen layout very different from a typical Control System Studio screen. ISIS specific views were created and some of the GUI features that are provided by Control System Studio needed to be either hidden or disabled. The current IBEX GUI is shown in Figure 3. New key features required for the IBEX GUI were a synoptic, a data browser and an embedded Python console for scripting.

The synoptic (as shown in Figure 3) provides an overview of the status of the instrument in terms of the layout of the instrument and key values. By selecting an item on the synoptic it will open a window showing more information about that particular item. For this Control System Studio's BOY screens are used, an example of which is shown in Figure 4. An instrument may have multiple synoptics, either for different experiments or for showing various areas of the beamline separately. The layout information for the synoptics is served and managed by the BlockServer which is one of the ways the IBEX GUI is kept instrument independent. The GUI also

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

has a graphical synoptic editor which allows the scientists to create new synoptics or modify existing ones.

The data browser is a plotting tool developed as part of Control System Studio; it is designed to show historical PV values as a function of time. In the IBEX GUI it is used to plot the history of Blocks as well as standard PV values. The historical data is retrieved from the MySQL database mentioned previously.

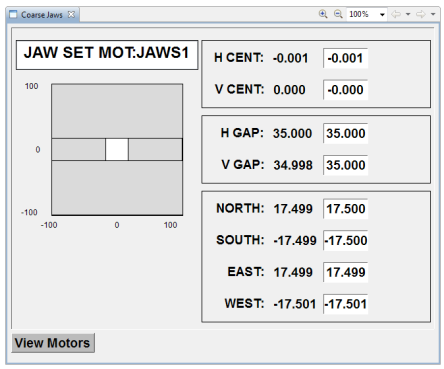


Figure 4: An example BOY screen.

The SECI system used an in-house scripting language called Open GENIE [20] for creating scripts to automate a series of experiments. For the IBEX system Python was chosen as the replacement for Open GENIE because it is easy to learn and many scientists will already be familiar with it. PyDev [21] is a Python integrated development environment and because it is an Eclipse RCP application it was simple to incorporate the PyDev Python command line directly into the IBEX GUI. To enable Python to communicate with the IBEX control system an in-house library was created called genie_python which uses CaChannel [22] for sending and receiving information over Channel Access and provides scientists with a set of useful control commands.

STATUS UPDATE

The IBEX system has been successfully used to commission the LARMOR small angle scattering instrument and is now being used to perform user experiments. The scientists using a prototype IBEX system to commission LARMOR provided invaluable testing time and feedback for the development of the system. Now IBEX is being used for performing user experiments more feedback is being received which will be used to direct further development.

One of the goals for IBEX was for it to be easy to incorporate bespoke equipment quickly. With this in mind lvDCOM, NetShrVar and SDTest were developed. Users on LARMOR have already successfully used SDTest to enable their own specialist equipment to be used within IBEX.

The intention was to develop a system where an instrument could be observed from multiple instances of the GUI at the same time and this has been achieved. Being able to monitor an instrument without disturbing the user has been a very useful feature for both the controls team and the scientists. Overall the IBEX GUI has been well received by the scientists - maintaining the familiar look of the SECI GUI whilst adding powerful new features like the synoptic has proved popular.

The general flexibility of having a system based on EPICS compared to LabVIEW has allowed a more future resilient control system to be developed. Being part of wider collaborations, such as EPICS and Control System Studio, have reduced the amount of development required to create the new control system.

THE FUTURE

The Mantid data analysis framework [23] is widely used at ISIS for analysing data after an experiment but there is a desire amongst the scientists for “intelligent data collection.” An example of this might be stopping data collection when the signal to noise ratio of the data reaches a certain level. As Mantid is Python-aware it is possible to either call genie_python from Mantid or vice versa – this could be a way of providing intelligent data collection. Initial investigations into enabling Mantid to communicate with the control system have proved promising. This is an area which will be explored further.

IBEX is currently being used to commission the new IMAT instrument which presents a new set of challenges. IMAT is an engineering beamline and, thus, is very different to LARMOR; however, this is the kind of challenge IBEX was developed to handle.

After IMAT there will be another new instrument called ZOOM which will also be commissioned using IBEX. Over time the existing instruments running the SECI system will be migrated to IBEX.

REFERENCES

[1] http://www.isis.stfc.ac.uk
[2] http://www.aps.anl.gov/epics/
[3] http://www.ni.com/labview/
[4] http://www.moxa.com
[5] http://sourceforge.net/projects/procserv/
[6] http://www.conserver.com
[7] http://controls.diamond.ac.uk/downloads/support/
[8] http://www.nexusformat.org
[9] http://www.aps.anl.gov/epics/extensions/gateway/
[10] http://controlsystemstudio.org
[11] https://www.mysql.com
[12] http://activemq.apache.org
[13] http://www.aps.anl.gov/epics/modules/soft/asyn/
[14] http://epics.isis.stfc.ac.uk/wiki/LVDCOM
[15] https://msdn.microsoft.com
[16] http://epics.isis.stfc.ac.uk/wiki/NETSHRVAR
[17] http://www.ni.com/white-paper/5484/en/
[18] http://epics.web.psi.ch/software/streamdevice/
[19] https://eclipse.org
[20] http://www.opengenie.org
[21] http://www.pydev.org
[22] https://cachannel.readthedocs.org
[23] http://www.mantidproject.org

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

100 Hz DATA ACQUISITION IN THE TANGO CONTROL SYSTEM AT THE MAX IV LINAC

P. Bell, V. Hardion, V. Michel, Max IV Laboratory, Lund, Sweden

Abstract

The MAX IV synchrotron radiation facility is currently being constructed in Lund, Sweden. A linear accelerator serves as the injector for the two storage rings and also as the source of short X ray pulses, in which mode it will operate with a 100 Hz repetition rate. The controls system, based on TANGO, is required to collect and archive data from several different types of hardware at up to this 100 Hz frequency. These data are used for example in offline beam diagnostics, for which they must be associated to a unique electron bunch number. To meet these requirements, the timing performance of the hardware components have been studied, and a TANGO Fast Archiver device created. The system is currently in the deployment phase and will play an important role in allowing the linac and Short Pulse Facility reach their 100 Hz design goal.

INTRODUCTION

The accelerator complex at the MAX IV laboratory consists of a 3 GeV, 250 m long full energy linac, two storage rings of 1.5 GeV and 3 GeV and a Short Pulse Facility (SPF). The repetition rate of the linac is a maximum of 10 Hz when serving as the injector for the rings and 100 Hz when providing pulses for the SPF.

The control system has a three-tier architecture, with specific hardware handling the real-time tasks and TANGO [1] representing the middle tier as the primary control system. Most equipment is interfaced directly to TANGO via TCP/IP. For the client layer, physicists and operators can interact with TANGO via its Python and MATLAB bindings or through the SARDANA [2] layer, which brings a macro server and standardised Graphical User Interfaces based on TAURUS [3].

In the operation of the linac it is necessary to be able to archive data through the TANGO system at the repetition rate or bunch frequency of the electrons. For example, oscilloscope devices connected to Current Transformers at various locations along the length of the linac are used to record the distribution in time of the charge of the passing electron bunches. These waveform data will be used, for example, in the offline diagnosis of any beam loss. The capturing of information from the control system at high frequency is referred to as Fast Archiving, since it must perform at up to 100 Hz to record every electron bunch when the linac is operating in SPF mode. The same system will also be used to read data from the linac and the storage rings at 10 Hz, where it will be used, for example, in the Slow Orbit Feedback system.

The work that is reported here has so far has focused on the integration of TANGO devices for controlling oscilloscopes into the Fast Archiver, though the inclusion of other equipment such as power supplies is foreseen in the future. The first section of the report documents how the performance of the many devices that will participate in the Fast Archiving is being systematically measured. The second section then describes the design and functioning of the Fast Archiver and how the oscilloscope devices have been adapted for inclusion in the system.

TIMING PERFORMANCE STUDIES

Timing performance studies are being performed on the TANGO devices that will take part in the Fast Archiving. The performance studies first check that the hardware itself is capable of sending or receiving commands at a sufficiently high frequency. An example is shown in Fig. 1 for a certain type of power supply used at Max IV.

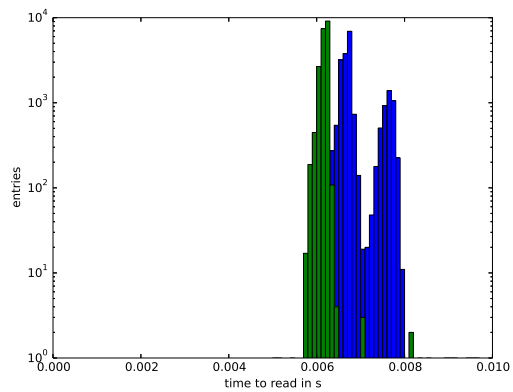


Figure 1: Distribution of time taken to read the current from a power supply over a direct raw socket connection (green) and via the "Current" attribute in its TANGO device integrated to the control system (blue).

A script is used that repeatedly reads, for example, the value of the current from the power supply hardware and plots the distribution of the time needed to receive a reply. In this example the distribution (in green) in well within the 10 ms envelope needed for 100 Hz operation. The performance of the TANGO device is then checked by measuring the distribution of the time needed to read the attribute "Current" once the device has been interfaced to the TANGO control system. The difference between the two distributions gives an indication of the overhead of the control system, including the network performance. In this case, the distribution (in blue) remains within the 10 ms envelope though adds a few ms to the "raw" performance of the hardware over a direct TCP/IP connection. It can be concluded that this hardware should be suitable for 100 Hz operation.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

At the present time we are using these simple tests to identify which devices will need improvements in their communication speed if they are to participate in the Fast Archiving in the future.

THE FAST ARCHIVER

Oscilloscope Device Adaptation

Similar studies to those described above have been performed on the Rohde and Schwarz RTO oscilloscopes [4] that are used in the linac to measure the charge-time distributions of the passing electron bunches. For these devices, the attribute of interest is the "Waveform" which is a spectrum attribute with length of up to 10,000 corresponding to the captured data from the Current Transformer hardware. This attribute can be read from all channels of the oscilloscope simultaneously in around 3 ms as shown in Fig. 2, so the hardware performance is proven to be acceptable.

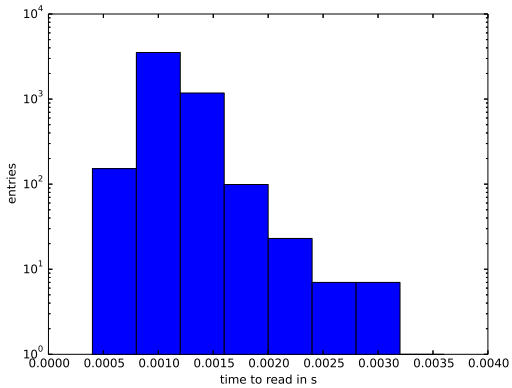


Figure 2: Distribution of time taken to read the "Waveform" attribute from the Rohde and Schwarz oscilloscope TANGO device. The device reads all four channels in the same request.

The oscilloscopes receive an external trigger synchronised with the repetition rate of the linac. Libera timing modules [5] make this trigger signal available in the TANGO system, with the Libera TANGO devices sending TANGO events as the trigger counter increments. The event system in TANGO is known to perform many times faster than 100 Hz. In the first attempt to synchronise the reading of the waveforms to the trigger counter, the oscilloscopes were configured to continually capture waveforms on the external trigger, while the TANGO device subscribed to events from the Libera. The reception of an event from the Libera prompted the request of the the waveform data, which at that moment should correspond to the previous trigger. However, to guarantee that the waveform read out corresponds to the one just captured by the hardware, this mechanism relies on the event jitter being less than around 7 ms (given the 3 ms needed to read out the waveform data) which could not be assured. A method that did not rely on the timing performance of the TANGO event system was therefore sought. The oscilloscopes are now configured to wait for the external trigger and block until a new waveform is captured. This ensures that any waveform read out after the block is released is guaranteed to correspond to the previous trigger. The TANGO device implements a thread which repeatedly runs these single acquisitions. Each time a new acquisition is made, the device sends an event containing the waveform data (and its timestamp). In this way the oscilloscopes respond to their external triggers but do not rely on any further software triggering of the readout of the waveform data; rather, the oscilloscope TANGO device is responsible for pushing events as the oscilloscope receives external triggers.

Fast Archiver Design

The purpose of the Fast Archiver is to associate data collected from any participating TANGO device to a unique electron bunch number in the linac. The bunch number (or trigger counter) comes from the Libera timing modules; as described above the Libera TANGO devices are configured to send TANGO events containing the trigger number each time it increments, and these events also contain the timestamp of the trigger. The only requirement on the TANGO devices participating in the Fast Archiving is that they send events containing the data to be archived with a timestamp, as explained above for the case of the oscilloscope device. Given these constraints, the design of the Fast Archiver system is then shown in Fig. 4.

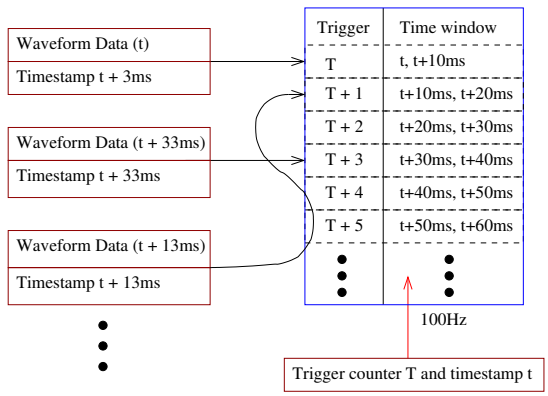


Figure 3: The Synchroniser device receives events containing the trigger counter, T, and timestamp, t, from the Libera device and events containing the data to be archived and corresponding timestamps from one or more other devices. Irrespective of the order in which the events arrive, the data to be archived can be associated with the correct trigger number by comparing timestamps. In this example, waveform data will typically arrive 3 ms after the event from the Libera.

The system comprises one or more Synchroniser devices (Fig. 3) which receive events from the Libera and also from one or more devices that are pushing data to be archived. The Synchroniser devices make the association of the data with the correct trigger number by comparing the timestamp

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

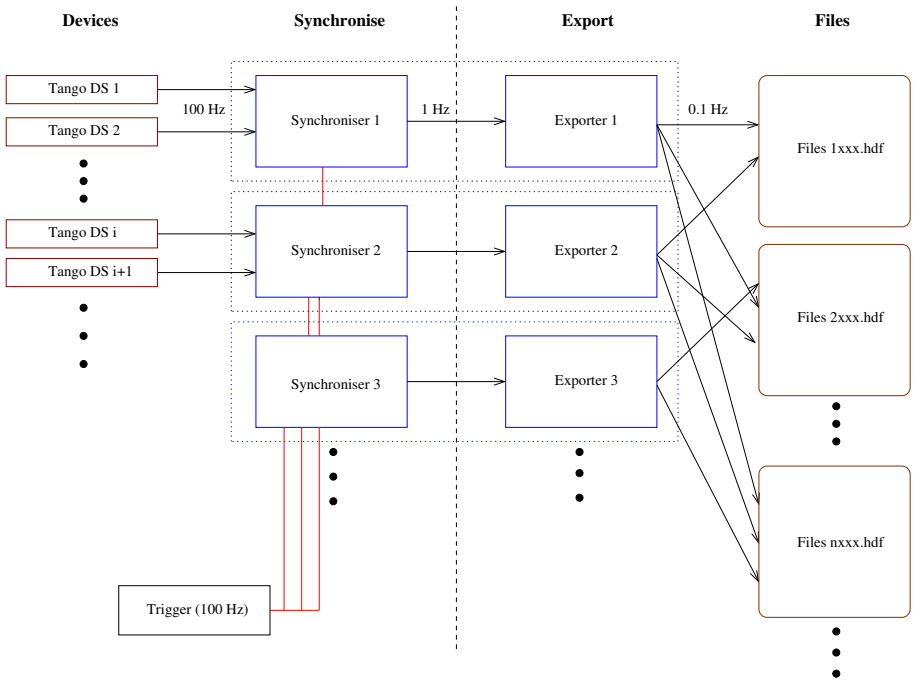


Figure 4: Design and operation of the Fast Archiver in 100 Hz scenario. Multiple Synchroniser devices subscribe to events from the participating hardware, arriving at 100 Hz. An Exporter process buffers these at 1Hz in association with the correct trigger number (see Fig. 3). Every 10 s the data, now with an associated trigger number, is written to files.

of the Libera events to the timestamps that come with the data to be archived. In this way the data from the hardware participating in the Fast Archiving becomes correctly associated with the unique bunch number, irrespective of the order in which the events arrive at the Synchroniser. A validity flag is used to handle any ambiguities in the association of the trigger number, for example if two data events arrive with timestamps that may both be matched to the same trigger.

The Synchroniser devices each implement an Exporter process, which takes the data buffered with the corresponding trigger counter and writes them to an hdf5 file. A single file is created to hold a certain number of events, and is written to by all Synchroniser devices. As such, the data from all participating hardware for a given range of trigger numbers ends up in the same file.

CONCLUSIONS

A system has been designed to allow the archiving of data from the TANGO control system at up to 100 Hz, assuming that the data orginates from devices that are capable of addressing their hardware within a 10 ms time interval and can push the data together with their timestamps as TANGO events. A single Fast Archiver device taking the data from a single oscilloscope device has been successfully tested in the laboratory at 100 Hz. For the MAX IV linac a Fast Archiver comprising 18 Synchroniser devices has been deployed. These are currently archiving 152 waveform attributes from 38 oscilloscope devices, though the maximum repetition rate of the linac has so far been 0.50 Hz.

ACKNOWLEDGMENT

The authors would like to thank the whole of the Controls and IT Group (KITs) at the Max IV Laboratory, along with the physicists and operators, without the collaboration of which this work would not have been possible.

REFERENCES

[1] The TANGO Control system website:
`http://www.tango-controls.org`

[2] SARDANA software website:
`http://www.sardana-controls.org`

[3] TAURUS website:
`http://www.taurus-scada.org`

[4] Rohde and Schwarz RT0 1024 Digital Oscilloscope on the R&S website:
`https://www.scope-of-the-art.com/en/rto/0`

[5] Libera Single Pass E, Electron beam position processor for single pass machine:
`http://www.i-tech.si/accelerators-instrumentation/single-pass-e/`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TANGO-KEPLER INTEGRATION AT ELI-ALPS*

Péter Ács, Sándor Brockhauser, Lajos Jenő Fülöp, Veronika Hanyecz, Miklós Kiss, Csaba Koncz, Lajos Schrettner, ELI-ALPS, Szeged, Hungary

Abstract

ELI-ALPS will provide a wide range of attosecond pulses which will be used for performing experiments by international research groups. ELI-ALPS will use the TANGO Controls framework to build up the central control system and to integrate the autonomous subsystems regarding software monitoring and control. Beside a robust central and integrated control system, a flexible and dynamic high level environment could be beneficial. The envisioned users will come from diverse fields including chemistry, biology, physics or medicine. Most of the users will not have programming or scripting background. Meanwhile workflow system provides visual programming facilities where the logics can be drawn, which is understandable by the potential users. We have integrated TANGO into the Kepler workflow system because it gives a lot of actors for all natural scientific fields. Moreover it has the potential for running the workflows on HPC or GRID resources. We demonstrated the usability of the development with a beamline simulation. The TANGO-Kepler integration provides an easy-to-use environment for the users therefore it can facilitate e.g. the standardization of measurements protocols as well.

INTRODUCTION

ELI-ALPS is one of the three pillars of the European Extreme Light Infrastructure project. The Attosecond Light Pulse Source (ALPS) facility will provide a wide range of ultrafast light sources (such as coherent XUV and X-ray attosecond pulses) for performing material, condensed matter and surface science, chemical, biological, physical or medical experiments. Besides, the development of the technology for generating 200PW peak intensity pulses is also a main mission. As a research facility, the infrastructure will contain a large number of experimental devices and equipment which have to be managed and controlled by a robust and flexible system. We found that the TANGO Control system [1] is able to address this complexity, it has already been used at several large research infrastructures for more than a decade.

TANGO Controls is an open-source control system framework which provides a foundation to develop control systems. Architecture of a TANGO-based control system has three main components: the Device Server, the Database and the Client. The Device Server component is the base element that provides monitoring and control capabilities over a set of devices. The Database includes information about the Device Servers and Devices. A TANGO Client may connect to Device Servers using the reference acquired from the Database to perform actions on a certain Device. Experimentalist and scientist users may create programs by using the available APIs. However, not all of the users have a programming or scripting background necessary for such a task.

Scientific workflow systems [2-3] provide a graphical interface to create scientific applications without a deep scripting and/or programming background. A workflow is basically a collection of jobs connected to each other. Each job has input(s), performs a specific function and produces output(s) that can be used by other jobs. The jobs are usually represented as boxes with input and output ports. This kind of graphical representation of unit jobs increases productivity and helps maintainability. It allows the user to save the workflows as well as to share them with others.

The motivation of the paper is to develop a prototype integration of the Kepler scientific workflow system [4] with the TANGO framework. This way a high-level graphical environment could be provided for the users to create programs controlling and monitoring research equipment. Some other institutes already follow a similar approach [5-6]. However, this work has some further motivations, e.g. to collect relevant experience about how to integrate the TANGO framework into other environments.

THE INTEGRATION

A Kepler workflow has an arbitrary number of actors and exactly one director. Every actor represents a job with its necessary input and output ports, and an additional configuration dialog may help to fine-tune the behaviour. Actors communicate by sending messages via interconnected ports. The role of the director is to supervise the execution of the workflow, e.g. it manages the number of executions, enables parallelization if possible, etc.

A fundamental component of the TANGO Control System is the TANGO Device Server, which represents a set of devices with all of their attributes, commands, events and other relevant information. The goal was to create a universal *TANGO actor* in Kepler, which can dynamically adopt any functionalities (commands, attributes, etc.) a TANGO Device may have. However, it is implemented in a way that when the TANGO actor is instantiated, then the necessary functions can be chosen and will only be available afterwards.

It is not completely obvious how to represent and execute a TANGO workflow in Kepler because the functional structure of Kepler is fundamentally different from that of TANGO. As it was mentioned before, a Kepler workflow is controlled by directors.

* The ELI-ALPS project (GOP-1.1.1-12/B-2012-000, GINOP-2.3.6-15-2015-00001) is supported by the European Union and co-financed by the European Regional Development Fund.

† https://www.eli-alps.hu/?q=en

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Kepler inherited five directors from Ptolemy II: PN (Process Network), DE (Discrete Events), CT (Continuous-Time), DDF (Dynamic Dataflow) and SDF (Synchronous Dataflow). Every director is suitable for a different kind of task. For example an SDF might be an appropriate choice for a simple data transformation, however, for example it is not the right decision if one would like to create time-based simulation workflows, because it has no understanding of passing time.

Examining the directors we found that the PN Director is the best choice as a TANGO workflow execution supervisor. A process network director models a workflow as a network of processes where each actor is running on a dedicated thread and generates output data only if all of the input ports have available input tokens to process [7]. Using this model, actors can run simultaneously without blocking each other unnecessarily. Therefore a TANGO actor can continuously gather data from a TANGO device while another TANGO actor is waiting for an event to be triggered from the same TANGO device. With the flexibility of the PN Director a workflow is able to reach that level of functional variety that the TANGO system requires. Figure 1 shows the architecture of the TANGO Controls System and the place of the integration in the system (Kepler-Tango binding in the top-right part).

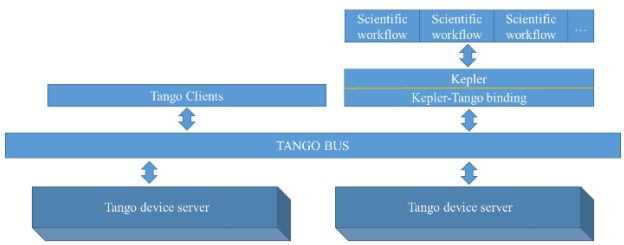


Figure 1: Tango Controls architecture with Kepler and scientific workflows.

Main Concept

A universal actor is created that behaves as a Device Server, e.g. handles the execution of commands, reads/writes attributes while supervises the connection between the Device Server and the actor. The requirements the solution has to fulfil:

- A TANGO Actor represents a TANGO Device
- The user is able to select a TANGO actor from the list of the available ones (i.e. the TANGO Devices in the TANGO database). TANGO Devices may have a complex name which is easy to mistype, therefore bypassing this interaction makes the user's job more convenient and efficient.
- An input/output port of a TANGO actor represents an attribute or a command of the corresponding TANGO Device
- The user is able to filter which TANGO attributes and commands appear as ports of the corresponding TANGO actor. A Device Server may have tens or even hundreds of commands/attributes, therefore filtrating of them makes the user's job more convenient and efficient.

Steps of the Development

The development is separated into four phases. Each phase is ended by a test sub-phase.

Hard-coded Integration. In this phase a basic workflow component with only the necessary functions was implemented. The goal was to discover the Kepler workflow system whether it is capable of adopting unique functionalities of TANGO.

The developed component is able to handle the connection to a predefined, hard-coded device, manage the execution of a preselected command and read/write of some attributes with different types.

Database Level Integration. The TANGO Database may have thousands of entries of devices, therefore it might lead to complex device instance names that the user may not be able to memorize easily and correctly. When one wants to deploy a TANGO device component on the workflow canvas, it requires the domain name of the TANGO Device so the workflow will be able to connect to and perform actions on that. The basic solution would be that the user types in the full path of the device which is time-consuming, inconvenient and the risk of mistyped name is not ignorable. A more sophisticated solution is needed in a form of a list of available devices with which the user is able to put the actors on the workflow canvas easily.

The developed extension is able to query the available devices from the TANGO Database as well as to integrate it in a way that the user can conveniently deploy TANGO devices as TANGO actors. This feature appears in the context menu, in a combo box as well as in a separate tab.

Complete Type Conversion. One of the biggest and most significant challenges of integrating TANGO into Kepler was the complete type conversion.

TANGO has all the main and basic datatypes, e.g. bool, double, string, etc. However, TANGO defines some unique types which makes the process of conversion ambiguous, e.g. array or 2D array of scalar types, doublestringarray and so on.

In Kepler, data is passed with data tokens, and each data type has its own structural type. If one wants to use data coming from a TANGO device, one has to convert the TANGO data into an environment-dependent structural type, and then wrap it into the corresponding data token. For basic datatypes, the conversion is obvious: TANGO's SCALAR DevBoolean can be converted into a Boolean variable, and then wrapped into a Boolean token.

For IMAGE DevString (matrix of strings) the conversion is more complex. First the data has to be converted into an array of array of strings. Although Kepler has a Matrix token, it is not available for strings, therefore the matrix has to be wrapped into an arrayType(arrayType(string)) token. Although these datatypes conversions may take time to implement, they can be hidden from the user. However, there are exotic types in TANGO, which is a challenge for the developer. In case of DoubleStringArray type of TANGO, which is an array of doubles and string, the

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

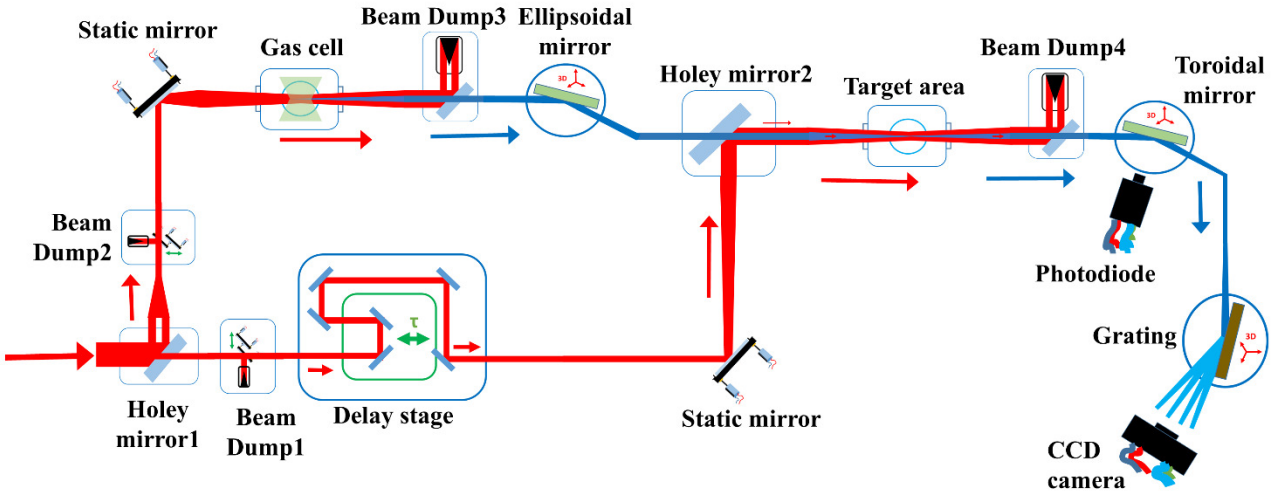


Figure 2: Beamline control system architecture.

developers have to agree on a general integration of how to instantiate it into Kepler.

Configuring ports (attributes, commands). As mentioned before, one TANGO Device may have large number of attributes and the same is true for commands. Most of the components of workflow systems display individual ports for every possible inputs and outputs. Following this concept, a TANGO actor may have hundreds of input and output ports for attributes and commands, which leads to an untraceable communication of workflow actors. Therefore limiting the number of ports for a more user-friendly appearance is a must.

The base concept is that the users should choose the ports (TANGO attributes/commands) in advance. However, the solution should guarantee the opportunity to add and remove ports later as the workflow may change.

In addition, it is restricted that an actor can have only one command, so the role of the actor stays clear and will be easier to handle.

BEAMLINE SIMULATION

For a proof of concept, a simple control system workflow is created on top of a simulated beamline to demonstrate the usability of the prototype. The architecture of the system is shown in Fig 2.

In the beamline simulator the initial InfraRed (IR) beam is split in two parts with a special mirror with a hole in the middle (Holey mirror1). After the beam splitting, both arms have a beam dump with motorized mirrors. The outer ring is focused into a gas cell to generate the XUV attosecond pulse. The generated beam is cleared out from the initial beam by a holey mirror (Beam Dump3). The next element in the beamline is a grazing incidence ellipsoidal mirror with 3D tilting option, and the light is directed to the Holey mirror2. The other arm is delayed with a delay stage and directed to the same mirror. Both beams are directed to the target area. The final beam is cleared out of the IR

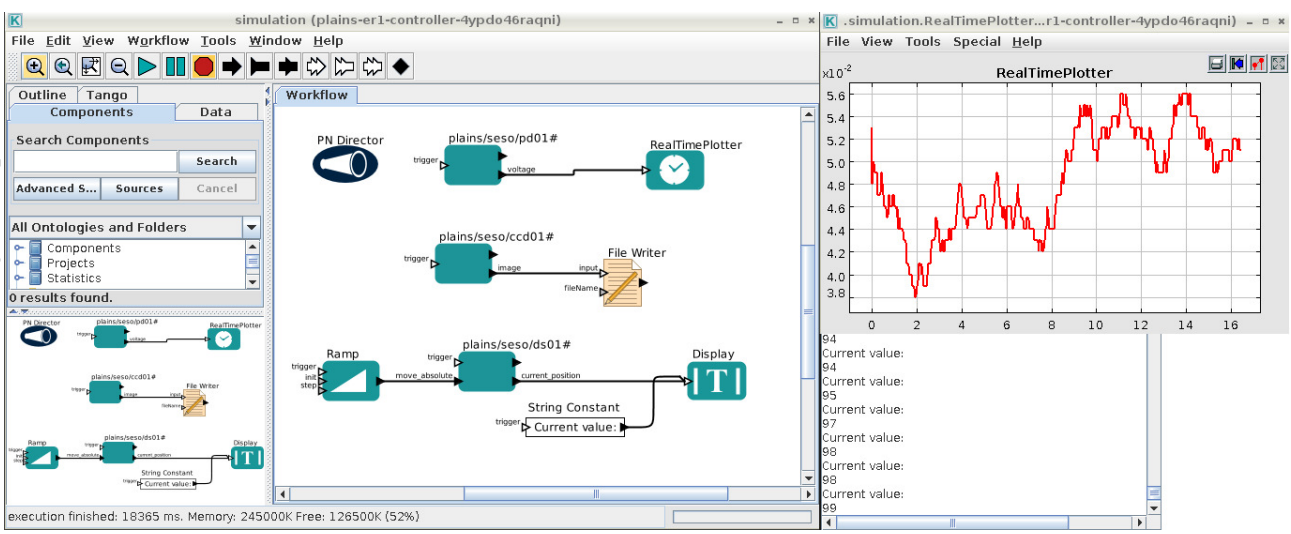


Figure 3: Kepler-TANGO integration: workflow and displayed output.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

beam (Beam Dump4) and the XUV pulse is directed by a toroidal mirror to the Grating and detected by the CCD camera unit. A photodiode is placed in the toroidal mirror's chamber for the pulse intensity monitoring. The delay stage has an important role in the recombination process, so called pump-probe experiment. It is delaying the IR pulse to the XUV pulse. The elliptical mirror and the toroidal mirror are positioning the generated XUV beam. The beam dumps with sliding mirrors (1 and 2) can block the beam in both arms.

From control side the following parts are static: holey mirrors, gas cell, beam dump 3 and 4 and target area while beam dump 1 and 2, ellipsoidal mirror, spherical mirror, photodiode and CCD camera can be controlled. In this Kepler demonstration only the delay stage was moved.

A demonstration of the integration can be seen in Fig. 3. The workflow has eight actors plus one director. The plains/seso/pd01 actor represents the photodiode detector, which provides voltage values every time the director executes the actor. The output port of the voltage parameter is connected to a RealTimePlotter, which draws the voltage value on a plotter as the time goes by. The output of the plotter can be seen in the top right corner of the figure.

The second TANGO actor is the plains/seso/ccd01, which provides images from the CCD camera on its output and which will be saved in a file with the help of the File Writer actor.

The last TANGO actor is the plains/seso/ds01, which represents the delay stage component of the control system. Its move_absolute input port is connected to a Ramp actor. The Ramp actor counts from an initialized value to a maximum or infinite with defined steps. With every step, the magnitude of the delay increases with a short delay. With the help of the display actor, the current value can be followed on a dialog on the bottom right corner.

CONCLUSIONS AND PERSPECTIVES

The ELI-ALPS facility will use the TANGO Control system for managing and controlling the experimental devices and equipment. A prototype integration of TANGO with the Kepler Workflow system has been developed. This development could facilitate the creation and modification of experimental protocols. It could be a big benefit for the users and developers, too.

REFERENCES

[1] A. Götz et al. (2003). "Tango a Corba Based Control System" in Proceedings of ICALEPCS2003, Gyeongju, Korea

[2] V. Curcin, M. Ghanem (2008). "Scientific Workflow Systems – Can One Size Fit All?" in Proceedings of the 2008 IEEE, CIBEC'08

[3] B. Ludäscher et al. (2009). "Scientific Process Automation and Workflow Management" in *Scientific Data Management: Challenges, Technology, and Deployment*, ISBN: 978-1-4200-6980-8, chapter 13

[4] Kepler homepage: https://kepler-project.org

[5] A. Götz et al. (2011). "Data Analysis Workbench" in Proceedings of ICALEPCS2011, WEPKS019, Grenoble, France

[6] G. Abeillé et al. (2007). "A Graphical Sequencer for Soleil Beamline Acquisitions" in Proceedings of ICALEPCS07, RPPB20, Knoxville, Tennessee, USA

[7] C. Brooks et al. (2008). "Heterogeneous Concurrent Modeling and Design in Java (Volume 3: Ptolemy II Domains)", EECS Department, University of California, Berkeley

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ELI-ALPS CONTROL SYSTEM STATUS REPORT *

Lajos Jenő Fülöp, Sándor Brockhauser, Sándor Farkas, Veronika Hanyecz, Miklós Kiss, Miklós Tamás Koncz, Árpád Mohácsi, Kwinten Nelissen, Lajos Schrettner, Balázs Szalai, Péter Szász, Chris Turner, ELI-ALPS, Szeged, Hungary

Abstract

ELI-ALPS will provide a wide range of attosecond pulses which will be used for performing chemical, biological, physical or medical experiments by international research groups. It is one pillar of the first international laser facility for the scientific user communities. ELI-ALPS use the TANGO Controls framework to build up the central control system and to integrate the autonomous subsystems regarding monitoring and control. It will be also used for the implementation of some autonomous systems' control system while others will be implemented differently. The central control system and the integration strategy of the autonomous systems is designed. The centralization and integration needs are surveyed and the requirements are collected. Prototypes have been developed to clarify the requirements and to test the designs. Requirements elicitation, designing and prototype development follows a Lean-Agile approach and includes several fields: device drivers and simulators; integration logic; central supervision, archiving, logging and error recovery; graphical user interfaces and so on.

INTRODUCTION

ELI-ALPS is one of the three pillars of the European Extreme Light Infrastructure project. The Attosecond Light Pulse Source (ALPS) facility will provide a wide range of ultrafast light sources (such as coherent XUV and X-ray attosecond pulses) for performing material, condensed matter, surface science, chemical, biological, physical or medical experiments. Besides, the development of the technology for generating 200PW peak intensity pulses is also a main mission. As a research facility, the infrastructure will contain a large number of experimental devices and equipment which have to be managed and controlled by a robust and flexible system. We found that the TANGO Control system [1] is a good candidate, it is already used by several research institutions, mostly in synchrotrons, but also in laser projects [2]. The basic layout of ELI-ALPS is the following: different primary laser sources generate laser pulses with different characteristics that are delivered by the beam transport lines to the secondary sources, which then generate attosecond light pulses for the experiments.

In the first phase of the development of the ELI-ALPS Control System the fundamental concepts, the high-level architectural design and the frames of the development process have been identified. In this paper the most important requirements, concepts, aspects, and prototypes are summarized briefly.

REQUIREMENTS

The general requirements are to follow public standards [3] as well as in-house policies and handbooks that also declare the application of frameworks such as TANGO and Taurus [4]. In order to maximize efficiency, different techniques are planned to be used. Automation techniques will be applied as much as possible, e.g. automatically registering devices [5], generate and configure GUIs and so on. Simulation environments, mimicking the devices and some key relationships will allow to perform most of the software development and testing independently from the real hardware.

Scientific Systems

Laser sources will be delivered as black-box, turn-key systems together with hardware and software. The secondary source systems are addressed by dedicated projects. The requirements, the technical design, as well as the hardware shopping list are provided by expert institutes of the corresponding secondary source field. The beam transport systems are addressed by a dedicated in-house project, which provides the requirements, the technical design and the hardware. Software and integration will be covered by another dedicated project.

Central Control System

The Central Control System will include the following services. The *archiving* subsystem collects all the preconfigured local or central variables for later usage and trends. The *alarms* subsystem collects and shows all the local and central alarms that are the result of evaluating predefined formulas on attribute values. The *logging* subsystem collects all the logs.

The *Integration Platform* supports, manages and orchestrates the collaboration of the systems (lasers, secondary sources, beam transport, etc.). Each system provides a gateway for communication. The gateways should be accessible only from the central system. For the other systems the central control system defines a uniform interface in order to provide controlled access to the gateways of a connecting system. A locking mechanism of the central system is responsible for system (proxy/gateway) allocation (i.e. granting exclusive access, when a system can be allocated to only one other system at a time). Moreover, it is preconfigured which systems may allocate a certain system for monitoring and control purposes.

* The ELI-ALPS project (GOP-1.1.1-12/B-2012-000, GINOP-2.3.6-15-2015-00001) is supported by the European Union and co-financed by the European Regional Development Fund.

† https://www.eli-alps.hu/?q=en

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

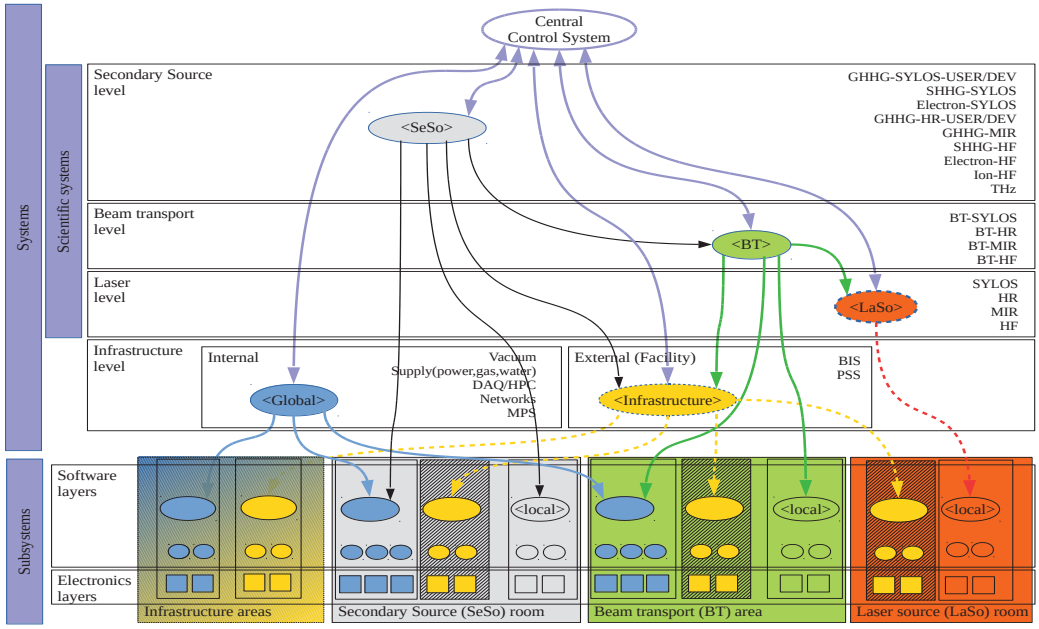


Figure 1: Control System Architecture.

The *Overview GUI* provides a general overview, monitoring all the systems. This GUI can be (semi-)generated based on the Gateways.

Data Acquisition

The data acquisition system will provide a framework for acquiring and processing experimental data originating from all of the experiments, as well as augmenting the data with metadata. Secondary sources and experiments should apply and adopt this framework. Data acquisition will use the common facility level timestamps for both triggering the experiments and timestamping the acquired data (the timing system is to be delivered by an external provider). Already existing solutions and results will be considered [6]. The experiment execution framework should have data processing capabilities at different granularities.

Each experiment will have a unique ID. An experiment consists of a series of **batches**, each batch has a unique batch ID. During a batch, low level data acquisition takes place that may involve low level reduction/ compression and pre-processing done in the low level device and electronic logics layers (e.g. by FPGAs).

After the completion of a batch, **higher level post-processing** can take place, maybe invoking scientific algorithms run on local machines or on a cluster accessible on the network (HPC services). The results of post-processing can influence the execution of the next or subsequent batches, this provides a feedback mechanism during experimentation (online data processing).

SYSTEM DESIGN

The following system design concepts have been identified based on the requirements and constraints given above.

Systems and Subsystems

Briefly, a subsystem incorporates the devices and electronic components (represented by rectangles at the bottom of Figure 1), as well as the subsystem specific software layers: device servers are implemented for physical devices on multiple levels (low level, compound, software logics). These are represented by ellipses in the Figure.

The higher level compound device servers implement gradually higher level functions. These device server components are organized in a layered structure, each layer providing services to the layer(s) above and using services of the layer(s) below.

Systems are composed by combining subsystems. These can be organized into different levels, indicated by boxes in Figure 1. The level name (e.g. Beam Transport Level) is indicated in the top-left corner, while the actual instances (the corresponding systems) are indicated on the right side of the box (e.g. BT-SYLOS): 4 laser sources, 4 beam transport lines and 11 secondary sources are listed.

The Scientific Systems include three levels: the *lasers* will be delivered by external suppliers as turn-key solutions; the *beam transport* systems use the services of beam transport-specific subsystems and the corresponding laser system; the *secondary source* systems use the services of subsystems located in a designated area, and the beam transport system (as well as the central system).

The Infrastructure systems (built on top of localised subsystems) are provided either internally (in this case subsystem device servers are white-box and accessible) or externally (in this case subsystem device servers are black-box and not directly accessible). The various systems comprise an architecture built on top of the TANGO bus. The design makes it possible to control devices located in a designated area of interest (e.g. secondary source room, beam transport area), and also to

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

control, monitor and manage all the devices of the systems supporting the same functionality (e.g. vacuum, power supply, computer and network monitoring, etc.)

Layers

Each system (and its subsystems) has some layers and each layer has to address hardware and software aspects (see Figure 2). The layers from bottom to top are the following.

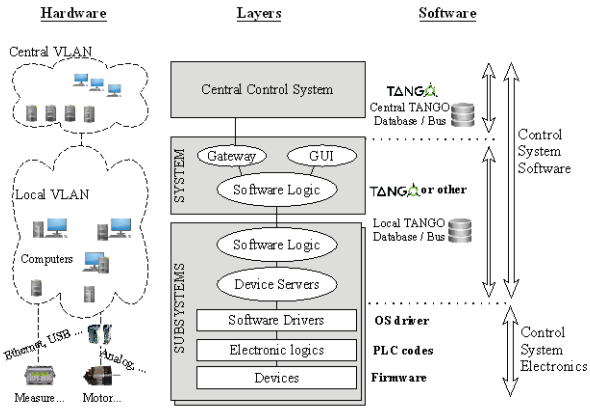


Figure 2: Control System Layers.

Control System Electronics. The *Devices* layer includes the elementary building blocks; it is an electronically controllable device, e.g. a detector or a motor. The *Electronic logics* layer controls the device and provides fast logics through an electronic controller, e.g. a relay or a PLC. The *Software drivers* layer connects the electronic and software worlds: a device or controller can be connected to a local or remote computer controlled by operating system or other drivers (e.g. LIMA, etc.).

Control System Software. The *Device Servers* layer makes the software devices available on the TANGO network and hides all the underlying layers. One computer can host several device servers on the local network, and one device server can provide access to several devices. The *Software Logic* layers provide functionalities on sets of related devices on subsystem and system levels, respectively. These are also implemented as TANGO device servers and are hosted on one or more computers. The *GUI* (Graphical User Interface) layer gives interactive access to the software logics and to the devices for the operators and users. TANGO provides generic GUI clients as well as toolkits to create custom ones. The *Gateway* layer is the access point towards to the central control system. It is implemented also as TANGO device server(s).

It is important to protect the different systems from each other in order to avoid unintentional interactions. A particular system should have its dedicated (virtual) computers, terminals and (Virtual) Local Area Networks; as well as it should have a dedicated TANGO database/bus. Similarly to the systems, the central control system will have its own (V)LAN and TANGO database/bus.

The layer concept can be applied only to the Beam Transport, Secondary Source and Internal Infrastructure systems (those that are not delivered by external suppliers as black-box systems).

PROTOTYPES

Two types of prototypes have been developed in order to validate the design concepts. A vertical prototype includes all layers of a small demonstrational system with physical devices. A horizontal prototype includes the software layers of almost all systems (laser sources, beam transport lines, secondary sources) with software simulated devices.

Vertical Prototype

A simplified optical system was assembled and gave the basis of a vertical prototype (see Figure 3). It was suitable to build up a prototype with TANGO. In the software logic layer there were two loops for stabilizing the manually pre-aligned beam. The GUI (Figure 4) displays these loops and also gives action buttons to the users.

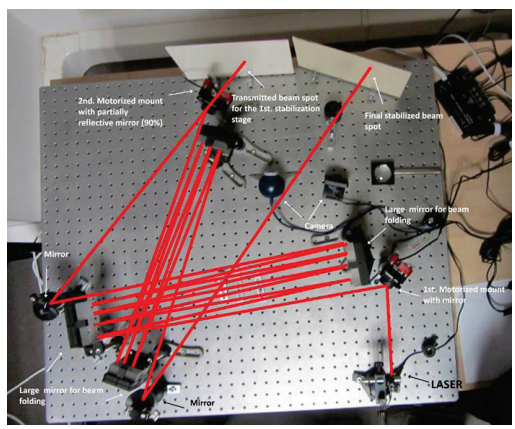


Figure 3: Layout of the vertical prototype.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Horizontal Prototype

This prototype includes a skeleton of the four laser sources and the ten secondary sources: the prototype is based altogether on 700 simulated devices. The prototype provides GUIs for each system, and a central one. The simulation was implemented on the *device servers* layer. This prototype was not generic enough and reusable directly for development and testing, therefore a simulation framework has been elaborated.

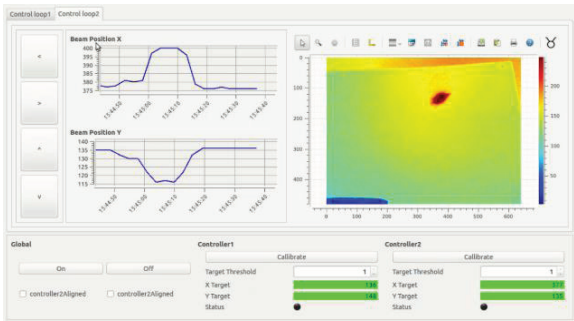


Figure 4: GUI of the vertical prototype.

Simulation Framework

Building on the experiences gained from the constructed vertical and horizontal prototypes, a framework has been designed that aims to cover as many aspects of the ELI-ALPS control system development as possible.

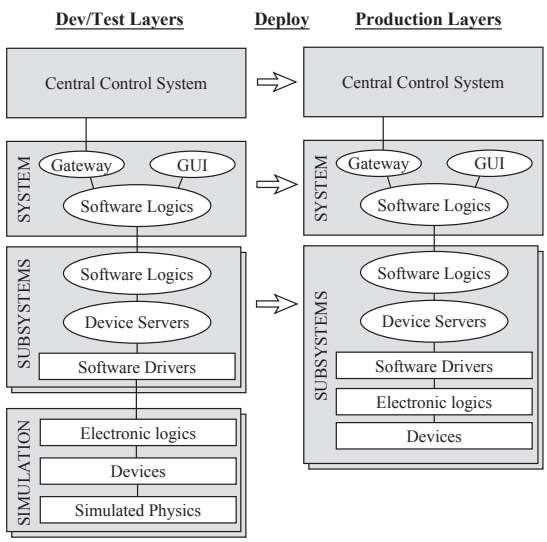


Figure 5: Architecture for development and testing purposes with simulation back-end.

The framework is based on the simulation concept, the system is clearly separated into two parts: the control(ling) environment (left side of Figure 5: top 3 grey boxes) and the controlled environment (left side of Figure 5: bottom grey box). The connection between the controlling and the controlled (simulated) parts can completely be described by a configuration file. This separation provides many advantages because most of the development can be done without the target hardware environment: unit testing, functional and integration testing, deployment test, prototyping, operator training etc. It is important to note that the simulation environment is not expected and does not have to exhibit the same real-time behaviour as the target environment in order to be useful. In fact, different simulation levels can be defined ranging from the provision of the expected interface to gradually higher degrees of realism.

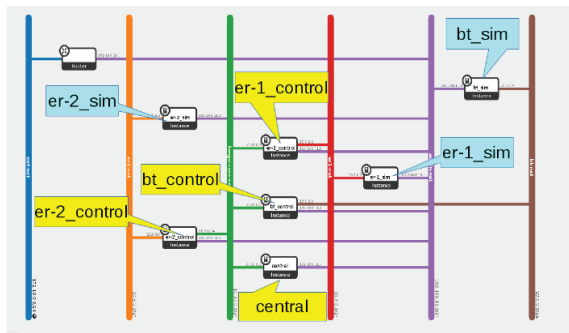


Figure 6: OpenStack environment of the demo system.

A demo system has been developed as a proof of concept for applying the simulation framework that utilizes OpenStack technology as a supporting infrastructure. The demo system consist of a laser source and its beam transport system, two secondary source (experimental) areas and a central control room. The OpenStack environment provides the necessary (virtual) machines for both the simulation (bt_sim, er-1_sim, er-2_sim) and control side (bt_control, er-1_control, er-2_control, central), and an isolated network area per system (Figure 6).

The simulation environment contains components for a sufficiently broad set of equipment: laser source, motors, motor controllers, mirrors, beam dumps, as well as photodiodes and CCD cameras as detectors. The connection to these equipment is described in a configuration file, which is fed into the controlling side to be able to connect the two together. The controlling side provides several functions accessible on GUIs: The central system provides overview status information for the operators. The beam transport system has setup functions as well as can be instructed to set its configuration so that the beam propagates to one of the secondary source areas. In these areas, it is possible to calibrate the local beamline (adjust beam target position and beam diameter, set delay step granularity), as well as start/stop a simplified pump-probe experiment involving an automatic scan through a range of delays between the pump and probe pulses.

CONCLUSIONS

The conceptual system design has been defined according to the requirements. Different kind of prototypes have been developed in order to check the

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

design. There is a lot of work in the future because the requirements of the scientific systems will be continuously changing and evolving in addition to new ones arriving.

REFERENCES

[1] "TANGO Controls Homepage," [Online]. Available: http://www.tango-controls.org/. [Accessed Oct 2015].

[2] M. Pina et al. (2013) "Controlling Cilex-Apollon Laser Beams Alignment and Diagnostics Systems with Tango", in Proceedings of ICALEPCS2013, San Francisco, USA

[3] K. Stouffer et al. (2015) "Guide to Industrial Control System (ICS) Security" U.S.: National Institute of Standards and Technology (NIST)

[4] "Taurus homepage," [Online]. Available: http://www.taurus-scada.org. [Accessed Oct 2015].

[5] S. Rubio-Manrique et al. (2011) "A Bottom-Up Approach to Automatically Configured TANGO Control Systems" in Proceedings of ICALEPCS2011, Grenoble, France

[6] R. Borghes et al. (2013) "A Common Software Framework for FEL Data Acquisition and Experiment Management at FERMI" in Proceedings of ICALEPCS2013, San Francisco, USA.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A FRAMEWORK FOR HARDWARE INTEGRATION IN THE LHCB EXPERIMENT CONTROL SYSTEM

L. Granado Cardoso, C. Gaspar, R.Schwemmer, J. Barbosa, F. Alessio, CERN, Geneva, Switzerland
P-Y. Duval, CCPM, Marseille, France

Abstract

LHCb is one of the 4 experiments at the LHC accelerator at CERN. During the upgrade phase of the experiment (planned for 2018), several new electronic boards and Front End chips that perform the data acquisition for the experiment will be added by the different sub-detectors. These devices will need to be integrated in the Experiment Control System (ECS) that drives LHCb. Typically, they are controlled via a server running on a PC which allows the communication between the hardware and the experiment's SCADA (WinCC OA). A set of tools was developed that provide an easy integration of the control and monitoring of the devices in the ECS. The fwHw is a tool that allows the abstraction of the device's models into the ECS. Using XML files describing the structure and registers of the devices it creates the necessary model of the hardware as a data structure in the SCADA. It allows then the control and monitoring of the defined registers using their name, without the need to know the details of the hardware behind. The fwHw tool also provides the facility of defining and applying recipes - named sets of configurations which can be used to easily configure the hardware according to specific needs.

INTRODUCTION

The LHCb experiment is one of the detectors collecting data at the LHC accelerator at CERN. It is specialized in b-physics and is composed of several sub-detectors and subsystems. All of these subsystems have specialized custom electronic boards and devices to acquire the data from the events produced by the collisions of the LHC beams in the detector. In order to easily and reliably control and configure these devices they need to be integrated in the SCADA System (WinCC OA) [1] so they can be driven by the Experiment Control System (ECS). Due to the custom nature of these devices, their description varies greatly between subsystems, so, to effectively integrate these devices in WinCC OA there was the need to create a tool that could abstract the description of the hardware, model it into the data structures used by WinCC OA and provide a layer to interface with this hardware.

THE FWHW TOOL

All the LHC experiments at CERN use as their SCADA system the software WinCC OA. In order to reduce duplication of development, the JCOP (Joint Controls Project) [2] was created in order to provide common controls solutions for the four experiments. JCOP provides a framework for the creation of JCOP components – WinCC OA packages containing all the required user panels, libraries, scripts and other software – that can be easily installed and distributed.

The framework Hardware tool (fwHw) [3] is developed as a JCOP component, allowing for easy installation by the sub-detectors. The tool provides a user interface, which allows for the easy modelling and configuration of the existing hardware devices and can also serve as a debugging tool. However, its purpose is mostly setting up the system and the different sub-detectors or subsystems should then implement their hierarchical control Finite State Machine (FSM) trees and their own user interfaces more suitably geared to the operation of their electronics.

Hardware Abstraction

The fwHw tool models the hardware into WinCC OA datapoint structures. The WinCC OA data structure is a tree-like structure, where a logical model is defined in Datapoint Types, which can be instantiated in datapoints that share this logical model.

Similarly the fwHw tool produces the logical model of the electronic devices as Datapoint Types, but providing an interface more adapted to the modelling of electronic devices.

In the fwHw tool, the data is hierarchical organized with the following types:

- Registers.

The registers are the representation of a register on a real hardware device. A register can be of any size and implement any type of protocol. The hardware tool allows the configuration of each register according to the settings necessary to access the hardware via the different protocols, the size of the register and the type of readout it will require.

The registers can also represent a local variable, i.e. a logical data item of a given area (e.g. a string which holds a filename to configure a given area).

These are the leaves or the lower level nodes of a hardware device model tree.

- Areas/Sub-Areas

Areas are logical groups of registers and/or other areas, which represent a logical division of a part of the hardware device (e.g. a chip or a group of chips). Defined areas can be declared as sub-areas of other areas and can be used multiple times. For example, in Figure 1 we can see that both areas 2 and 3 have the same type and thus have the same structure. The defined areas can also be used by different Device Types, so if some devices share a common structure, there is no need to duplicate similar structures.

- Device Types

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The topmost node of the description of the hardware is the Device Type. The device type is the definition of a device (e.g. a board) with the areas and registers that compose it and the definition of the interface to the hardware they will have.

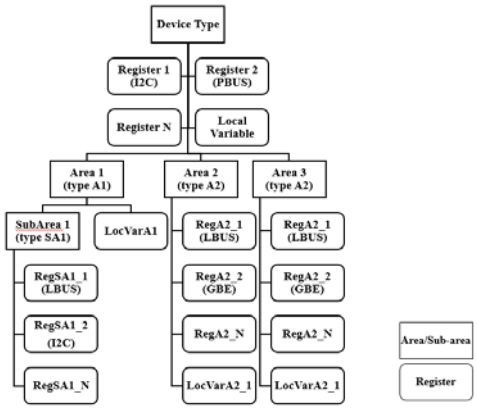


Figure 1: Example of a model of a device.

After the model of the hardware device is defined, we can create as many instances of that model as required and configure the particular settings for those devices.

The created models hide the complexity of the communication with the hardware. Settings like the specific protocol, addresses, sub-addresses and all others necessary to access the hardware registers are included in the model and this allows the access of the hardware registers simply by using the name defined in the model.

Hardware Model Creation

There are three ways to create the model of an electronics device using the fwHw tool:

- via the User Interface Panels:

The fwFw tool provides a user interface, which can be used to create the models of the electronics devices to be configured. It is intuitive and easy to use, however it can be cumbersome to create a device model with many registers/sub-areas, as the registers need to be created one by one and the sub-areas will also need to be included one by one into other areas. Another disadvantage is that it is not particularly easy to update a given model if the changes required are somewhat extensive.

- via scripts:

The tool also provides the functionality of using a script to create the hardware models; this improves the creation of big hardware models as well as improves the update or modifications to those models. It can also be used to automate the creation of devices quite easily. It requires writing of scripts using the provided library functions, which requires a learning curve and the scripts may be quite long.

- via XML description files:

There is also the possibility of creating the hardware models by describing them in XML files. These XML files will then be used to generate the scripts, which will create the hardware models in the fwHw tool. The usage of an XML hardware description file has the advantage of being easily readable, being validated against an XML schema file and being easily changed or updated. It is also foreseen in the future to have these XML files generated automatically from other descriptions (e.g. an FPGA firmware file).

The implementation of the configuration via XML description files allows also for the easy export of the currently defined models of hardware in a project to be adapted by different sub-detectors or sub-systems.

Hardware Interface

The electronics devices in LHCb are controlled using specific interfaces. Currently two interfaces are used according to the location of the devices to be controlled:

- SPECS [4] – Serial Protocol for the Experiment Control System – used in radiation areas;
- CCPC [5] – Credit Card PC – a small embedded PC running Linux – used in non-radiation area;

A third interface will be used in the future:

- GBT [6] – Giga-Bit Transceiver – a radiation hard chipset – used in the future upgrade of LHCb

The type of registers available depends on the type of interface. A summary of the available register types is shown in table 1.

Table 1: Types of Registers per Interface

CCPC	SPECS	GBT
I^2C	I^2C	I^2C
JTAG	JTAG	JTAG
LBUS	PBUS	Memory Bus
GPIO	REG	PIA
GBE	DCU	SPI
-	-	ADC
-	-	DAC

These interfaces provide the connection between the ECS and the hardware devices. They run a server that communicates with the fwHw tool; this server is based on DIM (Distributed Information Management system) [7], which can easily be interfaced from WinCC OA.

DIM is a communication system for distributed / mixed environments and it provides a network transparent inter-process communication layer. It is based on the server/client paradigm and it uses the concept of publishing/subscribing services and commands.

The way these interfaces are integrated into WinCC OA require also the installation of specific components for each interface, which provide the communication between the fwHw tool and the DIM server running on the interface PC. These components provide the lower-level functions to access the different types of registers, which are used by the fwHw tool. They are appropriately named

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

fwSpecs, fwCcpc and fwGBT for the SPECS, CCPC and GBT interfaces respectively.

After an electronic device is modelled in the fwHw tool, devices of this type can be instantiated. Specific settings need then to be set, in order to configure the interface (or more exactly the DIM server) with which the device will connect.

Once the devices are instantiated in the fwHw tool it is needed to make available to the DIM server the names of the declared registers of the model. The configuration is automatically passed to the DIM server at startup, and the server is then aware of all the existing registers of the hardware as well as all the settings necessary to access them, their size and the desired readout mode.

In Figure 2 we can see the fwHw tool user interface, showing declared hardware types on the left and the registers of that hardware type on the right.

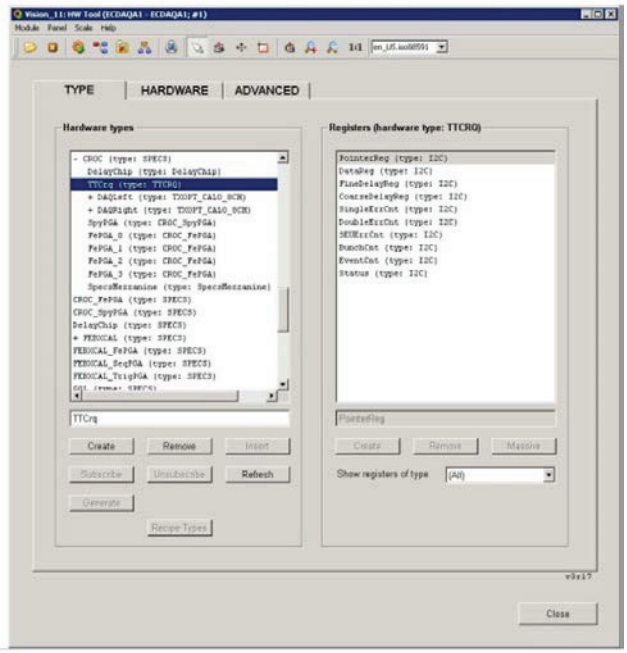


Figure 2: fwHw User Interface showing an existing model and the named registers for one device of this type.

As mentioned earlier, in the upgrade of LHCb, the new interface GBT will be used to interface the hardware both in radiation and non-radiation areas (it will link to both back-end and front-end devices). The GBT is a common CERN Serializer-Deserializer radiation-hard chipset for data, fast and slow control distribution to and from the electronics for the upgraded LHC experiments. This new interface requires the development of the new component fwGBT, which contains the lower level communication interface and the GBT DIM server.

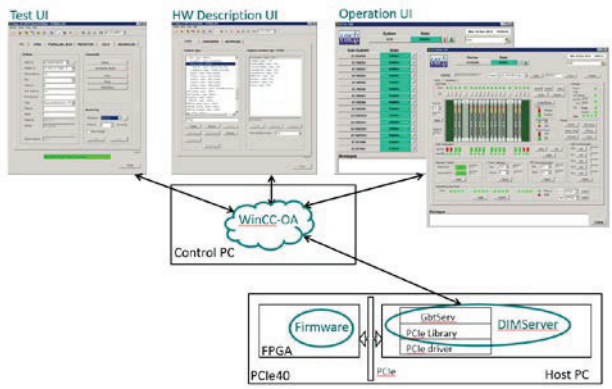


Figure 3: Architecture of the framework solution for the GBT Interface.

In Figure 3, we can see the schema of the architecture of the whole solution for the GBT Interface (the SPECS and CCPC interfaces share the same architecture), with a control PC running WinCC OA, which has the framework components installed (fwHw and fwGbt) and where the hardware is modelled and available for the FSM Control Tree and User Interfaces. The control PC is connected to the host PC where the DIM Server is running via the network. The DIM Server is aware of the modelled devices, the necessary settings to access them, and the required readout modes (poll, update on change, etc). It integrates the lower level libraries to communicate with the devices via the several available protocols.

Configuration DB and Recipes

One of the components provided by JCOP is the fwConfigurationDB [8]. This component provides a way to store and apply different sets of data to WinCC OA datapoints. It introduces the concept of recipes - named configurations for a given device or set of datapoints. These configurations can be both static configurations in order to setup equipment (e.g. configuring a spare device that is replacing one that failed) as well as dynamic configurations (e.g. configure the devices for different LHC modes of operation).

The fwHw Tool provides an interface for the configuration of recipes for the defined hardware models and this allows for the easy configuration of the devices according to specific needs. For instance it is possible to set specific values to the registers of the devices when the LHC provides physics conditions for data taking.

The created recipes can be both cached locally in the systems where they will be used and also stored on an Oracle database.

CONCLUSION

The fwHw tool provides an easy and reliable solution to integrate custom electronics devices into the LHCb Experiment Control System. Even though the existing electronics devices in LHCb are of varied types, this tool provides a way to easily abstract and create models of these devices and control them from the global

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Experiment Control System. It is also easy to modify the already existing models without major impact to the existing system.

The tool is able to facilitate the integration of the sub-detectors electronic devices as it provides a base for the sub-detectors to integrate their devices into their ECS control trees and develop their specific user interfaces (Figure 4), communicating with the hardware using the functions provided by the fwHw tool libraries.

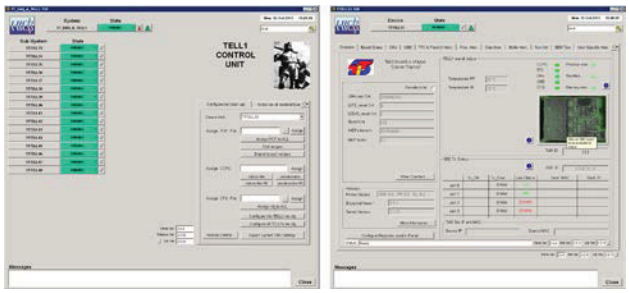


Figure 4: Control Tree and device panel for one of the LHCb sub-detectors.

This tool also makes possible to easily control the devices by providing a way to interface the registers of these devices using named registers. This hides the complexity of the communication from the user and enables the possibility of using the configuration DB and recipes to configure the electronics according to the specific needs of the experiment in different operation modes. It also provides a base for increasing the automation of the running of the experiment [9].

In the future upgrade of the LHCb, new interfaces will appear to replace or work along the existing ones. Due to the generic nature of the fwHw tool, these new interfaces can be easily integrated.

REFERENCES

[1] SIMATIC WinCC Open Architecture made by ETM Professional Control GmbH, Eisenstadt, Austria: http://www.etm.at

[2] O. Holme et al., "The JCOP Framework", ICALEPCS 2005, Geneva, Switzerland.

[3] fwHw Tool: http://lhcb-online.web.cern.ch/lhcb-online/ecs/FWHW/default.html

[4] SPECS website: https://lhcb.lal.in2p3.fr/Specs/

[5] CCPC website: https://lhcb-online.web.cern.ch/lhcb-online/ecs/ccpc/default.htm

[6] F. Alessio and R. Jacobsson, "A new readout control system for the LHCb upgrade at CERN", JInst Topical workshop on electronics for particle phusics, 2012, Oxford, United Kingdom.

[7] C. Gaspar, "DIM - A Distributed Information Management System for the Delphi experiment at CERN", IEEE Real Time Conference, 1993, Vancouver, Canada.

[8] JCOP Framework Configuration DB website: https://wikis.web.cern.ch/wikis/display/EN/JCOP+Framework+Configuration+Database

[9] C. Gaspar and B. Franek, "Tools for the automation of large distributed control systems", IEEE Transactions on Nuclear Science., Vol. 53, NO. 3, June 2006.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SYNCHRONISING HIGH-SPEED TRIGGERED IMAGE AND META DATA ACQUISITION FOR BEAMLINES

N. De Maio, A. P. Bark, T. M. Cobb, J. A. Thompson
Diamond Light Source Ltd., Didcot OX11 0DE, UK

Abstract

High-speed image acquisition is becoming more and more common on beamlines. As experiments increase in complexity, the need to record parameters related to the environment at the same time increases with them. As a result, conventional systems for combining experimental meta data and images often struggle to deliver at a speed and precision that would be desirable for the experiment. We describe an integrated solution that addresses those needs, overcoming the performance limitations of PV monitoring by combining hardware triggering of an ADC card, coordination of signals in a Zebra box [1] and three instances of areaDetector streaming to HDF5 data. This solution is expected to be appropriate for frame rates ranging from 30Hz to 1000Hz, with the limiting factor being the maximum speed of the camera. Conceptually, the individual data streams are arranged in pipelines controlled by a master Zebra box, expecting start/stop signals on one end and producing the data collections at the other. This design ensures efficiency on the acquisition side while allowing easy interaction with higher-level applications on the other.

INTRODUCTION

The kinds of data rates coming out of a typical modern synchrotron beamline continue to push up against the current limits in experimental complexity, processing power and storage space. Ongoing improvements in automation and computing infrastructure help scientists get past those limits, decreasing the time individual scans of a sample take to set up and carry out.

When it comes to taking data, a number of well-established technologies and beamline designs provide stable frameworks to acquire as much as possible across a number of different devices and channels. Each of these comes with its own requirements with regards to configuration and operation. Each of these may or may not also provide its own native mechanisms to sample data at user-defined regular intervals.

Getting these devices to perform their measurements in a synchronised way poses some unique challenges. If the samples are taken a sufficient time apart, the middleware layer provided by the controls software can, and often does, take on the role of coordinator. But there is a point where the interval between two samples gets too small to guarantee accurate time stamps. Different devices may also expect different kinds of triggers, and every additional layer of software mediating between those makes the delays between the data acquisition system and the hardware more unpredictable. Figure 1 illustrates this problem.

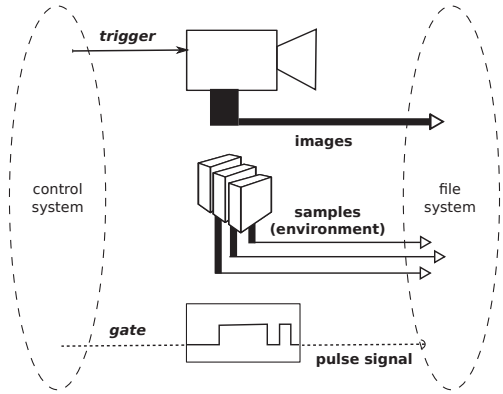


Figure 1: Unsynchronised trigger and data flow during image acquisition.

ARCHITECTURE

The answer to coordinated sample acquisition at small time scales is to use hardware triggering. Using a Zebra box [1], we present a design for a fast data acquisition system intended for sampling rates within the range of 30Hz and 1kHz. It not only coordinates the different hardware triggers involved but also bundles the acquired samples in three parallel pipelines, resulting in three files containing measurements for the same time frame.

Figure 2 gives an overview of the architecture. As Diamond Light Source (DLS) uses EPICS [2] as its control system, we rely on the areaDetector framework [3] and the EPICS implementation of the Zebra driver for configuring and collecting data from the underlying devices. We also rely on areaDetector plugins to post-process this data to some extent and to redirect it to files.

The control signal flow in our model is driven by the Zebra box. In our beamline-specific application, it receives an external trigger signal of its own. That signal causes the Zebra to produce one or more outgoing pulses, which are then forwarded to the connected devices. The incoming pulse need not be another hardware signal. Arm or Acquire commands to the relevant PVs are just as suitable because it is the configuration of the FPGA logic in the Zebra that ultimately decides what kinds of pulses to send out in response, and how many.

If individual devices expect different pulse shapes as trigger signals, the Zebra box acts as a mediator between incoming and outgoing triggers while ensuring that all outgoing pulses are sent off at the same time. If they are capable of receiving external triggers, they should be set to operate in this way. If they are not, the trigger signal needs to be recorded on one of the data channels so it can be extracted

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

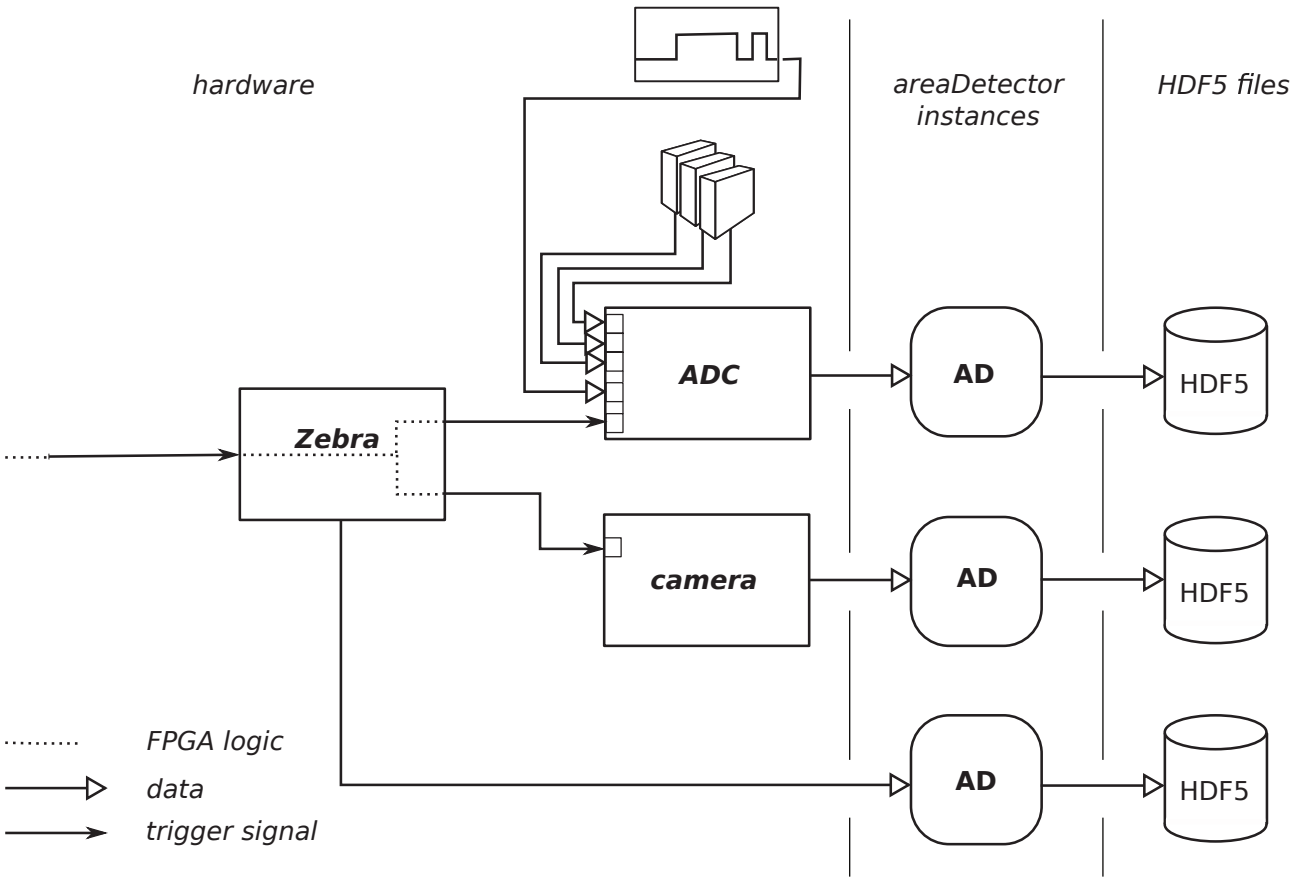


Figure 2: Zebra-driven data acquisition architecture.

later. We describe an EPICS-based way of doing that in our beamline-specific application in the next section.

The number of devices controlled in this way is only limited by the number of outgoing connectors on the Zebra box. So it is completely possible to trigger more than one type of detector or ADC card at once - or any other device that meets the requirements outlined in the previous paragraph.

Each of the devices involved has an instance of areaDetector associated with it. They ensure a consistent interface to EPICS and enable the architecture to write generic frames to, typically, HDF5 files. They also provide a well-defined set of parameters to interact with when configuring a specific scan or experimental run.

It is also worth pointing out that the entire design is is self-contained within the EPICS control system. So once the three pipelines are set up and configured, all a larger application needs to do is send start and stop commands to one end and pick up the data files at the other.

IMPLEMENTATION

Besides the areaDetector framework, the EPICS control system also provides drivers for a number of different cameras. These typically derive from areaDetector and use it in the conventional way, to acquire a series of image frames. Although an essential part of our data acquisition system, we did not need to extend any of them to make them work within it. So they will not be discussed further here.

The Zebra box and the ADC card, on the other hand, do not produce image frames but time series of samples across a set of channels. So the interpretation of a frame changes: it no longer represents an image in the traditional sense but a kind of ribbon of sampled values. Because we group several channels together, we continue to acquire two-dimensional data. As such, the areaDetector framework remains a suitable abstraction layer for this kind of measurement. The width of such a frame is fixed and corresponds to the number of available input channels. By contrast, the height has to be set by the user. As we explain below, approaches on what to set it to vary depending on the device.

The hardware triggering presented another, unrelated set of problems. While external trigger signals are a native feature of the Zebra box, our choice of ADC card does not really have this feature in a way that is useful to us. So both devices needed different kinds of drivers and driver extensions, which needed to be in line with the requirement that they both fit into the areaDetector framework. But on top of that, our ADC cards need to have the trigger signal fed in as data so one of the areaDetector plugins can extract it later. We will discuss each of those aspects in turn.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

areaDetector Driver for Zebra

The original Zebra driver, developed at DLS, derives from class asynPortDriver and uses a serial port to communicate with the hardware. It also stores samples from ten different channels (encoder inputs, results of some of the logic blocks) in individual waveforms whenever a trigger is issued. So the time series for all of the channels are already available but there is no time stamp to go with them, and no native way of writing them to file.

Changing the inheritance chain of the Zebra to ADDriver turns it into an areaDetector. As ADDriver itself derives from asynPortDriver, no existing functionality is lost. Instead, the new parent gives the Zebra driver access to all of the areaDetector data structures. Figure 3 shows the details of the old and new parent class relations.

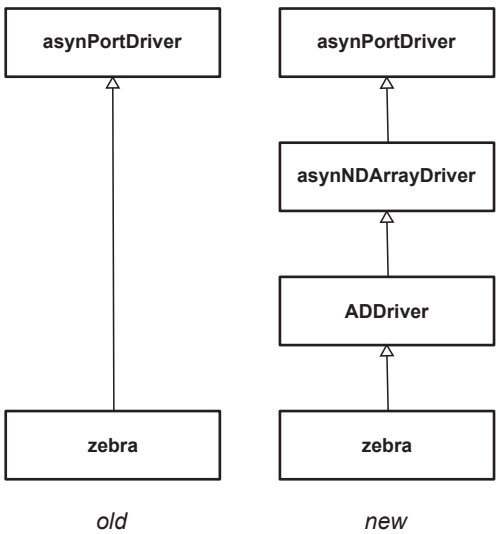


Figure 3: Old Vs. new inheritance tree for a Zebra driver in EPICS.

Every time the ten channels are sampled for the waveforms, we now store the same values as an NDArray in an areaDetector frame. A time stamp is added as the eleventh column. The Zebra's native interpretation of the Arm and Disarm commands as representing the beginning and end of a scan makes them equivalent to the start and stop points of an areaDetector acquisition. We extended the driver to treat them equally as a result. So while we acquire a stack of N frames from the camera during a scan of N points, we also get a stack of N one-line frames from the Zebra.

areaDetector Driver for D-tAcq ADC

DLS developed the original version of the areaDetector driver for the D-tAcq [4] ADC card. It connects to TCP port 4210 to read a continuous stream of data from the ADC, and to TCP port 4220 to send control commands to it. As the ADC samples its channels at a fixed rate, the driver's main task is to collect the samples reliably and store them as lines in a frame.

The sampling rate of the D-tAcq ADC far exceeds the frame rates we expect to see during our scans. Since we have no way of turning the data stream on and off, the driver's main focus is on capturing everything in sensible chunks for further combined processing in the downstream plugins. One of those chunks corresponds to a frame of raw data. This turns the user-defined frame height into more of a performance parameter: because everything runs in continuous mode, the length of a scan is defined in the file writer plugin instead.

The fact that there is no way to connect a native external trigger signal to the device was one of the most difficult conceptual challenges to overcome. In the end, we achieved it not within the device driver itself but by using a specialised areaDetector plugin. Using the trigger signal recorded on one of the data channels, it turns the massively over-sampled raw data into a series of one-line frames after the fact.

Using areaDetector's Reframing Plugin

The key to extracting the ADC samples from the raw data stream at the appropriate rate lies in the reframing plugin for areaDetector developed at DLS. It needs to be told which channel the triggers were recorded on in the raw data frame, and what the threshold value for a trigger pulse was for the application at hand. It then uses this information to downsample the incoming data by picking out those lines that have a trigger pulse in them. Each time a trigger is found, it reads out a user-defined number of lines - one in our case - into a new frame, creating a second data stream. It is this post-processed stream that is handed over to the downstream plugins. Figure 4 explains the exact relation between the two streams.

Every time a raw frame reaches the reframing plugin, there is a burst of re-sampled frames going to the downstream plugins, followed by a pause until the next raw frame arrives. So there are some practical considerations that need to be observed when setting up the core driver and plugins. As mentioned, the frame height in the raw data stream influences performance. During our tests, it became apparent that a size that corresponds to about one frame per second strikes the best balance between filling up the queue for the reframing plugin and overwhelming those of the downstream plugins during bursts. Even so, the queue sizes need to be far larger than for other areaDetector applications. A value around 1000 lets them cope with bursts every second or so with a comfortable margin.

The reframing plugin also expects a threshold voltage to be able to identify trigger pulses from the raw data frames. Since the pulse comes from a Zebra and we trigger on a rising edge, a value of 3V has turned out to be appropriate for this purpose in the early tests.

INTEGRATION

While it is perfectly possible to use the pipeline manually, its real power comes from controlling the endpoints using either Python scripting or a data acquisition package - GDA [5] in our case.

As long as the high-level application knows about PVs, it

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

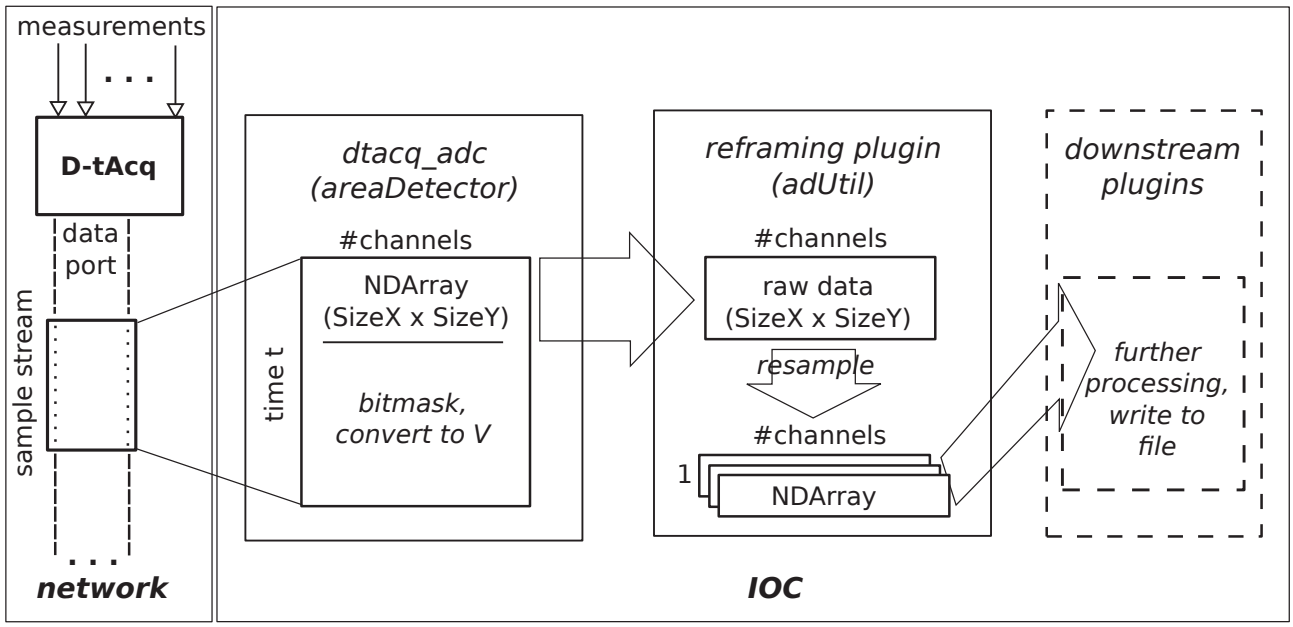


Figure 4: Data flow in the areaDetector plugin chain. Unusually, the raw data stream is discarded after the downsampling step in the reframing plugin and a stream of new frames of a different size is passed on to the downstream plugins.

can configure and monitor the lower-level devices using caput, caget and camonitor. It can also aggregate the results into NeXus files that provide a single entry point into the collected data from the three pipelines. Because it knows about the rest of the experimental setup, it can start and stop acquisitions and file captures at the appropriate times.

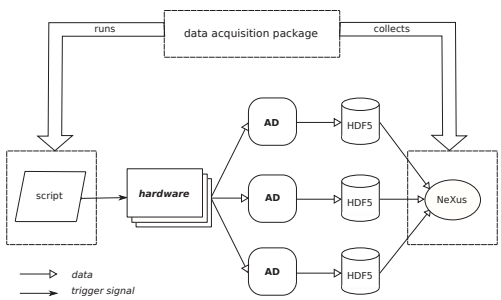


Figure 5: Role of high-level data acquisition package.

Figure 5 shows how GDA or a similar application brackets the lower-level components during a scan, acting in a kind of management role. This approach turns the entire integrated data acquisition system into a single building block within the beamline infrastructure.

CONCLUSION

Using hardware-based trigger signals in combination with the Zebra box's programmable FPGA logic ensures exact timing and synchronised acquisition of data and meta data at higher sampling rates. EPICS drivers derived from areaDetector provide a well-defined interface to configure the system as a whole, all the way down to its individual components. Structuring the controls and data flows in three parallel pipelines allows us to integrate such a system into the rest of the beamline software infrastructure in a modular way. While there are some performance parameters to keep in mind when interacting with the 'pipeline-within-a-pipeline' chain of areaDetector plugins, using them is essential and ultimately adds to the overall flexibility when choosing individual hardware components. Fine-tuning and testing of the complete system is ongoing and we expect to deploy it to one of our high-throughput tomography beamlines at some point next year.

ACKNOWLEDGEMENT

The authors would like to acknowledge that the original Zebra driver was developed by Tom Cobb, DLS, the D-tAcq driver by Adam Bark, DLS, and the areaDetector reframing plugin by Edmund Warrick, DLS.

REFERENCES

[1] T.M. Cobb, Y.S. Chernousko, I.S. Uzun, "ZEBRA: A Flexible Solution for Controlling Scanning Experiments", Proceedings of ICALEPCS 2013.

[2] EPICS collaboration website: `http://www.aps.anl.gov/epics/`

[3] areaDetector documentation: `http://cars9.uchicago.edu/software/epics/areaDetector.html`

[4] D-tAcq website: `http://d-tacq.com`

[5] GDA website: `http://www.opengda.org`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

QUICK EXPERIMENT AUTOMATION MADE POSSIBLE USING FPGA IN LNLS

M. P. Donadio*, H. D. Almeida, J. R. Piton
CNPEM / LNLS, Caixa Postal 6192, Campinas - 13083-970, Brazil

Abstract

Beamlines at LNLS are being modernized to use the synchrotron light as efficiently as possible. As the photon flux increases, experiment speed constraints become more visible to the user. Experiment control has been done by ordinary computers, under a conventional operating system, running high-level software written in most common programming languages. This architecture presents some time issues as computer is subject to interruptions from input devices like mouse, keyboard or network. The programs quickly became the bottleneck of the experiment. To improve experiment control and automation speed, we transferred software algorithms to a FPGA device. FPGAs are semiconductor devices based around a matrix of logic blocks reconfigurable by software. The results of using a NI Compact RIO device with FPGA programmed through LabVIEW for adopting this technology and future improvements are briefly shown in this paper.

INTRODUCTION

Experiments at LNLS historically were made using software running in conventional operating systems. Our first software for beamline control centralized all the operations, sensor reading and actuators controlling in a single thread. This was the cause of lots of disturbances in motor movement, when the user moved a mouse or selected a menu item, which almost never ran in a smooth pattern. We could see also a big dead time caused by many tasks running sequentially rather than in parallel.

In 2010 we started to use EPICS [1] and, this way, we could decentralize all the access to the devices. Doing so, we could get many improvements in motor controlling and reduction of dead time. The system became easier to maintain as each device is accessed by a single piece of stan-dalone software. Another advantage was to provide the infrastructure to build a web system named LabWeb [2], based on Science Studio [3] developed and applied at the Canadian Light Source (CLS), that became possible for users to operate the beamline from their universities.

From 2010 until recently in the current year not only the software, but most of the control hardware was replaced in all beamlines. After some years developing, improving and using the new EPICS based system, we faced a new challenge: the beamlines had its photon fluxes and detector sensitiveness increased by the hardware improvement and software was again the bottleneck for some experiments. EPICS can provide a good experiment control using only software and not firmware, in conventional operating systems, if there is no concern about times lesser than 100 ms. Control loops quicker than 10 Hz are not guaranteed to be attended inside a good error limit. To go beyond this limit we needed to change the paradigm adopting a real-time system, for example, or implementing the fast piece of the experiment in hardware.

The first beamline to see this limitation was IMX [4], followed by XRF [5]. Both needed to synchronize motor position, detector shot and shutter in a few milliseconds with error margin of some microseconds.

The third beamline case was SAXS1 [6], whose system is presented in this paper. The development of the system for SAXS1 was used also to test the approach to the beamlines that will be built in Sirius [7], the new Brazilian Synchrotron. The timing for Sirius beamlines needs to be in the microsecond loop or faster.

SAXS1 HARDWARE

We need a system with low jitter between the trigger signal and the start and the end of data acquisition. Going near the hardware is an approach to reduce jitter and that is the motivation to choose FPGA technology. We had available in our inventory a cRIO NI 9144 [8], a unit with FPGA, so this was a natural choice to use in the work. cRIO NI 9144 can be connected to a PXI [9] only by an EtherCAT connection.

Figure 1 shows the connection between the cRIO and detectors and actuators.

There are two main detectors, a Pilatus [10] 300K to measure SAXS and a Pilatus 100K to measure WAXS. Pilatus 300K is configured to collect data in a determined frequency and sends a trigger TTL signal every time an acquisition is being done. cRIO is responsible to send a TTL trigger to Pilatus 100K as it can start acquisition.

Two photo-diodes provide the beam intensity data through a Stanford SR570 [11]. One of the photo-diodes is located in the beam stopper and receives radiation after the sample. The other one is located before the sample and receives the reflection of a small part of the beam thanks to a mylar or kapton thin film positioned in a 45-degree angle related to the beam. cRIO reads the voltage data from Stanford SR570 using a NI 9215 module [12].

cRIO also controls a fast shutter to limit to a minimum the time the sample is irradiated, by opening the shutter only when Pilatus is acquiring.

A cRIO is used to:

- run the FPGA code that sequences the experiment
- read voltage from Stanford using a NI 9215 module

* marcio.donadio@lnls.br

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

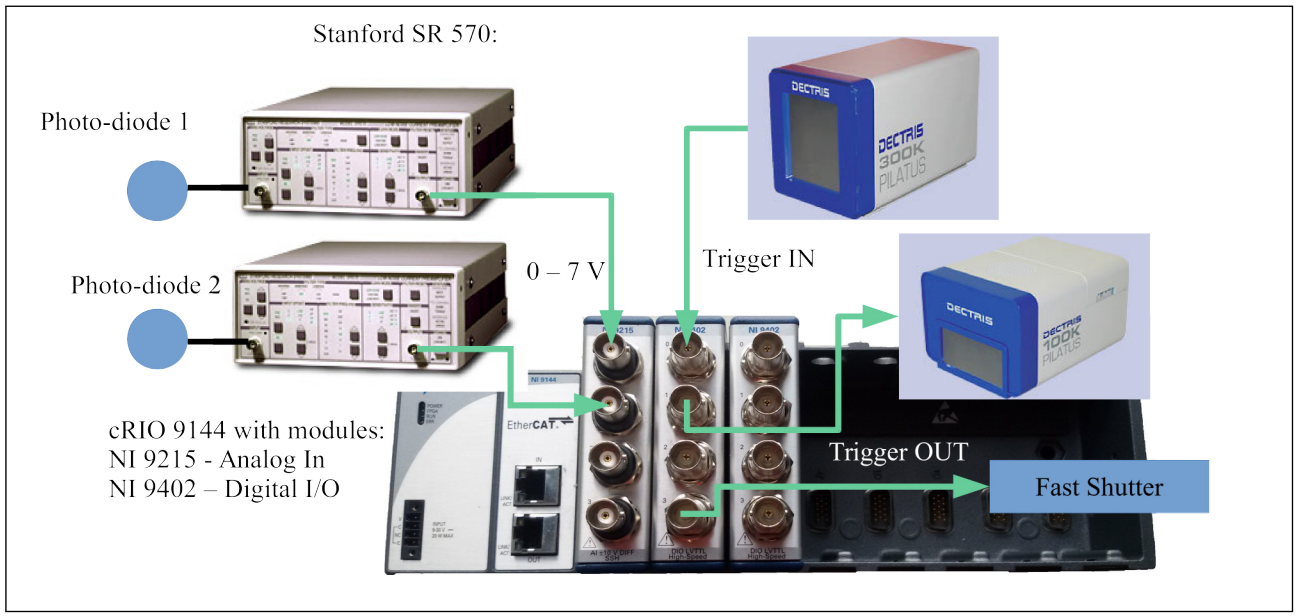


Figure 1: SAXS1 Hardware Connections.

- read trigger signals from Pilatus and to write trigger signals also to Pilatus and to the fast shutter using a NI 9402 module [13].

Figure 2 shows the connection between cRIO, PXI and EPICS clients.

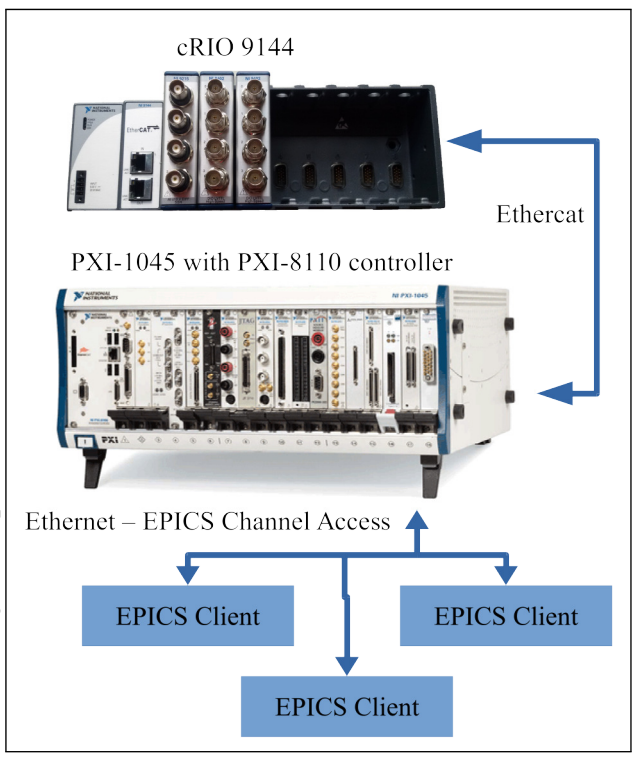


Figure 2: SAXS1 cRIO - PXI connection.

Data collected in cRIO is transferred to the PXI through an EtherCAT connection to be used by a software running in LabVIEW RT operational system inside the PXI controller. Data from LabVIEW RT is passed to the Linux running in the same PXI by Hyppie [14, 15]. In Linux we have an EPICS IOC that sends the collected data to client software by using the beamline network.

FPGA IN cRIO

FPGA in cRIO is programmed using LabVIEW [16]. FPGA code is responsible to control the experiment sequence: receive trigger signal → open shutter → collect voltage from Stanford until trigger signal is low → close shutter → calculate voltage average during the time the trigger signal was high → save the data.

Figure 3 shows the loop where data is collected from the analog input module and summed over the time. When the trigger signal goes TTL low, a division of the sum by the number of loop iterations is made to calculate the voltage average while the detector was acquiring. Not shown in this picture, this average is saved in a FIFO for later use. Let's call this FIFO as FIFO_PDn.

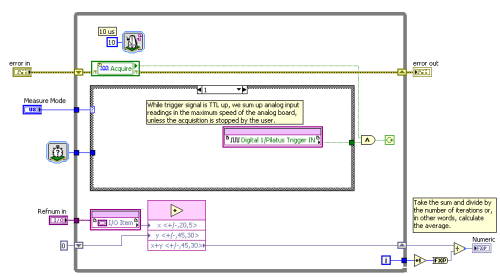


Figure 3: FPGA Voltage Acquisition.

The code is also responsible to administrate data sending over EtherCAT. This is the part of the FPGA code in the deterministic domain. Figure 4 shows a loop where cRIO waits for the rising edge of Scan Engine [17] clock. When

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

it arrives, an inner loop gets all data from FIFO_PDn and writes it to a sequence of 50 shared variables per photo-diode, emulating an array. As the Scan Engine speed of cRIO is set to 500 μs in this experiment and 100 data values are sent per EtherCAT packet, we guarantee that the software is able to collect one analog data per photo-diode each 10 μs and send it to the PXI with no data loss. Each data uses fixed point representation with 5 bits for integer part and 20 bits for decimal part. No handshake is implemented because transmission of data over EtherCAT is deterministic and lossless. In fact, we ran millions of tests at the maximum rate and never got a single data loss. Nevertheless it is possible to diagnose when a data is lost and, in this case, the experiment is aborted.

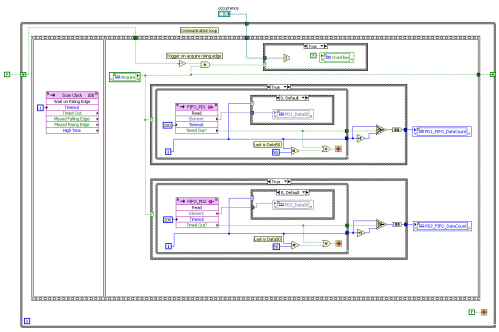


Figure 4: FPGA sends data to LabVIEW RT.

SOFTWARE IN LabVIEW RT

Inside LabVIEW RT there is a LabVIEW software responsible to get information from cRIO and to send it to the shared-memory between LabVIEW RT and Linux. The program collects data using a real-time loop synchronized with the scan engine period (Fig. 5). This is the part of the LabVIEW RT code in the deterministic domain. Doing so it is guaranteed that no data will be lost, as long as the loops fit within the scan engine cycle period, because the EtherCAT cycle is synchronized to the scan engine cycle. In the case of communication failure, experiment is aborted.

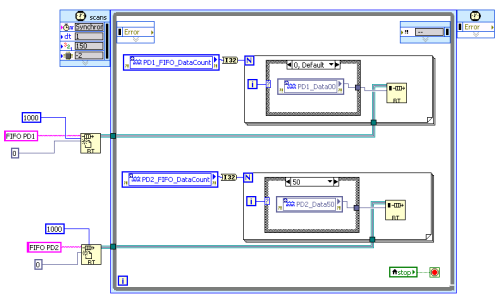


Figure 5: LabVIEW RT software gets data from FPGA.

The collected data is written in an array and is sent to the shared-memory asynchronously. The array is passed to the Linux IOC using Hyppie. The Hyppie shared-memory structure used is described in Table 1 and 2. The tables show the description of each memory address and points which software is responsible to write in it: software running in LabVIEW RT or the IOC running in Linux.

Table 1: Shared-memory Structure - Part 1

Addr	Description	Written by
0	Command	RT / Linux
1	Status message (0 = stopped, 1 = running)	RT
2	Last error found	RT
3	Current measurement point	RT
4 - 13	Current amount of data read from cRIO Diode 1 - 10	RT
14	Delay time before each frame acquisition (in ms)	Linux
15	Delay time after each frame acquisition (in ms)	Linux
16	Number of points to measure (maximum 1000)	Linux
17	Trigger from? (1 = Pilatus, 2 = environment)	Linux
18	Close Shutter	Linux

Table 2: Shared-memory Structure - Part 2

Addr	Description	Written by
19 - 28	Type photo-diode 1 - 10 (1=Stanford,2=Keithley)	Linux
29	Dark current time(in ms)	Linux
30 - 39	Dark current value from photo-diode 1 - 10	RT
40-1039	Measurement array photo-diode 1	RT
1040-2039	Measurement array photo-diode 2	RT

SOFTWARE IN LINUX

In Linux we developed some device supports [18] that read data from the shared-memory, using Hyppie to access memory addresses described in Tables 1 and 2. This IOC provides PVs [19] to configure, start, stop and read parameters from the experiment sequencer programmed in cRIO's FPGA. To access these PVs we are using Py4Syn [20] and CS-Studio [21] as EPICS clients.

RESULTS AND DISCUSSION

Figure 6 shows on the top the trigger signal that is sent to the system, with 1.25 μs high and 20 μs low, resulting in

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

a total period of 21.25 μs. On the bottom we can see the signal that indicates when FPGA stopped to read the voltage signal and detects that the trigger signal was low. We could run all the algorithm in FPGA cyclically with a worst case period of 21.25 μs (see the red square in Fig. 6). In the best case we could get 13.6 μs, as seeing in the first trigger pulse shown in Fig. 6. The acquisition time of NI 9215 module is approximately 10 μs and we read it inside a loop. The time difference between the best and the worst case happens when the trigger signal goes low in some point of the start of this loop. So FPGA runs another complete acquisition time before detecting that the measurement needs to be stopped.

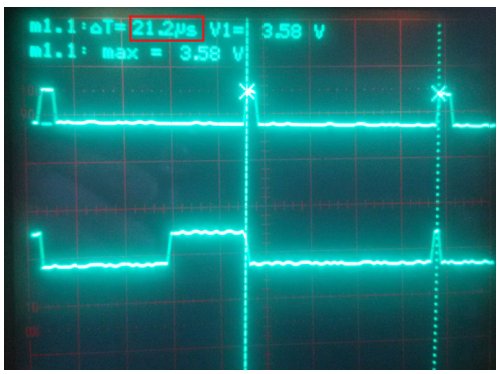


Figure 6: Time to detect trigger unset signal.

CONCLUSION

This entire system is used in simple SAXS and WAXS acquisitions, using or not chemical kinetics or any other complex experiment using synchronization of the system with the sample, as temperature and tensioning forces acquisitions with good stability. The period of 21.25 μs outranges the frequency needed by SAXS1 beamline - 20 Hz - as the fastest image acquisition that can be done is 50 ms, due to available x-ray flux. Even with sufficient flux, the next bottleneck would be the fast shutter that opens in 25 ms and closes in 15 ms.

Despite that, the presented system was built to be prepared for new challenges in Sirius. In fact, the new Coherent and Time-Resolved Scattering beamline that will be built in Sirius, named CATERETE, will study fast kinetics in microfluidic and the beamline will need a few microseconds in resolution. So we could say that the architecture presented here is a strong candidate for Sirius beamlines control solutions. Nevertheless, we will need to build faster systems to achieve our new nanoseconds frontier and keep the work that was started with the presented system.

ACKNOWLEDGMENT

The authors want to acknowledge SAXS1 beamline staff Florian Edouard Pierre Meneau, Tiago Araújo Kalile and Carolina Vieira Comin who tested this system over and over, adding valuable comments and suggestions to improve reliability, correctness and precision of the solution.

REFERENCES

[1] EPICS website: `http://www.aps.anl.gov/epics/`

[2] H. H. Slepicka *et al.*, "LabWeb – LNLS Beamlines Remote Operation System", TUPPC037, Proc. ICALEPCS 2013

[3] N. Sherry *et al.*, "Remote Internet Access to Advanced Analytical Facilities: A New Approach with Web-Based Services.", *Analyt. Chem.*, 2012, Vol. 84 (17), p. 7283–7291

[4] G. B. Z. L. Moreno *et al.*, "On-the-Fly Scans for Fast Tomography at LNLS Imaging Beamline", *these proceedings*, THHB3O03, ICALEPCS 2015, Melbourne, Australia (2015).

[5] XRF description: `http://lnls.cnpem.br/beamlines/xafs/beamlines/xrf/`

[6] SAXS1 description: `http://lnls.cnpem.br/beamlines/saxs/x-ray-beam/`

[7] L. Liu *et al.*, "Update on Sirius, the New Brazilian Synchrotron Light Source", MOPRO048, Proc. IPAC2014

[8] NI 9144 Expansion Chassis Under the Hood (2012)

[9] What is PXI? (from NI)

[10] Hybrid Photon Counting Detectors for Your Laboratory (from Dectris)

[11] MODEL SR570 Low Noise Current Preamplifier Manual (from Standford Research Systems)

[12] Operating Instructions and Specifications NI 9215 (from NI)

[13] Operating Instructions and Specifications NI 9402 (from NI)

[14] J. R. Piton *et al.*, "Hyppie: a Hypervisored PXI for Physics Instrumentation under EPICS", MOPG031, Proc. BIW12

[15] J. R. Piton *et al.*, "A Status Update on Hyppie - a Hypervisored PXI for Physics Instrumentation under EPICS", TUPPC036, Proc. ICALEPCS 2013

[16] LabVIEW System Design Software (from NI)

[17] Using the NI Scan Engine (ETS, VxWorks, Windows) (from NI)

[18] Basic EPICS Device Support, M. Davidsaver: `https://pubweb.bnl.gov/~mdavidsaver/epics-doc/epics-devsup.html`

[19] EPICS Channel Access Overview, K. Kasemir, p. 5 - 6: `https://ics-web.sns.ornl.gov/kasemir/train_2006/1_3_CA_Overview.pdf`

[20] H. H. Slepicka, *et al.*, "Py4Syn: Python for Synchrotrons", *J. Synchrotron Rad.* 22, pp. 1182-1189 (2015).

[21] Control System Studio: `http://controlsystemstudio.org/`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

NEUTRON SCATTERING INSTRUMENT CONTROL SYSTEM MODERNIZATION - FRONT-END HARDWARE AND SOFTWARE ADAPTION PROBLEMS

M. Drochner, L. Fleischhauer-Fuss, H. Kleines, M. Wagener, S. v. Waasen
FZ Jülich / ZEA-2, Jülich, Germany

Abstract

When the FRM-2 neutron source went into operation (2002) and many instruments were moved from the closed-down Juelich reactor to the new facility, it was agreed on a choice of front-end hardware and the TACO middleware from ESRF. To keep up with software standards, it was decided recently to switch to TACO's successor - the TANGO control software. For a unified "user experience", new graphical user interface software "NICOS-2" is being developed by the software group at FRM2.

While general semantics of TACO and TANGO don't look very different at a first glance, and adaption of device servers seemed to be straightforward at first, various problems in practical operation were found, due to difference in state handling, timing behavior and error reporting. These problems, and the changes that had to be made to ensure reliable operation again, will be described.

INTRODUCTION

After some different developments turned out to be rather short-lived, partly due to political reasons, partly due to manpower issues, the "NICOS2" instrument control and user interface framework got administrative backing and developer personnel to be further maintained and adopted by neutron scattering instruments at FRM-2. (See [1] and [2] for some historic information.)

For device access, it was decided to abandon TACO and rewrite device servers to use the more modern TANGO framework. Since the basic functionality of TANGO device servers, the remote, synchronous, execution of commands, is quite similar to that of TACO, an easy migration was expected. It was assumed more or less that the old implementation of device server commands just needed to be adapted to a slightly different API syntax.

BOUNDARY CONDITIONS, DEVICE SERVER ARCHITECTURE

A number of device servers, in particular those which handle detectors, have a structure as shown in Fig. 1.

It is often a matter of discussion whether detector data which come as individual events should be histogrammed immediately during data acquisition. It is not subject of this paper to discuss this design decision – it should just be considered that real-time histogramming needs to be done anyway, because instrument operators want a live picture of measured data. If single event data provide no additional advantage, the cost (both in terms of runtime overhead and storage media) can be saved. Up to now, no compelling reason for single event data was found.

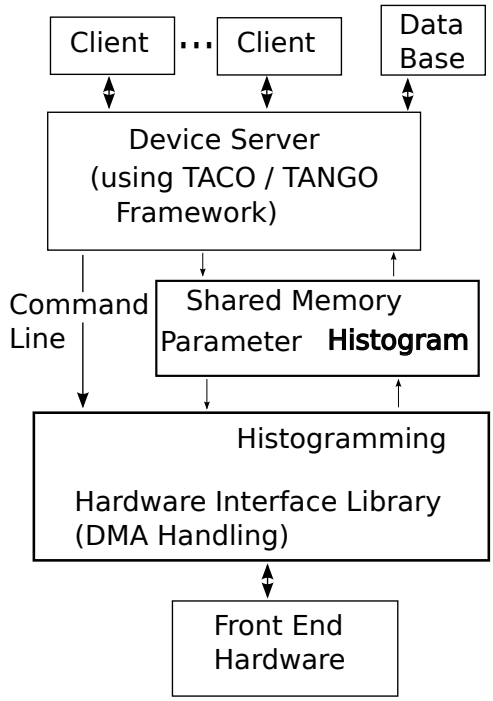


Figure 1: Structure of a detector device server which does real-time histogramming of incoming data.

Since TACO is not multithreaded from the beginning, and very likely not prepared to run with multiple threads active (actually, it was seen some years ago that even the underlying RPC library was not thread-safe), detector access was put into a separate process which runs asynchronously to TACO command execution. Another advantage of this was that the detector code could be run from a standalone process, for tests when the TACO manager and database services were not available.

Communication between the server core and the detector access program uses shared memory — keeping a table telling detector parameters (e.g. TOF slot settings) in one direction and the detector data histogram in the other. The shared memory was implemented as a memory-mapped file. This was helpful for debugging because the histogram data could be looked at everytime just using standard UNIX command line tools like od(1)/hexdump(1).

NICOS is a comfortable user interface to control the instrument, define and run measurement scans, show a life display of detector data and support logging and event han-

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

dling. Figure 2 shows just a little part of it, a window which shows results of instrument state polling.

This comes at a cost – it scans the instrument state permanently and accesses device servers from multiple threads.

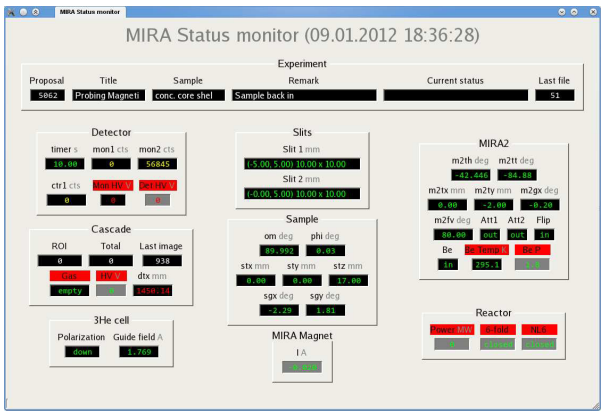


Figure 2: Screenshot of a status display window of the NICOS UI.

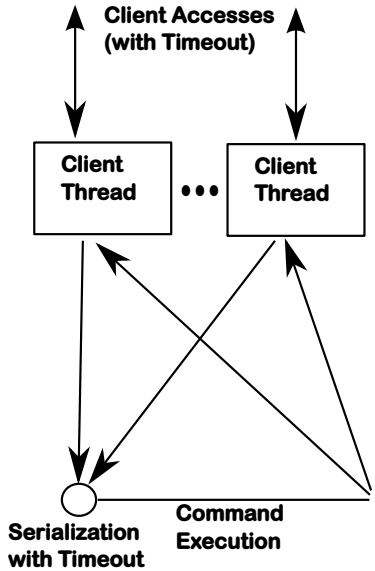


Figure 3: TANGO timeouts for transaction and serialization.

PROBLEMS SEEN, ANALYSIS AND SOLUTIONS

While the new TANGO device servers worked well in tests using just simple clients which do a single action at a time, strange errors occured when the full NICOS interface was used. Commands to start or stop a measurement failed due to timeout for no apparent reason (which is considered fatal), and the background task updating process variables for status display got timeouts as well (which is not fatal but leads to slow UI updates and logged error messages). Bumping the TANGO client timeout, even to unreasonably high values, did not help.

The error messages did indicate a "serialization timeout" in the depth of TANGO. After asking the original developers (see [3]) this led to the explanation that not the command currently executing is to blame for being too slow, but a command started previously by some other client task.

Each TANGO client connection is handled by its own thread. As Fig. 3 illustrates, these threads need to synchronize before hardware is dealt with, or if data global to the device are accessed. The maximum time to acquire this synchronization mutex is fixed to 3 seconds within the TANGO server framework.

Since the status display part of TANGO polls the state of devices permanently, and measurement control is done from another thread, there is a certain likelihood that calls collide and need to be resolved by the synchronization mutex. With TACO (see [4]), there was just a single thread multiplexing all client connections (select(2) in UNIX). An incoming request did stay in the network buffer queue until the server got ready to handle it, without any timeout. As long as the TACO clients had set their transaction timeouts large enough to cover the worst case, no error was reported – just sluggish user interface behaviour was possibly noticed.

This means in effect that no command within the TANGO server can be allowed to take more than 3 seconds, because subsequent commands or data accesses might be issued by another client within that time window. All commands which operate on slow hardware, and which we could just easily deal with before by bumping the TACO timeout need to be split into two halves – one which initiates the action and one which polls for the result.

One might argue that the fixed 3-second timeout can be easily fixed and made adjustable, but, on the other hand, any limit might be too small if a certain number of client threads is in the waiting queue. So it would have been a possible solution to handle errors differently in the clients and allow retries if possible. This would lead to practical problems however because client code is written by different people (most notably the NICOS developers), and it is not always obvious which commands are idempotent or where side effects can occur.

So it was decided to make sure that all TANGO commands are executed within much less than 3 seconds. It has shown that even simple operations like starting (fork(2) in UNIX terms) or stopping the external histogramming process mentioned above can take longer, depending on usage history (whether the program and data needed are already in cache) and operating system background activity. It even happened that the act of zeroing the histogram data array (256MBytes – for 64 channels and 20 bits time resolution) was not done in time after a period of inactivity, obviously because the data were paged out to swap space. The machine these problems were observed on, and where the problems were tracked down, is a Core2Duo with 2GBytes of memory. This is certainly not a powerful system by today's measure, but absolutely appropriate for a front end computer which just has a single task to fulfill.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

To be able to finish the TANGO calls in time, command semantics had to be redefined to be asynchronous. The command now just initiates the actions and returns immediately, actual work is done in a background thread. The TANGO server returns some "busy" state until the command is finished. The client needs to poll the state before it issues another command. This kind of side-steps the serialization mechanism described above, but it allows for more flexibility because clients can wait as long as appropriate for command completion.

Unfortunately, we could not add individual "busy" states telling all clients what the server is currently waiting for. The semantics of TANGO states is well-defined and custom values would cause confusion in other parts of the framework.

CONCLUSION

One can't assume that a TANGO installation works like "TACO, just using CORBA instead of Sun-RPC", even if just the subset of TANGO was used which is analogous to functions provided by TACO. The changes in timing behaviour are subtle, but require design changes towards asynchronous operation if any command can possibly take more than three seconds.

This three seconds limit must be met, even if the operating system (which is not aware of real-time requirements) has dedicated memory and CPU ressources to other tasks. On unfortunate occasions, even seemingly innocious acts like clearing some hundreds of megabytes of histogram data hits that time limit.

A standard desktop installation comes with many service tasks which potentially eat up system ressources and slow down intended uses, and there is a tendency that memory requirements grow on each software upgrade. Thus, hardware upgrades need to be done early enough.

REFERENCES

[1] T. Unruh, Instrument Control at the FRM-II using TACO and NICOS, Proceedings of the NOBUGS 2002 conference, 2002, arXiv cond-mat/021043

[2] M. Drochner et al., Adoption of the "PyFRID" Python Framework for Neutron Scattering Instruments, ICALEPCS 2013, San Francisco CA, 2013.

[3] TANGO control system, http://www.tango-controls.org/

[4] TACO control system, http://www.esrf.eu/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE NEW TANGO-BASED CONTROL AND DATA ACQUISITION SYSTEM OF THE NEUTRON SPECTROMETER DNS AT FRM II

H. Kleines, M.Drochner, M.Wagener, L. Fleischhauer-Fuss, S. Keuler, F. Suxdorf, R. Möller, S. Janasche, S. van Waasen, K.-H. Mertens, M. Bednarek, K. Bussmann, Y. Su,
Forschungszentrum Jülich, 52425 Jülich, Germany

Abstract

Forschungszentrum Jülich has been operating the neutron instrument DNS at the neutron source FRM II for about 10 years. DNS is a time of flight neutron spectrometer with polarization analysis that experienced a major upgrade in 2014 and 2015. During the upgrade DNS was equipped with new electronics and a new control and data acquisition system, including a transition from the existing TACO system to its successor TANGO. On the client side the NICOS software developed at FRM II is used for the implementation of measurement operations and user interface. The design of the new control and data acquisition system is presented and the lessons learned by the introduction of TANGO are reported.

INTRODUCTION

Forschungszentrum Jülich developed and operates about 20 neutron instruments at its outstations ILL in Grenoble, the Spallation Neutron Source in Oak Ridge and at the FRM-II in Garching near Munich. The control and data acquisition systems of almost all these neutron instruments are based on the so-called "Jülich-Munich Standard", a joint effort of ZEA-2 (central electronics institute of Forschungszentrum Jülich) and Technical University Munich (TUM) to define a common framework for the electronics and software of neutron instruments that is followed by most instruments at the FRM-II [1]. It is based on the TACO [2] control system developed by the ESRF and the extensive use of industrial type front-end equipment, e.g. PLCs, fieldbus systems or remote I/Os. Because TACO is considered to be outdated now, there was the decision to introduce its successor TANGO [3] at FRM-II as joint effort between TUM and ZEA-2. Since the neutron instrument DNS experienced a major upgrade it was selected as one of the pioneer instruments for the introduction of TANGO, simultaneously with the neutron instrument BIODIFF. At the same time the new instrument control software NICOS (developed by TUM) was implemented on both instruments.

THE "JÜLICH-MUNICH STANDARD"

The "Jülich-Munich standard" is a framework for the selection of technologies and components at each level of the control system. The definition of this framework was motivated by synergy effects and the reduction of spare parts on the shelf. A guiding principle for the framework was to minimize the development efforts and to acquire as much from the market as possible. A key component of the framework is the consistent use of industrial technologies like PLCs, fieldbus systems or decentral periphery in the front end. Main motivations are:

- low prices induced by mass market
- inherent robustness
- long term availability and support from manufacturer
- powerful development tools

A control system according to the Jülich-Munich Standard is organized hierarchically into the following levels:

Field level: The field level is the lowest level, at which devices that are not freely programmable reside, like motor controllers, SSI controllers, PID controllers, analogue and digital I/O modules, or measurement equipment. For all industrial type of I/O modules PROFIBUS DP based decentral periphery is recommended. Siemens ET200S is the preferred one. The ET200S modules 1STEP and 1STEPdrive are the predominantly used the stepper motor controllers.

Control level: The control level resides on top of the process level. Devices at the control level are freely programmable. They must meet real time requirements and guarantee robust operation in a harsh environment. At the control level Siemens S7 PLCs are used, because they dominate the European market.

Process communication: Process communication covers the communication of devices at the field and control level with supervisory controllers or computers. For lab equipment GPIB and proprietary RS232/RS485 connections are unavoidable. For industrial automation equipment PROFIBUS DP and PROFINET are the recommended choices. They are the dominating fieldbus systems in Europe and naturally supported by S7 PLCs and many other devices. A major reason for their success is the technological and functional scalability based on a common core as well as the programming model, which easily maps to PLC operation.

Experiment Computer: For economic reasons, all experiment computers should be PCs. Linux, being well established in the scientific community, is the only supported operating system. There is no definition of a specific kernel version or distribution. Direct device access should not be implemented on conventional PCs but on CompactPCI systems. CompactPCI allows deploying a variety of existing software in a mechanically more robust platform that fits into 19" racks.

Middleware: Since the framework aims at an inherently distributed system, software support for the transparent distribution of services between systems is

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

required. For this purpose TACO has been selected as the middleware system. TACO is a client-server framework developed for beam line control at the ESRF in Grenoble. In a TACO environment each device or hardware module is controlled by a TACO server. The server offers a set of device-specific functions, which can be accessed by TACO clients via a RPC-based mechanism over a TCP/IP network. To make its functions available to clients, the device server registers itself with the so called manager process and the data base server. The manager in combination with the data base server operates as a name server, which is consulted by clients to get the actual location of a device server. TACO includes a simple database for sharing of configuration data and operational variables between clients and servers.

Application level: On the client side, two variants of application programs are used: Where flexibility is desired and no GUI is needed, the scripting language Python is used. More static GUI applications are implemented in C++, using the "Qt" class library, with TACO access provided by device specific C++ wrapper classes.

NICOS

NICOS was developed around 2000 at TUM mainly by Tobias Unruh as general purpose instrument control software for neutron instruments at the application level on client computers[4]. It follows a client server approach and uses a consistent object-oriented device model on top of TACO.

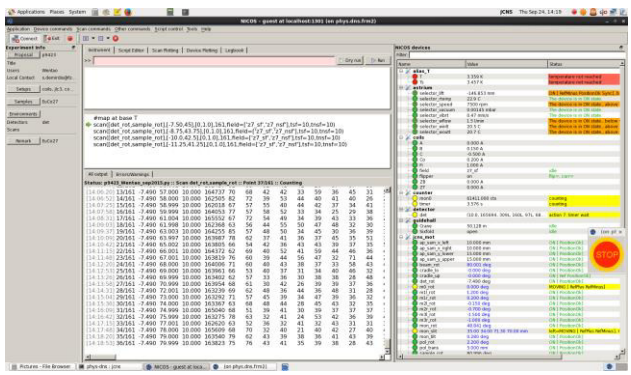


Figure 1: NICOS GUI for DNS.

Recently it has been completely rewritten, mainly by Georg Brandl from Heinz Maier-Leibnitz Zentrum (MLZ) [5]. It experienced major functional extensions, while keeping the essential ideas of the original implementation. NICOS is written in python using pyQt for the GUI. It supports TACO as well as TANGO.

NICOS offers scripting in python and in a simpler command language as well as a configurable GUI for graphical user operation. Functionalities comprise electronic logbook, history plots, detector data plots,…..

Due to its power and flexibility it was decided to use NICOS at all MLZ instruments in order achieve a homogeneous environment for instrument users and to reduce the overall maintenance effort.

BIODIFF and DNS were the first neutron instruments of Forschungszentrum Jülich in the process of replacing the existing instrument control software by NICOS. The NICOS implementation on DNS was done by ZEA-2. Besides configuration issues, extensions for detector readout and data storage had to be implemented. A snapshot of the NICOS GUI for DNS is shown in Fig. 1.

TRANSITION FROM TACO TO TANGO

TANGO is a successor of TACO based on the middleware system CORBA. Contrary to TACO, it is consistently object-oriented and removes many of the deficits TACO had. As an example, it provides generic multi-threading, data caching and proper event handling. Additionally, it comes with many standard tools not available in TACO, e.g. alarm system, logging system, code generators for device servers, process data base, graphical editor for the configuration data base, start up tool.

As a consequence, TANGO is much more complex than TACO and the code base is huge – e.g. it is not possible to compile a complete TANGO distribution on low end front end systems in a reasonable time. Due to the much higher complexity of TANGO, it was clear that its introduction would be a major effort and that performance and stability could be serious issues.

Since TANGO is well-documented and has many structural similarities to TACO, e.g. the device access via functions/methods (contrary to process variable interface of EPCIS), it was relatively easy for developers to get acquainted with TANGO. Even more important, these similarities enabled a quite standardized approach to porting our existing TACO device servers to TANGO, which allowed reusing a major part of the existing code with minimal effort. Tests with simple clients showed a stable behaviour of our TANGO installations with good performance. But in real instruments experiments with NICOS clients we experienced several serious problems:

- NICOS relies on the consistent use of TANGO device states which is different form TACO.
- TANGO device servers are internally multithreaded (one thread per client) leading to problems with our existing code written for the single-threading model of TACO.
- An internal thread serialization timeout of TANGO requires special care in servers for slow hardware.

All these issues could be solved with an additional software effort in the implementation of device servers.

THE DIFFUSE NEUTRON SCATTERING INSTRUMENT DNS

DNS [6] is a versatile time-of-flight diffuse neutron scattering instrument for cold neutrons with polarisation analysis shown in Fig. 2. It is equipped with a supermirror-based polarizer and a large double-focussing monochromator. Lead blocks in the monochromator shielding automatically open in the direction of the outgoing beam, when the secondary spectrometer is

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

moved in order to change the neutron wavelength. A failsafe system controlling shutter and lead block movement has been implemented. DNS is equipped with a liftable selector and a chopper system. About 20 mechanical axes with stepper motors are used for variety of slits, movement of sample, detector bank,…

One detector unit for polarized neutrons consists of 24 standard ^{3}He detector tubes and one detector unit for unpolarised neutrons consist of 128 position-sensitive ^{3}He detector tubes.

Figure 2: The diffuse neutron scattering instrument DNS.

THE DNS CONTROL AND DATA ACQUISITION SYSTEM

Physical Architecture of the Control System

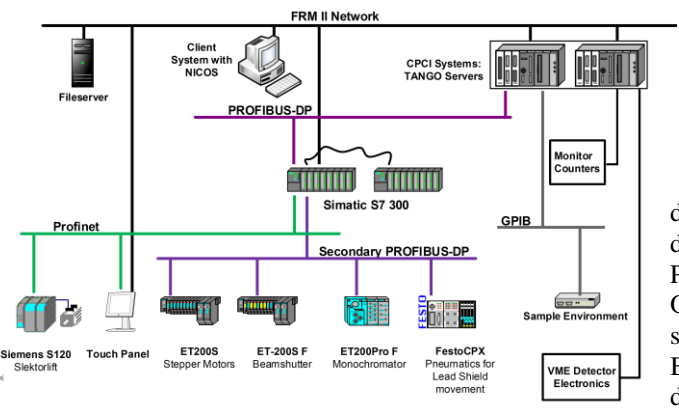


Figure 3: Physical architecture of the DNS control and data acquisition system.

According to Fig. 3 the control and DAQ system is implemented as a distributed system with a hierarchical architecture. On top of the system resides the so-called control computer with all application software – GUI-based as well as script-based. Via the experiment network the control computer accesses the "server computers", to which all front end systems (detectors, position encoders, motor controllers, digital IOs, analogue IOs, …) are attached. On the "server computers" TANGO servers are running, which access the peripheral devices via dedicated device drivers.

The "slow control" peripherals are indirectly connected to the "server computers" via a PROFIBUS segment with the main S7-300 PLC equipped with the failsafe CPU 319F-3 PN/DP. This CPU contains both the "standard" program as well as the failsafe program for the personal protection in parallel.

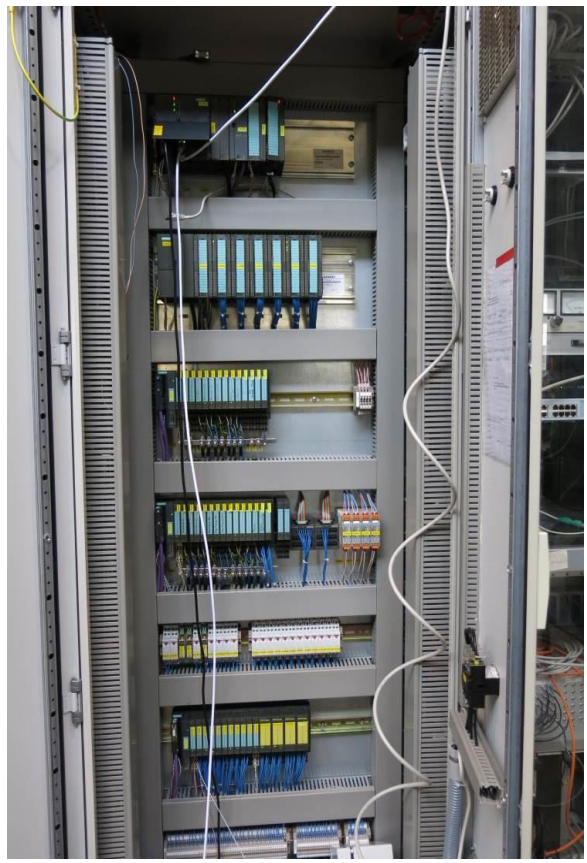


Figure 4: Main cabinet with PLC and 3 ET200S systems.

Stepper motor controllers and SSI modules as well as digital and analogue I/Os reside in three modular ET200 decentral periphery systems, which are connected to the PLC via an additional subordinate PROFIBUS segment. One ET200pro failsafe system and one Festo CPX systems are connected to the same PROFIBUS segment. Both modular systems with protection class IP65/67 are directly mounted on the monochromator without any cabinet. All failsafe signals are connected to special failsafe input modules in one ET200S system and in the ET200pro system.

A Sinamics S120 frequency converter for the selector lift drive and a touch panel for local operation of the PLC are connected to the PLC via a subordinate PROFINET segment.

The readout of the 24 standard ^{3}He detector tubes is done via time-stamper modules from Struck Innovative Systeme GmbH in a VME crate, which is connected via a SIS1100 optical link to the server computer. The readout of the 128 position-sensitive ^{3}He detector tubes is not yet

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

in routine operation and will be done by a Mesytec MPD system.

Software Architecture

As shown in Fig. 5, the implemented software is distributed between three levels of the system hierarchy. All software below the lower dashed line runs on the PLC in the front end. The software modules shown between the dashed lines are running on the server computers. This comprises TANGO servers and device drivers for dedicated HW modules, e.g. detector electronics, counter/timer board, PROFIBUS controller or GPIB controller. The TANGO middleware is the glue that connects the server computers to the control computer, where the client application programs as well as the TANGO database server (all above the upper dashed line) are running. Since TANGO is location-transparent, the application programs could run on any Linux-based system. The instrument control software NICOS, which internally has a client server architecture, resides as a client on top of TANGO using the pyTango interface.

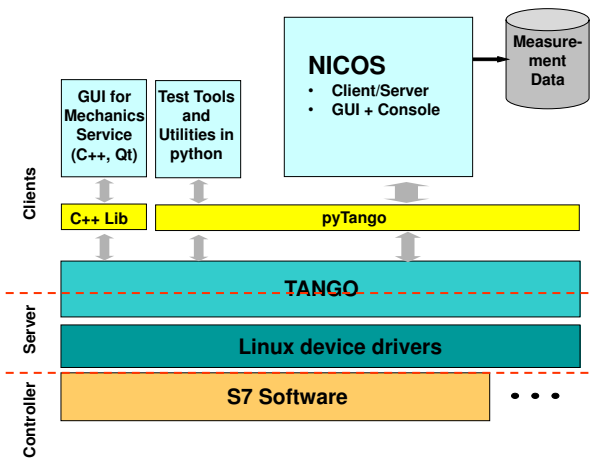


Figure 5: DNS software structure

CONCLUSION AND OUTLOOK

TANGO and NICOS have been successfully implemented in the new control and data acquisition system of the diffuse neutron scattering instrument DNS. After solving problems coming from the architectural differences between TACO and TANGO the instrument is in stable user operation, now.

On the base of the experiences with DNS, TANGO and NICOS will be used for all future neutron instruments of Forschungszentrum Jülich at FRM II and existing TACO-based instruments will gradually be upgraded to TANGO and NICOS in future.

REFERENCES

[1] H. Kleines et al., Implementation of the Control and Data Acquisition Systems for Neutron Scattering Experiments at the New "Jülich Center for Neutron Science" According to the "Jülich-Munich Standard", Proceedings of the ICALEPCS 2005, Geneva, 2005.

[2] A.Götz, W.-D.Klotz, E.Taurel, J.Meyer, J.-L.Pons, M.C.Dominguez (ESRF),TACO manual version 2.1, http://heanet.dl.sourceforge.net/project/taco/manual/2.1/TACO-2.1.pdf .

[3] Official TANGO homepage: www.tango-controls.org

[4] T. Unruh, Instrument Control at the FRM-II using TACO and NICOS, Proceedings of the Nobugs 2002, Gaithersburg, 2002.

[5] E. Faulhaber, G. Brandl, J. Krüger, B. Pedersen, A. Lenz, C. Felder, NICOS – The Instrument control solution for MLZ, Poster at the PCaPAC 2014, Karlsruhe, 2014.

[6] K. Nemkovskiy, Y.Su, S.Demirdis, W. Schweika, A. Ioffe, T. Brückel, DNS – a versatile diffuse neutron scattering spectrometer with polarization analysis at MLZ: present and future, Poster at Deutsche Tagung für Forschung mit Synchrotronstrahlung, Neutronen und Ionenstrahlen an Großgeräten, Bonn, 2014.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MOTION CONTROL ON THE MAX IV SOFT X-RAY BEAMLINES WITH TANGO AND SARDANA

M. Lindberg, J. Forsberg, L. Kjellsson, A. Milán,
C. Såthe, P. Sjöblom, S. Urpelainen, MAX IV, Lund, Sweden

Abstract

MAX IV Laboratory, a synchrotron facility in Lund, has selected TANGO as the control system framework for the entire facility. On the beamlines that are being built the Python-based SCADA (supervisory control and data acquisition) system Sardana will be used for experimental control.

SPECIES, one out of eight new soft X-ray beamlines, is used as a test bench for evaluating the chosen standards. Sardana is used to control the energy setting of the PGM (plane grating monochromator) as well as to provide macros and other utilities for the user. Generic Taurus GUIs and a SVG-synoptic give the user a way to interact with the control system and display relevant information. The standardized graphical interfaces give a familiar look and feel across the entire facility.

All motorized axes are controlled with the IcePAP motion controller. For the axes of the PGM, the IcePAP driver operates in hardware closed loop. Special care is taken in order to avoid slow and inaccurate movements of the PGM energy due to the non-linear relationship between the motors and the angular encoders.

INTRODUCTION

MAX IV Laboratory [1] will be a cutting edge synchrotron located in Lund, Sweden, and it is currently commissioning its main storage ring. This new facility builds on the experience from the previous MAX-lab in Lund, which is being decommissioned after 25 years of successful service to users from all over the world.

At this stage fourteen beamlines are funded and some are already in the process of being assembled at the MAX IV. The first users are expected in 2016. The SPECIES soft x-ray beamline [2, 3] is currently commissioned and taking users at the older facility MAX-lab with the intention to serve as a test bench for the control system that will be used at all the other beamlines at MAX IV.

During the commissioning of the SPECIES beamline, several motorized optical elements needed to be controlled to adjust the alignment of the beamline and its two branches, 40 meter long in total, as seen in Fig. 1. Also, the characteristics of the photon beam must be well known. This is achieved by using a set of macros making calculations from motor positions and current measurements.

MAX IV BEAMLINE CONTROL SYSTEM

By making use of control system standards already in use at other facilities, it is ensured that there are several tools such as pseudomotors, predefined scans and generic GUIs available from the start of the beamline commissioning. For the majority of the motorized axes the motion depends linearly on the corresponding encoder which makes it straight forward to achieve precise closed loop control but the PGM requires more effort which is further described below.

Control System Standards

The control system for the MAX IV facility [4] applies several standards for both accelerators and beamlines, that have been adopted with the aim to make the deployment and maintenance of a complex machine and a large number of beamlines more efficient and also to grant some familiarity to the user community.

Some of these standards are the TANGO Controls toolkit [5], the beamline SCADA Sardana [6], and the PyQt based Taurus GUI framework [7]. The IcePAP motion controller developed at the ESRF [8] has been chosen as a facility wide standard. Other common references and tools have been developed at MAX IV such as the MAX IV unified coordinate system and the SVG-synoptic interface [9].

Tango, Sardana and Taurus are all open source software and, since they have been developed and used at several other synchrotron facilities such as ESRF, Alba, and Soleil, code and experience have been accumulated over time. To deepen the transfer of knowledge and to contribute to the community, MAX IV is a part of the Tango collaboration, the Sardana collaboration, and the IcePAP collaboration [10].

Commissioning with the Control System

During the development a generic Taurus GUI was used to access motors and configure them, to run predefined scans and modify the scan parameters. The generic Taurus GUI is fast to set up and flexible to use. During the progression of the beamline installation and commissioning new panels for each equipment were added as each set of motors was configured. An example can be seen in Fig. 2.

After the commissioning a synoptic like the one in Fig. 3 will replace the development GUI. The synoptic gives a good overview of the beamline based on the physical location of the equipment. It also shows alarm states and allows quick access to detailed panels for all equipment when the user clicks the corresponding icon. Commonly used functions such as acquisition GUIs may also be launched from the synoptic.

MOTION CONTROL

Motion axes at MAX IV, including the ca 60 axes of SPECIES, will mostly be motorized using 2-phase stepper motors controlled by the IcePAP motion controller [11]. Sardana interfaces well with IcePAP and provides the possibility

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

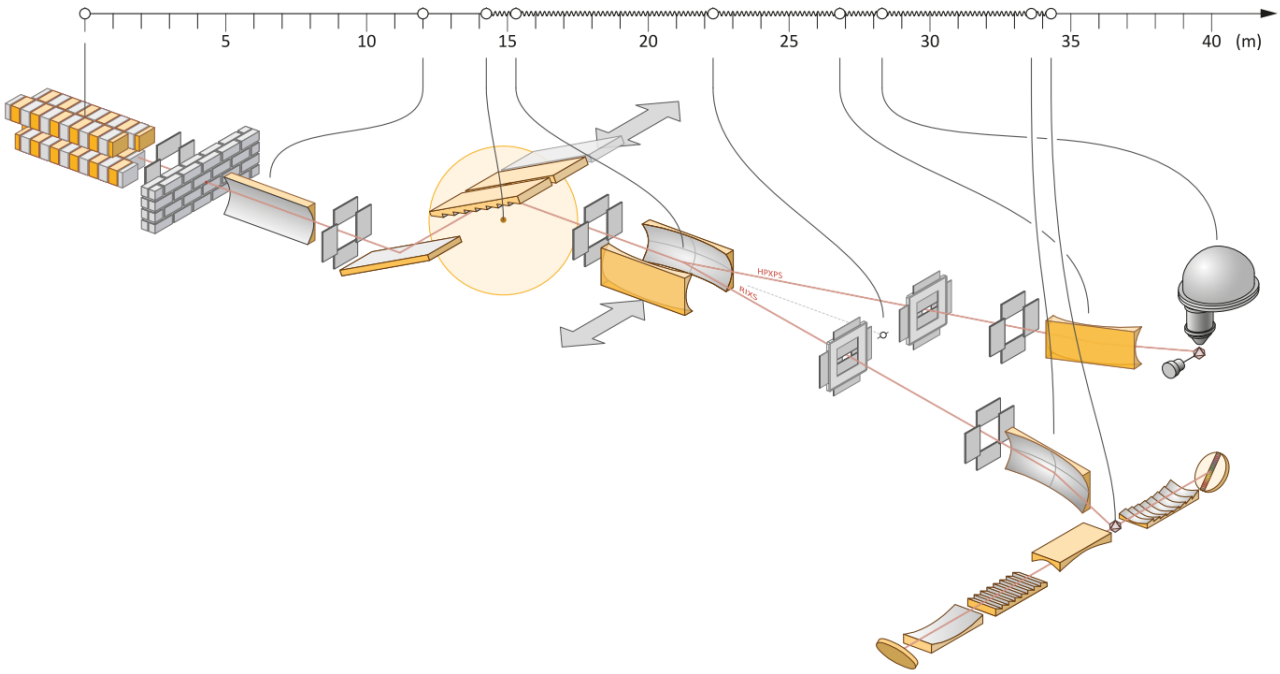


Figure 1: The general layout of the optical elements of the SPECIES beamline showing the main elements that guide the photon beam to the experimental station. Starting on the left: the undulator, a mirror, a set of baffles, the plane grating monochromator, a switching mirror to choose between the two branches, exit slits, more baffles, focussing mirrors and finally the HP-XPS and RIXS endstations.

to define pseudo motors to create complex motions, such as roll, pitch, and yaw, as well as compensate in software for mechanical nonlinearities.

Controlling the Energy of the PGM

The PGM has six motorized axes, but the most important are the mirror axis and the grating axis. On each of those axes, there is a non-linear relationship between the motor steps counted by the stepper motors and the counts of the angular encoders placed on the turning points of the mirror and grating mechanics.

The IcePAP driver operates the PGM axes in hardware closed loop, connecting the encoder signals as positional input to the IcePAP. This is done to ensure that the position is maintained with high accuracy. For the PGM it is important to overcome the non-linear relationship mentioned above, otherwise movements may be slow and inaccurate. The IcePAP accepts only motor steps as input for the desired position, but with a well tuned closed loop and a carefully chosen approximation of the encoder resolution as expressed in motor steps, the motor moves until the encoder value equals the wanted position.

CONCLUSION

Using the control system standards and tools described above we have achieved a stable control system for the SPECIES beamline. Much insight was gained along the way that will be applied for the commissioning of the new MAX IV beamlines.

Results for the PGM Energy

Measurements show a resolution of 0.5 µrad for the mirror and 0.3 µrad for the grating. A one-step movement in closed loop with the highest resolution, including the settling time, takes less than 0.1 ms for the mirror and the grating.

SPECIES' PGM uses RON905UHV angular incremental encoders from Heidenhain. These possess a small well known built in error. The error is visible when the two sine-shaped currents (phase shifted $\pi/2$ rad) from an encoder head moving over one encoder line are plotted against each other. Instead of a perfect circle, the graph is elliptical and offset. The solution is Heydemann correction [12].

In Fig. 4, at the left, this error is visible as a non linear increase in DAQ values during a linear motion. Especially the grating shows this cyclic error, isolated in the right plot in Fig. 4. From the cyclic error in the grating encoder head a correction table was created and each encoder value can now be corrected based on its DAQ value. This correction is implemented in a pseudomotor and the corrected positions are then used to calculate the energy.

To illustrate the performance increase, an N_2 absorption spectrum has been recorded in Fig. 5 and the width of the peaks as well as the distance between them are much more equal with the correction than without, just as expected. The maximum energy correction is 80 meV.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

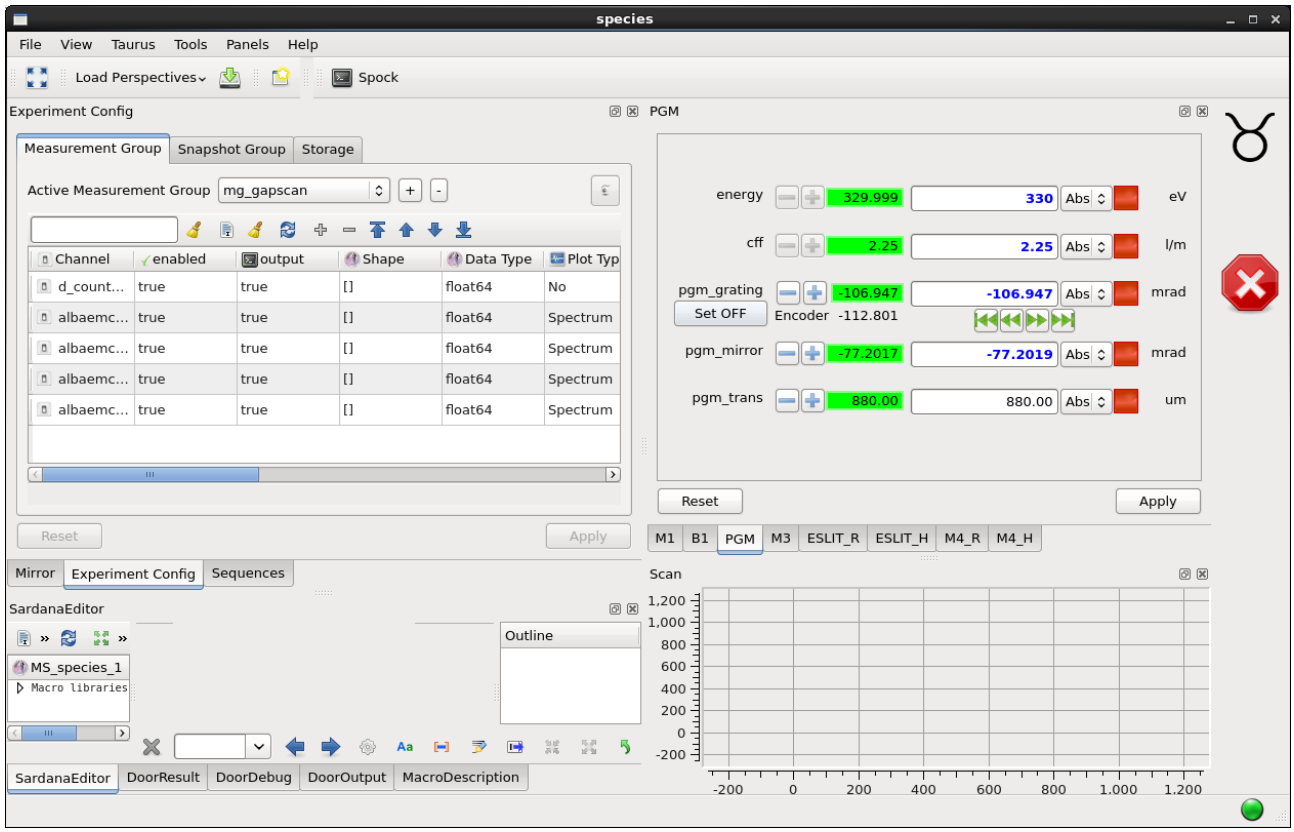


Figure 2: The commissioning GUI is a generic Taurus GUI which allows panels to be added with any combination of equipment and signals available in the control system. In the image above there are panels defined for the motors of each optical element. The GUI is also connected to the experimental control of Sardana.

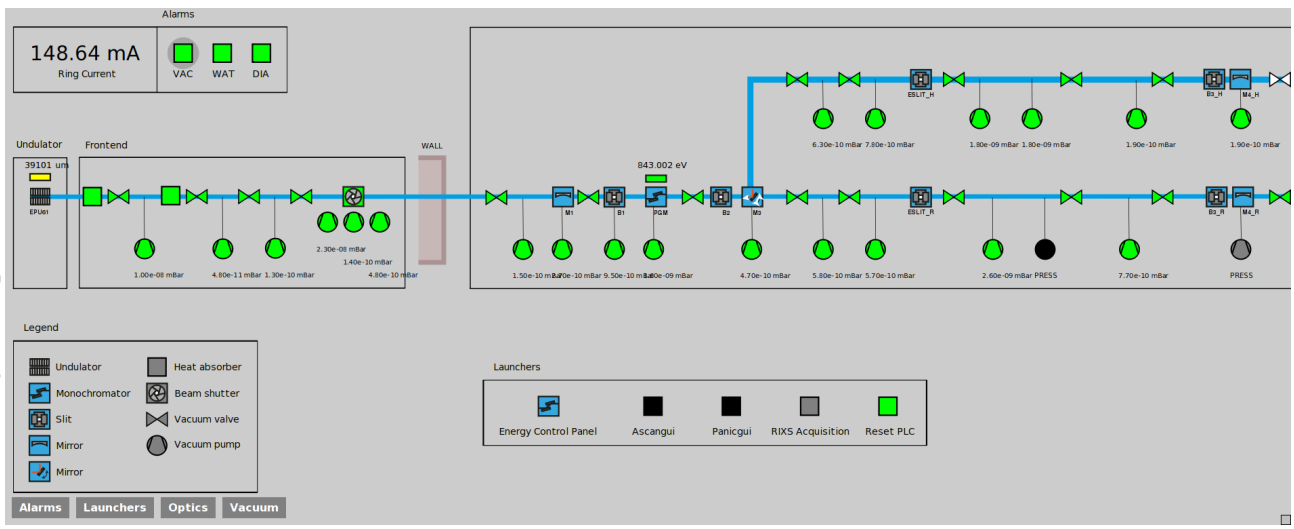


Figure 3: The SPECIES beamline synoptic provides an overview of the entire beamline with the possibility to display information about and control each piece of equipment. The view is defined by an SVG-image which is connected through a Python backend to the control system.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

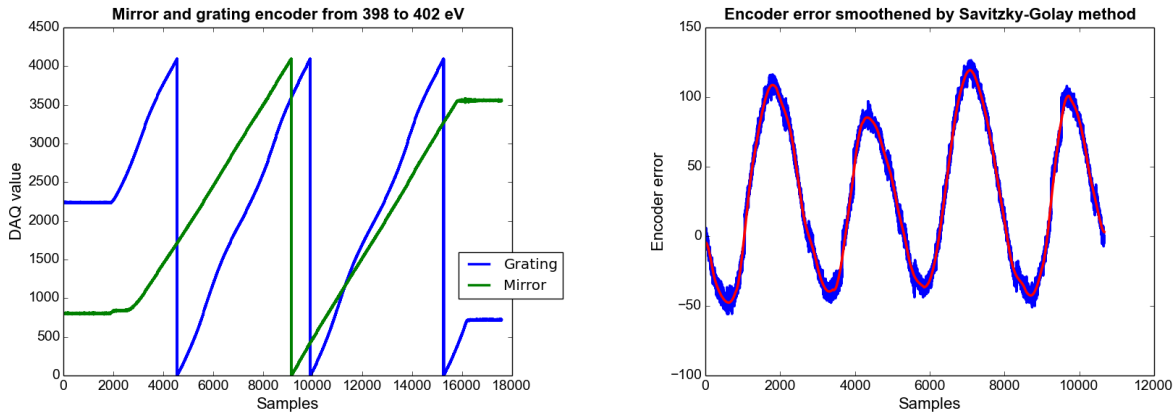


Figure 4: The left plot shows how encoder value increases in a linear motion and it is clear that the grating encoder suffers from a cyclic error. The right plot shows the cyclic error while the encoder passes a few encoder lines.

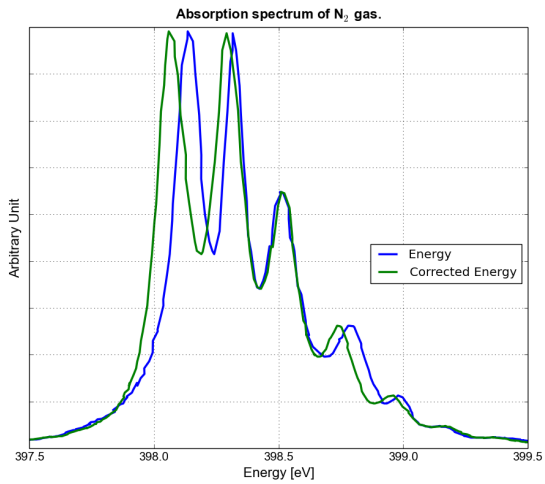


Figure 5: N_2 absorption spectrum with and without energy correction. The maximum correction is 80 meV.

REFERENCES

[1] The MAX IV project web page: https://www.maxlab.lu.se/maxiv

[2] The MAX IV web page about the SPECIES beamline: https://www.maxlab.lu.se/node/1505

[3] C. Såthe et al., "The SPECIES beamline at MAX-lab: a facility for soft-X-ray RIXS and AP-XPS experiments", in writing.

[4] J. Lidón-Simón et al., "Status of the MAX IV Laboratory Control System", Proceedings of ICALEPCS 2013, San Francisco, CA, USA, MOPPC109.

[5] The TANGO official website: http://www.tango-controls.org

[6] T. Coutinho et al., "SARDANA: The software for building SCADAS in Scientific Environments", ICALEPCS 2011, Grenoble. WEPMSO23. http://sardana-scada.org

[7] The TAURUS official website: http://www.taurus-scada.org

[8] N. Janvier, J. M. Clement, P. Farjado, "Icepap: An Advanced Motor Controller of Scientific Applications in Large User Facilities", Proceedings of ICALEPCS 2013, San Francisco, CA, USA, TUPPC081.

[9] J .Forsberg, V. Hardion, D. Spruce, "A Graphical Tool for Viewing and Interacting with a Control System", ICALEPCS2015, Melbourne, WEM309.

[10] V. Hardion et al, "Manage the MAX IV Laboratory Control System as an Open Source Project", Proceedings of ICALEPCS 2013, San Francisco, CA, USA, MOPPC086.

[11] P. Sjöblom et al., "Motion Control System of MAX IV Laboratory Soft X-ray Beamlines", SRI 2015, Synchrotron Radiation Instrumentation, New York City.

[12] J. Krempasky et al., "Heydemann interpolation correction for energy linearization of soft X-ray monochromators", Proc. of SPIE Vol. 8139, September, 2011.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SYNCHRONIZED RAMPING OF MAGNET POWER SUPPLIES FOR STREAMLINED OPERATION AT ENERGY RECOVERY LINAC (ERL) AND ELECTRON LENS (e-Lens)

P. Kankiya†, J. Jamilkowski, T. Samms
Brookhaven National Laboratory, Upton, NY 11973 USA

Abstract

Synchronous ramping of an assembly of magnets is critical for operation of beam in an accelerator. Magnet currents must remain within the operational limits to avoid dis-alignment of electron beam. In order to comply with the design specifications of ERL and e-Lens project, two different software control mechanisms have been developed. The ramp profile is automated and maintained by tracking current in all dipole magnets at ERL and superconducting solenoid magnets at e-Lens. This mechanism speeds up operations and adds a level of protection. The purpose of this application is to reduce unnecessary interlocks of the personnel protection system. This paper will describe the power supply arrangement, communication mechanism and the state machine algorithm used for feedback and control. A report on operating experience will be presented.

INTRODUCTION

In year 2015 Energy Recovery LINAC project went under commissioning and their operations required design of special magnet ramp sequence to ensure compliance with accelerator design. The Energy Recovery LINAC (ERL) is designed to serve as a test bed for future projects such as LeReC, eRHIC and CeC. The lattice is unique and has been designed with a significant amount of flexibility in order to cover the required operational parameter space [1]. Another accelerator project with special ramp profile requirements is the RHIC electron lens (e-Lens) [2]. The Yellow and Blue e-Lens assemblies have a collection of 17 superconducting magnets. Current reference waveforms for these solenoids are generated by high level software applications and are preloaded to the VME based Waveform Function Generator (WFG) modules which are custom boards developed in-house.

ERL MAGNET MANAGER

ERL's R&D prototype is currently installed with an assortment of magnets including several quadrupoles, dipoles and solenoids shown in Fig 1. The large beam size and very low longitudinal emittance of the e-beams dictate that the tolerances on the magnetic fields are very tight. Significant misalignment in the electron beam could lead to stray radiation; therefore very strict radiation safe functional bounds are enforced by an independent system called the Particle Accelerator Safety System (PASS). This system is designed to protect personnel working in the areas near ERL by interlocking critical devices under specific abnormal conditions. In order to avoid unnecessary PASS interlocks due to mismatched critical dipole magnet currents, it was advised by the external reviewers of the project that software controls be modified. The intent is to reduce accelerator downtime, since each interlock of this type requires a sweep of the blockhouse by trained personnel. Another role of this software that has been added to the operational requirements is to manage magnet power supply controls and ERL e-beam orbit feedback corrections.

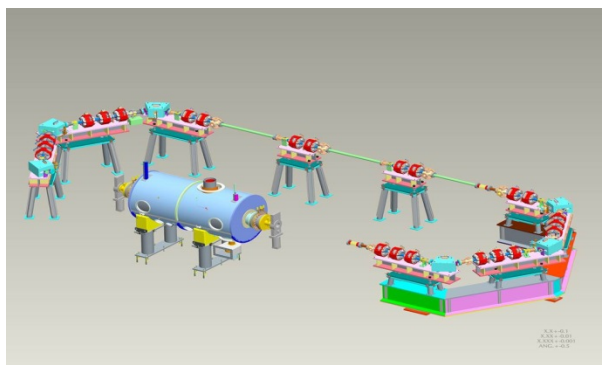


Figure 1: 3-D schematic of magnets in ERL beam line.

Software Implementation

Different COTS power supply systems with small amount of customisation are purchased and mounted to interface with various magnets based their on functional specifications. These supplies communicate over TCP/IP by means of custom multiple types of software developed using the Accelerator Device Object ADO framework [3].

"erlMagMan" is a program consisting of two levels of object oriented entities controlling power supplies which interface directly with the magnets. The first level is a an ADO class representing each magnet and consists of magnet properties such as bending angles of each dipole and nominal current at designed bending angle. These coefficients are provided by system experts and are obtained by direct measurements or simulations. Desired magnet currents are calculated at a given level of energy or at adjustment of trim angles. This energy will be entered manually by the shift operator or through the orbit correction program. Reverse conversion logic for translating magnet current to correction angles is also in place.

The second ADO layer acts as a governing class which interacts with all magnet ADOs simultaneously. It broadcasts the change of energy so all power supplies are configured with the correct current set point at the same time.

*† pkankiya@bnl.gov *Work supported by Brookhaven Science Associates, LLC under Contract No. DE-AC02-98CH10886 with the U.S. DOE

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

An asynchronous receiver client set up in the magnet ADO monitors the relationship between main dipole magnet output current and selected downstream dipoles. A threshold value of 2% less than PASS trip threshold is enforced and if any dipoles violate the limit, an alarm is generated alerting the machine operator to terminate beam in order to avoid a PASS interlock.

e-Lens WFG MANAGER

Designed to improve the proton beam luminosity in RHIC, an electron lens (e-lens) magnet system has been built at BNL in the 10 o'clock Interaction Region at the RHIC [4]. For the e-lens to function efficiently and facilitate the necessary cooling of the proton beam, the electron beam must be generated with a comparable size and aligned parallel to the proton beam within the acceptable magnetic field region of the main superconducting solenoid (Fig. 2).

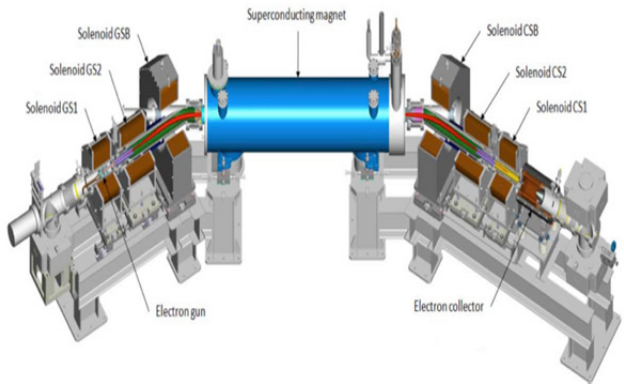


Figure 2: The Main superconducting magnet and corrector solenoids of blue e-Lens.

Software Implementation

The ramp design parameters of the software are provided by power supply engineers who are responsible for setting up the power supplies and quench protection system. Four different operating modes are defined, involving which magnets (main and fringe) are to be controlled and which e-Lens (Blue or Yellow) is targeted. The ramp rate of each magnet varies at different stages and direction of the ramp, increasing or decreasing current. As listed in Table 1, the ramp rate of each magnet has been predetermined and must be maintained while ramping. This is critical to prevent the superconducting magnets from quenching, as each quench will lead to machine protection system interlocks of the e-lens as well as interrupt the operation of RHIC during recovery. The latter can be a costly affair and must be avoided.

Table 1: Maximum Ramp Rates for Stable Operation

Magnet Type	Ramp Rate (A/s)
Main Solenoid (Up/Down)	0.4/0.2
Fringe Solenoid	1

A test involving powering of all of the solenoids simultaneously to maximum power was performed. This situation proved to be problematic due to Quench Detector (QD) trips from large induced voltages generated by interactions between the coils and resulted in frequent trips of the anti-fringe solenoid QD and one quench of the main solenoid. In order to resolve this problem, system experts specified a new scheme such that the solenoid circuits will be ramped alternately in steps equal to 10% of the maximum test current. Hence, the software shall ramp main magnets first and fringe afterwards in an alternating fashion to reach the desired magnetic field level (see Fig. 3).

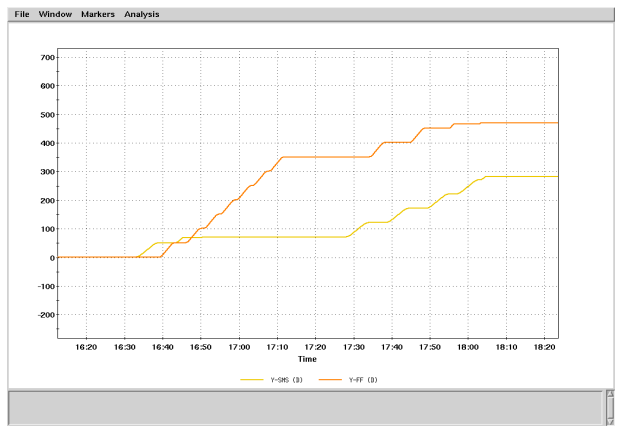


Figure 3: Ramp profile of yellow e-Lens main and fringe solenoids for 5 Tesla field. Each step is 50Amps.

The current set points of each magnet depend on the desired solenoid field, and these values have been established after systematic testing as described in [4]. Other software requirements include the ability to halt the ramping sequence.

Class Description

A C++ ADO class known as elensWfgControl is constructed to control all magnet supplies which are also represented as ADOs in the Controls System. In order to reduce the complexity of ramp sequence, it is partitioned into elements of state machines based on concepts such as states, transitions, and actions. State machines for the fringe and main magnets are performed separately, although the two associated threads are executed simultaneously. When the main magnet is ramping the auxiliary coils are in and idle state. Ramp step size and wait time between ramps referred to as park time are configurable via the control ADO interface.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

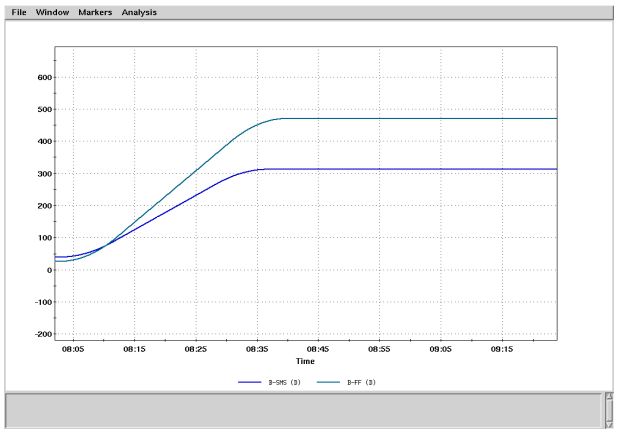


Figure 4: Ramp profile of blue e-Lens main and fringe solenoids with smooth ramp for 5Tesla field.

Upgrades to e-Lens Wfg Manager Program

Soon after commissioning of the ramp manager software, repeated testing showed that it is actually possible to ramp magnets in one step instead of 50Amps per step. The only caveat to this revelation is that the start of ramp has to be at a much slower rate. Keeping this is mind, a special scheme of slow ramps were implemented within the WFG code to reduce the current ramp rate by 20% during the first 20% of the ramp cycle. Figure 4 displays same level of magnetic field reached with significant reduction in ramp time using the smooth ramps.

ACKNOWLEDGEMENTS

Author would like to thank Dmitry Kayran at Brookhaven National Lab C-A Department for contributing design specifications of ERL magnets, and Al Marusic also at C-AD for designing the algorithm used for “soft” ramps of the e-Lens magnets.

CONCLUSION

A software scheme to generate two unique sets of magnet current ramping profiles has been created and deployed. Both resulted in successful beam steering in case of ERL and in determining the electron beam size and trajectory by the field in the solenoid in case of e-Lens. During operations and considerably reduced expert man hours by reliably automating each set of processes. A similar software management scheme is under consideration for the corrector magnets associated with the e-Lens system.

REFERENCES

[1] W. Meng et al, “Unique features in magnet designs for R&D energy recovery LINAC at BNL” presented at the 22nd Particle Accelerator Conference (PAC) Albuquerque, New Mexico June 25 - 29, 2007

[2] X. Gu et al., “RHIC Electron Lenses Upgrades,” IPAC’15, Richmond, VA, USA pp..3830 (2015); http://www.JACoW.org

[3] L.T. Hoff, J.F. Skelly, “Accelerator devices as persistent software objects”, Proc. ICALEPCS 93, Berlin, Germay, 1993.

[4] J. Muratore et al, “Test Results for the Electron Lens Superconducting Magnets at RHIC”

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MeerKAT CONTROL AND MONITORING SYSTEM ARCHITECTURE

Neilen Marais*, SKA South Africa, Cape Town

Abstract

The 64-dish MeerKAT radio telescope, currently under construction, comprises several loosely coupled independent subsystems, requiring a higher level Control and Monitoring (CAM) system to operate as a coherent instrument. Many control-system architectures are bus-like, clients directly receiving monitoring points from Input/Output Controllers; instead a multi-layer architecture based on point-to-point Karoo Array Telescope Control Protocol (KATCP) connections is used for MeerKAT. Clients (e.g. operators or scientists) only communicate directly with the outer layer of the telescope; only telescope interactions required for the given role are exposed to the user. The layers, interconnections, and how this architecture is used to meet telescope system requirements are described. Requirements include: Independently controllable telescope subsets; dynamically allocating telescope resources to individual users or observations, preventing the control of resources not allocated to them; commensal observations sharing resources; automatic detection of, and responses to, system-level alarm events; high level operator controls and health displays; automatic execution of scheduled observations.

INTRODUCTION

The MeerKAT Control and Monitoring (CAM) subsystem is the high level control system that enables all the other subsytems to operate cohesively as a single telescope. Most of the CAM design principles were presented at ICALEPCS 2013 [1]. While much detail design and implementation has been done since then, the only major architectural addition to CAM is the concept of telescope subarrays and the subarray-manager components that implement them. An unusual feature for the control architecture of a large scientific instrument is the lack of any messaging middleware or message bus.

A brief MeerKAT status update is given, and a high level overview of the MeerKAT CAM architecture is presented with a description of how it is used to meet CAM requirements in practice. The scalability and reliability of this architecture are discussed.

MeerKAT DESCRIPTION AND STATUS

MeerKAT will be a 64-receptor[1] aperture-synthesis inteferometric radio telescope array. Receptor antennas are offset Gregorian antennas with a 13.5m main reflector in a feed-low configuration. An offset optical configuration has been chosen because its unblocked aperture provides optimal optical performance and sensitivity with good rejection of unwanted radio frequency interference from satellites and terrestrial transmitters. It also enables the installation of multiple receiver systems in the primary and secondary focal areas and provides a number of operational advantages.

The antenna platform supports up to four receiver systems mounted on a carousel, allowing the telescope to switch frequency bands within minutes. The initial design committed to cryo-cooled L-band receivers. Some experimental uncooled K-band receivers are installed during commissioning. Since then, the MeerKAT project has committed to installing UHF-band receivers and the Max-Planck-Institute for Radio Astronomy has committed to funding and building S-band receivers for MeerKAT.

MeerKAT receptors directly digitise the received signal, without transmitting analogue signals to a central facility – a first for radio telescopes. A central Time and Frequency Reference (TFR) subsystem using a GPS-disciplined Rubidium clock provides a highly accurate absolute time reference and a sub-picosecond accurate clock signal for phase-coherent digitiser operation. The clock and timing pulses are distributed to the receptors via dedicated fibre links. The TFR will later be upgraded to a MASER timing source.

Signals are transmitted from from the receptor digitisers via multiple bonded 40 Gigabit Ethernet (40GbE) fibre links to the Correlator-Beamformer (CBF) subsystem in the Karoo Array Processor Building (KAPB) using the packetised SPEAD [2] format. A separate physical Ethernet network is used for CAM traffic from the receptors.

The CBF subsystem performs the first level of signal processing and data reduction using the SKARAB open FPGA platform [3], and multicast capable commodity 10/40GbE switches as a data fabric. It produces correlated visibilities for imaging, and beam-former voltage data that is consumed by the Science Processing (SP) subsystem and optionally by user supplied equipment provided by third parties. SP is a software subsystem running on general purpose computers in the KAPB, utilising GPU- and/or other forms of acceleration. SP processes the CBF output into images and other relevant science outputs, and also archives observation data.

Ancillary subsystems include: the MeerKAT site Camera subsystem; the weather monitoring subsystem; the KAPB building management subsystem that monitors ambient conditions and RFI-door intrusion detection where appropriate; the Power Distribution Units (PDUs) of several subsystems to facilitate emergency power shutdowns. These subsystems are not directly involved in observations but are monitored to ensure safe and reliable operation of the telescope.

All subsystems are coordinated and monitored by CAM using a logically separate CAM network, using Ethernet as a fieldbus. Communication with all other subsystems is via the Karoo Array Telescope communications protocol (KATCP) [4]. CAM Device Translators are used to provide a KATCP interface where this is not available natively.

* nmarais@ska.ac.za

[1] A receptor logically groups the subsystems of a single antenna dish.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

MeerKAT's full functionality is divided into six Array Release (AR1-AR6) milestones with the goal of deploying AR1 in H1 2016. During 2014 all the earthworks, power and other civil infrastructure were completed, including installation of fibre to all the receptor sites and construction of the underground, RF-shielded KAPB that hosts the MeerKAT data centre. First light was achieved with the first of the 64 receptors, confirming that the system as designed will significantly exceed the 220 m^2/K system sensitivity requirement once all 64 receptors are installed. This would make MeerKAT the most sensitive L-band radio telescope in the world. During 2014 the CAM subsystem team completed feature development for the Receiver Test System (RTS), and commenced detail design and development of the MeerKAT AR2 features.

During 2015 the RTS was deployed to site. RTS is a standalone mini-telescope system that can work with up to four receptors for receptor acceptance testing and commissioning. RTS implements the minimum telescope functionality needed to commission receptors. Antennas under test are separated from the rest of the MeerKAT telescope data network by patching their physical data fibres into the RTS data-network switch, while the physical CAM network is shared between RTS and MeerKAT through configuration procedures.

Eight receptors are on site as of October 2015 and are at various stages of integration, commissioning and acceptance testing. Rolling construction and deployment of the remaining receptors will take place through 2016 and into 2017. New receptors are first added to RTS before graduating to the operational MeerKAT system for science use.

During 2015 the implementation of the AR2 milestone CAM functionality was completed and verified against simulators pending the completion of other subsystems' functionality. AR2 requires 16 science-ready receptors and a CBF that produces a limited selection of correlator data products, and an SP subsystem that ingests said products in real time. Furthermore, basic subarray functionality is required.

The early phase of MeerKAT development is mostly about constructing and commissioning the 64 Receptors and procuring a sufficient number of SKARAB boards to provide proposed CBF functionality. The later phases, while less visible, encompass the majority of the "construction" effort. Telescope functionality is added through software (CAM, SP and CBF local control) and FPGA firmware (CBF).

COMMUNICATION LAYERS

The CAM architecture is divided into three layers of components as shown in Figure 1 with connections strictly from higher to lower levels, and a fourth category of CAM controller components that collaborate as models and controllers in the Model-View-Controller software model. Apart from **katportal** clients that use http and websockets, all connections are made using KATCP over TCP/IP; each directed arrow starts at a client making a TCP connection to a KATCP server running at a well known IP and TCP port. For subsystems external to CAM the IPs and ports are assigned by System Engineering (SE), while CAM components have IPs and ports assigned internally by CAM within the CAM IP range[2]. These IPs and ports are maintained in the CAM config (**katconfig**). All these connections are made via the CAM network[3].

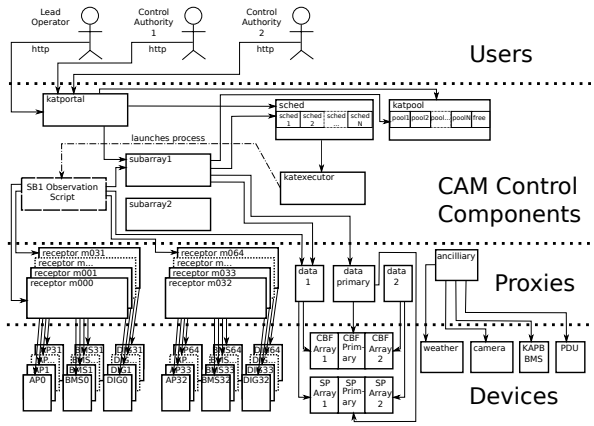


Figure 1: CAM Layers and Observation Control.

Members of the Devices layer each expose a KATCP interface, that is the interface between CAM and other subsystems; they are described using Interface Control Documents (ICDs) that are under SE configuration control. A single subsystem may have multiple instances, e.g. Antenna Positioner (AP) has an instance per receptor; each instance runs a separate KATCP server at a separate IP. These KATCP interfaces represent the controller of a specific piece of hardware, while the CBF and SP subsystems have their own internal control systems and present virtual devices. Devices make no KATCP connections via the CAM network, and act strictly as KATCP servers.

One level up are the CAM Proxies that make KATCP connections to devices, and act as KATCP servers to CAM controller components and to observation scripts. Each proxy exposes a single KATCP interface, but may connect to multiple logically related devices. Each proxy is managed by **katpool** (see next section) as a telescope resource. Devices are interrogated for KATCP sensors and requests, and these are proxied to the proxy's KATCP interface. Multiple instances of a proxy may exist, e.g. one receptor proxy per physical receptor, one data proxy per logical subarray. A proxy may implement special configuration/control for a device; e.g. the receptor proxies implement pointing corrections for the antennas.

Next come the CAM controller components that collectively implement the "intelligence" of the CAM subsystem. They may make KATCP connections to proxies and among

[2] The telescope network is treated as an infrastructure subsystem. IP ranges are assigned to various subsystems by SE.

[3] The CAM network is further segmented to avoid inappropriate communication between layers; the **katportal** component interfaces with the rest of the telescope LAN.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

themselves. At the top are the external telescope users. They connect to the **katportal** component using web browsers over authenticated http connections. They are allowed telescope control on the basis of their assigned roles. The two roles considered here are: Lead Operator (LO); and Control Authority (CA). The LO manages the assignment of telescope resources to subarrays, and the assignment of logged-in users as CAs of a particular subarray. The CA manages observations on a specific subarray. See [5] for more details of the operator interface.

OBSERVATION CONTROL

Figure 1 shows the main component interactions during observations. MeerKAT will initially have four[4] subarrays, but in the interest of space and clarity only two subarrays and the interactions of only one subarray are shown.

katpool is purely an accounting component; it keeps track of telescope resource allocations to subarrays and assignment to observations of resources within a subarray. Commensal observations are facilitated by only assigning controlling access of a resource to a single Observation Scheduling Block (SB)[5], while assigning non-controlling access to commensal observation SBs. **sched** manages the observation schedule of all the subarrays. **subarray1** is the subarray controller for "array1", managing its lifecycle.

The LO signs in to **katportal** and puts **subarray1** in configuration mode. She assigns (via the web GUI [5]) the SBs of the observations planned for her shift to **subarray1**, and assigns a subset of receptors and CBF product resources to **subarray1** as required by the scheduled observations. **katportal** requests **sched** (the CAM scheduler component) to assign the requested SBs to **subarray1** and requests **katpool** to allocate the requested resources to "pool1" (**subarray1's** resources). **katpool** moves the resources from the "free" pool to "pool1", making them unavailable to other subarrays. Satisfied with the configuration, the LO requests subarray activation. **katportal** requests **subarray1** to activate.

subarray1 observes **pool** to get its resource assignment. Next it requests the CBF primary interface[6] to set up "array1" with the list of receptors to subscribe to. The CBF primary interface is used for high level configuration and lifecycle management of CBF arrays. It creates a secondary KATCP interface for "array1". The **data1** proxy connects to this interface and exposes it the higher layers. **subarray1** then requests **data1** to activate the requested CBF products. CBF now assigns and programs FPGA resources to produce the requested products for "array1". **subarray1** configures SP to receive the CBF data product in a similar way.

subarray1 enters the active state, and the LO is notified. She delegates control to another user, making him the CA for **subarray1**; **katportal** will now relay all CA requests to **subarray1**. The CA puts the **subarray1** schedule in queue mode. **subarray1** relays this to **sched**. **sched** selects SB1 from the **subarray1** schedule and begins to activate it. It requests **katpool** to assign resources from "pool1" to SB1. **katpool** verifies SB1's resource requirements against those available in "pool1". If insufficient resources are available, the CA is notified, otherwise they are assigned to SB1.

Once resources are assigned, **sched** invites **katexecutor** to execute SB1, and informs **subarray1** that SB1 is now active. **katexecutor** examines SB1's instruction set to determine which observation script to run and the appropriate script parameters. It then launches the SB1 observation script in a subprocess. The observation script uses standard CAM Python library *katcorelib* to connect to **subarray1**; at startup the observation script has no knowledge of the telescope besides the address of **subarray1** and its SB ID code. Once connected, the script identifies itself as SB1. **subarray1** verifies that SB1 is indeed active, and replies with a datastructure describing the telescope resources assigned to SB1. This is passed to *katcorelib* which creates KATCP connections to the resources as required. The script is given a standard telescope object that exposes each KATCP resource (i.e. proxy connection) as a separate client object which exposes KATCP requests as function calls[7] and KATCP sensors as objects that can be polled, subscribed to, or monitored for a specific trigger value. Resource group objects allow all assigned receptors to be pointed to the same target with a single call, or the same sensor to be monitored across all the receptors. The observation script can now control its assigned resources until SB1 completes. Data products produced by the CBF are captured and processed by SP. SB1's metadata is also sent to SP so that the captured data can be accurately classified and archived.

Once SB1 completes, **sched** ensures via **katexecutor** that the script has terminated (ensuring that all SB1's KATCP connections are also closed), updates SB1's status to completed, and requests **katpool** to free the resources that were assigned to SB1. **sched** selects the next SB from the **subarray1** schedule, and repeats the process with the new SB. At the end of the observations, the CA or LO can request **subarray1** to be freed. The CBF is instructed to deprogram "array1", freeing up its FPGA resources, and **katpool** frees resources allocated to **subarray1** by moving them from "pool1" to the free pool.

MONITORING AND INTERVENTION

Figure 2 shows the main component interactions for telescope monitoring and interventions and is a logical overlay for Figure 1 – the monitoring connections are always open, whether an observation is active or not.

A number of **katmonitor** instances[8] make KATCP connections to all CAM proxies and all CAM controller com-

[4] CAM could support more subarrays, but limitations in other subsystems, the design of human factors in the control room, and the number of receptors in the telescope limit the useful number of subarrays to four.

[5] For scheduling, observations are broken up into several indivisible SBs.

[6] Interactions with SP are not shown, but follow a similar pattern to CBF.

[7] *katcorelib* only exposes functionality that the observation script should have access to by pruning the telescope object. Scripts could easily bypass this mechanism, but they are vetted by CAM beforehand. A future implementation might introduce an intermediate CAM component to limit access.

[8] Multiple instances allow multi-process scaling.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

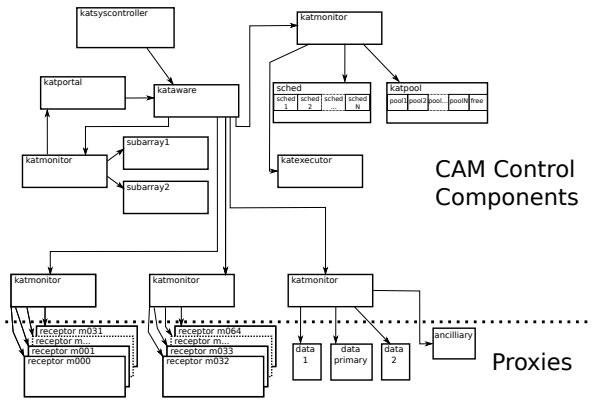


Figure 2: CAM Monitoring and Control Intervention.

ponents, and subscribe to every sensor by setting sensor strategies. **katmonitor** performs two functions on the received sensor samples: They are buffered in memory for the CAM sensor store [6], [7], and aggregation rules are applied in real time. Aggregation rules are defined in **katconfig** e.g. True if all the receivers on all receptors[9] are cold. The result of the rule is exposed as a new KATCP sensor on the **katmonitor** interface.

A single **kataware** instance observes all the **kataware** instances and applies alarm logic to the aggregate sensors. Alarm rules are read from **katconfig**. Each alarm has a rule for triggering an alarm level (e.g. WARN or ERROR), and an optional alarm action. **katportal** and **katsyscontroller** observe the **kataware** alarm sensors. **katportal** notifies logged in LO and CA users as appropriate; SMS, internet relay chat (IRC) or email notifications may also be configured. **katsyscontroller** implements the actions defined for the alarm and is responsible for taking telescope-wide action to ensure safe and reliable operation. It can make KATCP connections (not show in in Figure 2) to any proxies or CAM components needed to execute an action. Actions can be predefined (e.g. stow all receptor antennas due to high wind), or an arbitrary script file may be executed.

DISCUSSION AND SCALING ANALYSIS

MeerKAT CAM avoids the need for dynamic device discovery since all the nodes in the system are described in **katconfig** (see [1] for more details). This works well since MeerKAT is fairly static in terms of physical configuration; receptors are fixed in place, and have space for a limited number of receivers. While MeerKAT may potentially be expanded with more receptors, receptors are fairly homogeneous in configuration. Only a limited amount of unique calibration data is required per unit, making it almost trivial to add more receptors to **katconfig**.

[9] A **katmonitor** instance can only apply aggregate rules to the subset of proxies it is connected to. A multi-level **katmonitor** design is planned that will remove this limitation.

Direct TCP connections between components avoid the complication and overhead of message-bus middleware. All critical CAM operations are idempotent, so the combination of TCP and KATCP's request / response or timeout paradigm has been found to provide sufficient reliability. Some messaging middleware provide message multicast features that might improve efficiency for sensors with multiple listeners. Analysis of the MeerKAT design shows that the **katmonitor** components are the only high-bandwidth consumers of sensor data, implying that sensor multicast would not save much traffic.

From the CAM perspective, scaling a radio telescope essentially implies managing a larger number of receptors and associated sensor data. The CAM proxies are inherently scalable, since a separate proxy process is used per receptor and no state is shared. **katmonitor** is similarly scalable. The CAM sensor store and **katportal** GUI are already designed for performance scalability [6], [5]; **katportal** might need some redesign due to human factors. Most of the CAM control components (**sched**, **katpool**, **subarray1...N** and **katexecutor**) have very low transaction rates or are not directly exposed to the number of receptors.

A potential bottleneck is the observation scripts since they need to address every receptor. However, only a handful of receptor interactions are required per SB (continuous antenna pointing is handled by the receptor proxies), so this approach should suffice even for thousands of receptors. A similar concern applies to **kataware**. The addition of a multi-process receptor instruction repeater component would address this bottleneck should it ever become a limiting factor.

REFERENCES

[1] L. van den Heever, "MeerKAT Control and Monitoring - Design Concepts and Status", SKA SA, October 2013, ICALEPCS 2013 Proceedings MOCOAAB06.

[2] J. Manley, et al., "SPEAD: Streaming Protocol for Exchanging Astronomical Data", CASPER, June 2012, https://casper.berkeley.edu/wiki/SPEAD

[3] "Collaboration for Astronomy Signal Processing and Electronics Research", CASPER, https://casper.berkeley.edu/wiki/Main_Page

[4] S. Cross et al., "Guidelines for Communication with Devices", SKA SA, July 2012, http://pythonhosted.org/katcp/_downloads/NRF-KAT7-6.0-IFCE-002-Rev5.pdf

[5] T. Alberts and F. Joubert, "The MeerKAT Graphical User Interface Technology Stack", SKA SA, these proceedings THHC3O01.

[6] M.Slabber, "Overview of the monitoring data archive used on MeerKAT", these proceedings, THHD3O06.

[7] M.Slabber and M.T. Ockards, "Illustrate the Flow of Monitoring Data through the MeerKAT Telescope Control Software", WEPGF065 these proceedings, ICALEPCS'2015, Melbourne, Australia (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REPORT ON CONTROL/DAQ SOFTWARE DESIGN AND CURRENT STATE OF IMPLEMENTATION FOR THE PERCIVAL DETECTOR

A.S. Palaha, C. Angelsen, Q. Gu, J. Marchal, N. Rees, U.K. Pedersen, N. Tartoni, H. Yousef, Diamond Light Source, Harwell Science and Innovation Campus, Oxfordshire, UK
M. Bayer, J. Correa, P. Gnadt, P. Gottlicher, H. Graafsma, S. Lange, A. Marras, S. Reza *, I. Shevyakov, S. Smoljanin, J. Supra, M. Tennert, U. Trunk, C.B. Wunderer, Q. Xia, M. Zimmer, Deutsches Elektronen-Synchrotron (DESY), Hamburg, Germany
D. Das, N. Guerrini, B. Marsh, T. Nicholls, I. Sedgwick, R. Turchetta, Rutherford Appleton Laboratory (RAL), Harwell Science and Innovation Campus, Oxfordshire, UK
G. Cautero, D. Giuressi, A. Khromova, R. Menk, G. Pinaroli, L. Stebel, Elettra Sincrotrone Trieste, Italy
H.J. Hyun, K.S. Kim, S. Rah, Pohang Accelerator Laboratory, Pohang, Korea

Abstract

The Percival Collaboration is developing a high-speed, low X-ray energy detector capable of detecting single photons while maintaining a large dynamic range of sensitivity.

The increased brilliance of state-of-the-art Synchrotron radiation sources and Free Electron Lasers require imaging detectors capable of taking advantage of these light source facilities. The PERCIVAL ("Pixelated Energy Resolving CMOS Imager, Versatile and Large") detector is being developed in collaboration between DESY, Elettra Sincrotrone Trieste, Diamond Light Source and Pohang Accelerator Laboratory.

It is a CMOS detector targeting soft X-rays < 1 KeV, with a high resolution of up to 13 M pixels reading out at 120 Hz, producing a challenging data rate of 6 GiB/s.

The controls and data acquisition system will include a Software Development Kit to allow integration with third party control systems like Tango and DOOCS; an EPICS [1] areaDetector [2] driver will be included by default. It will make use of parallel readout to keep pace with the data rate, distributing the data over multiple nodes to create a single virtual dataset using the HDF5 file format for its speed advantages in high volumes of regular data.

This development project will culminate in a control and DAQ system capable of dealing with very high data rates while providing easy integration with site-specific control systems.

This report presents the design of the control system software for the Percival detector, an update of the current state of the implementation carried out by Diamond Light Source.

INTRODUCTION

The PERCIVAL detector is designed for high-speed and low noise detection of soft X-rays. The Percival CMOS chip is a large pixel array of ~13 mega-pixels with 15 bits per pixel (packed in 16 bit words) which encodes a substantially higher dynamic range of the sensor as described below [3]. This leads to 25 MiB per frame. For each data frame the detector will also produce a reset frame of the same dimension, so the total data produced is 50 MiB per frame. There will also be a ~2 mega-pixels version [4]. For the designed frame-rate of 120 Hz this leads to an overall data rate of ~6 GiB/s [5].

The Detector

The detector pixel is an adaptive gain design [4], allowing single photon discrimination at low flux while still being capable of high flux measurements at the cost of increased noise. It comprises a diode and three capacitors, each of increasing capacitance, to collect charge depending on whether the next smallest capacitance component has been filled; thus "adaptive-gain". As there are four "collectors" per pixel, there are four possible gain values also. A pixel array feeds scrambled data to a pair of "mezzanine" cards, each with an FPGA to perform a certain amount of low level unscrambling and formatting the data into UDP packets for transfer to the processing nodes. These nodes are a Linux cluster of 8 commodity servers, as recommended by an earlier feasibility study [3] in order to handle the processing of the expected data rate.

CONTROL SYSTEM ARCHITECTURE

Shown in Fig. 1 is an overview of the proposed Percival software architecture, where the green single direction arrows represent the flow of data and purple bidirectional arrows represent control links. The top half represents the data acquisition system, with the sensor board having its data collected by the carriers boards via LVDS lines and transferred to the mezzanine boards before it is sent through the deep buffer switch and onto the Linux cluster.

Each FPGA of the mezzanine board has four 10 GBps links to the deep buffer switch, one for control and three for transmitting data. Since there is a chance of UDP packet loss, and the goal for maximum packet loss is less than 1 in 10^6, a lightweight retry protocol will be used to ensure this aim.

The Linux cluster software stack and user workstation software suite comprise the control software that is being

* Mittuniversitetet, Sundsvall, Sweden

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

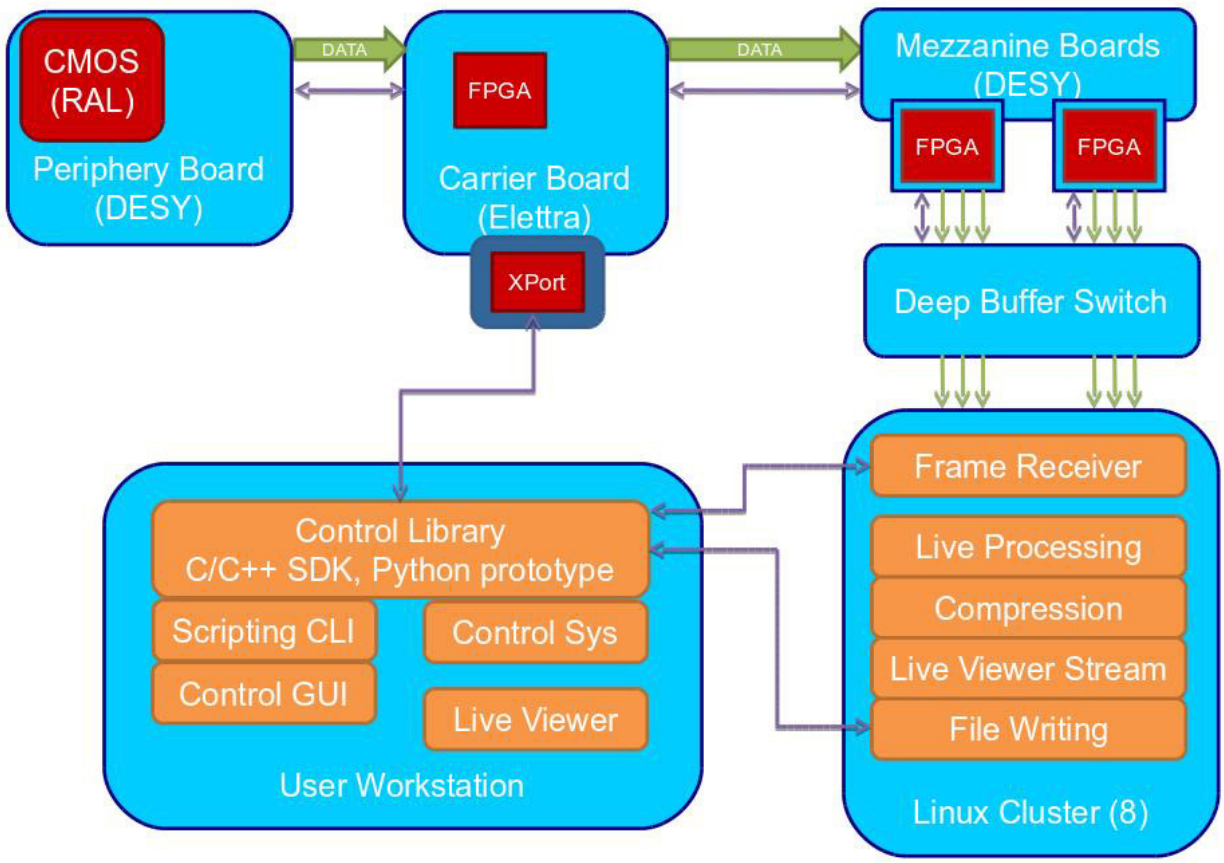


Figure 1: Percival Software Architecture: purple arrows represent control links, green arrows represent data links.

developed at DLS. The Linux cluster will run a chain of software for grabbing the data off the switch through 10 GBps Ethernet links for viewing and writing the processed data to file. The user workstation software will provide slow control of the detector and data acquisition, made of a core C/C++ library, with a Software Development Kit (SDK) to provide a means to interface with control systems.

Software Development Kit

The SDK will allow interfacing with controls systems such as EPICS, TANGO, DOOCS and others. A Graphical User Interface (GUI) can be applied with it to the control interface. The control software will come with EPICS and areaDetector support built in.

LINUX CLUSTER SOFTWARE

The Linux Cluster Software encompasses all the software involved in the live processing of the detector data. Live processing is necessary as it is required to be able to run the detector continuously, that is not only in burst or buffering mode, but also to take data almost indefinitely. The architecture is of a parallel processing pipeline with each piece of the software processing the data in turn. Each complete frame of detector data will be sent to a separate node in sequential order.

Frame Receiver

The Frame Receiver enables reception of frames of detector data transmitted over a network connection as a stream. It constructs data frames in shared memory buffers and communicates with external applications to allow processing and storage. It collects the data from the deep buffer switch into the cluster of Linux nodes, placing them in memory buffers to be accessed by the next step in the chain.

Live Processing

The live processing of the detector data into use-able images is implemented in a highly optimised C++ library.

The detector data at this stage has the format of a 16-bit integer per pixel. This is split into three parts; 2 bits for the gain multiplier, 8 bits for the fine Analogue Digital Converter (ADC) code and 5 bits for the coarse ADC code, with 1 bit left unused. The fine and coarse Analogue Digital Unit (ADU) codes represent the analogue voltage from each pixel as converted by the detector ADCs.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The first processing step is to decode the integer coarse, fine and gain codes from the packed 16-bit integer. The coarse and fine codes are converted to a floating point number representing the intensity measured by each pixel. This is called the ADU calibration, and requires a static array of offset and gain values for each of the seven coarse and fine ADCs.

Multiplying the ADU calibrated float by the pixel gain is the next step, the gain is one of four values encoded by the two gain bits of the now unpacked 16-bit integer. The gain values depend on which of the capacitors or the diode is read out from the pixel to the ADC, and depends on which pixel is read out (as the capacitances of the pixel components are not completely uniform over all the pixels).

Next, if the gain corresponds to the photo-diode of the pixel, i.e. the highest gain value, then the reset frame that has also undergone the previous processing steps must be subtracted from the sample frame. This is called the Correlated Double Sampling (CDS) subtraction step, reducing systematic errors in the signal.

Finally, dark image subtraction is performed by removing a constant frame from the result.

The coarse and fine offsets and gains, as well as the pixel gains and the dark frame are all intrinsic to the detector, acquired before measurements are taken and stored in static arrays available in memory for the processing steps to use.

In developing and optimising this library [6], the primary goal is achieving the fastest processing time, indicated by the bandwidth, i.e. the amount of data processed per unit time for both the sample and reset frames. Profiling tools were used to identify the most significant contributions to processing time and potential bottlenecks, and the library was developed as a processing chain so as to swap in and out different versions of the processing steps within the profiling and optimisation framework. Much of the tuning applied here is specific to the hardware and micro-architecture of the system.

- The first step taken to optimise the library was reducing the computation time in the first processing step by changing the floating point division to a multiplication of the denominator's inverse.
- The GCC compiler offers three levels of optimisation, and various techniques offered by the C++ language were employed.
- Parallelisation is employed using Intel Threading Building Blocks (TBB) in order to spread the processing of one node over multiple threads. It was found that the optimal number of logical threads was to use the number of physical threads, although some subtleties were encountered.
- Vectorising the operations such that calculations are performed on multiple pixels at the same offers further improvement in processing bandwidth, using the AVX (Advanced Vector eXtensions) instruction set that is enabled on Intel Ivybridge and Sandybridge architectures.
- Alignment of the data with the available vector sizes also improved throughput, requiring that the multiples of 7 pixels (due to the detector chip having 7 ADCs each with which to read out rows of pixel data) be padded with an empty slot.

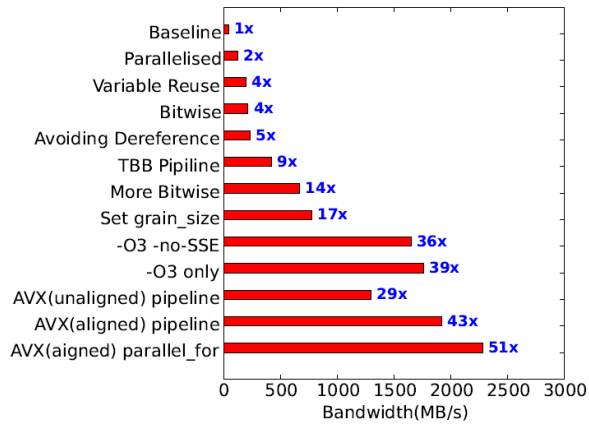


Figure 2: Optimisation steps versus processing bandwidth, with comparisons to the baseline [6].

The results of the various optimisation steps and the impacts on bandwidth are shown in Fig. 2. Improvements were optimal when parallelisation, compiler optimisation, replacement with bitwise operations and the AVX with a particular template better suited to large bandwidths is used.

Figure 3 shows the linear dependence of processing bandwidth with up to roughly 10 threads, after which there appears to be a saturation at ~2.5 GiB/s. The result shown is using a grain size, or chunk of data to process from the cache at one time, to be a factor of the number of pixels in the frame. The grain size is optimised to be a multiple of 7 (to match the number of ADCs, and the number of pixels rows is a multiple of 7), and is optimised to fill as much of the memory cache of the server node without saturating it. A grain size of 3528 was used for the result measured in Fig. 3.

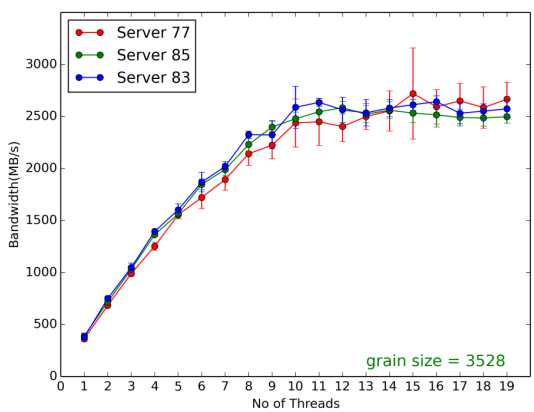


Figure 3: Processing bandwidth versus the number of threads [6].

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Overall, the optimised library is found to run stably on a single node at (2.5 ± 10%) GiB/s, which is 59× the baseline bandwidth, and more than 3× the required bandwidth to achieve the required data processing bandwidth.

Compression

Compression of data before writing to file has been investigated, with optimised parallel compression of interest in particular, to exploit the Linux cluster multi-core architecture. The "blosc" compressor tool [7] was used as it specialises in fast compression and decompression, use of multi-threading capabilities, and it can be applied to HDF5 data when writing to or reading from a disk. Initial and very basic measurement has yielded encouraging results but require further tests and development.

Live Viewer Stream

The live viewer requirement is only a reduced data-rate, sacrificing combinations of frame-rate, resolution and/or region-of-interest. Components for streaming the live data after processing on the Linux cluster already exist so will be incorporated into the Linux cluster software. This is also true for the live viewer; making use of an in-house developed version of an areaDetector plug-in that utilises the ffmpeg libraries to stream live data.

File Writing

File Writing is performed by an in-house developed file writer based on the HDF5 format. The HDF5 group is implementing support [1] for Single Write Multiple Read (SWMR) to the HDF5 libraries, allowing for live viewing of the data while the detector is capturing, processing and writing images [8]. This is an essential feature for long measurements, to inspect the data early throughout the data acquisition.

Also implemented is direct chunk writing [2]. This direct chunk write feature is also key in allowing us to do high-performance (i.e. outside of the HDF5 library) compression with something like blosc.

As each node of the Linux cluster will receive one complete frame in turn, and it has been found that parallel writing to one file is not as efficient as writing to multiple files. This will be addressed by adding Virtual DataSet (VDS) support to the HDF5 library. DLS, DESY and the Percival collaboration have jointly contracted the HDF5 group for this feature also. The HDF5 VDS will present data stored in several HDF5 datasets and files as a single HDF5 dataset and to access the data via HDF5 APIs without rewriting and rearranging the data [9].

CONCLUSION

The controls and data acquisition software for the Percival detector under development at DLS already has a design for the system architecture, and has achieved a number of significant milestones.

The Frame Receiver software can successfully receive data frames over a network switch, with a basic prototype working with the current mezzanine firmware. Live control and region-of-interest support is being developed.

Components developed at DLS for the live streaming and live viewing of processed data are ready to be used within the control framework for the Percival detector.

With the direct chunk write and SWMR feature, tests have shown acceptable performance writing to a parallelised file system.

A prototype of the scripting and CLI for detector control has been tested operating with emulated hardware.

A library written in C containing the live processing algorithms has been developed and extensively optimised to not only meet but substantially exceed the bandwidth requirements for processing data on each cluster node. Testing has shown it is capable of processing at a rate of (2.5 ± 10%) GiB/s on one of the Linux cluster nodes.

An interface for EPICS and areaDetector will be included in the developed controls software, and will be allow use with other control systems such as TANGO and DOOCS. The SDK also allows for the implementation of a GUI with the controls software.

REFERENCES

[1] Experimental Physics and Industrial Control System website: `http://www.aps.anl.gov/epics/`

[2] areaDetector website: `http://cars9.uchicago.edu/software/epics/areaDetector.html`, "areaDetector: EPICS software for area detectors", (22 September 2015).

[3] U.K. Pedersen et al., "Feasibility Study of PERCIVAL Data Acquisition Backend Architecture", IEEE NUCLEAR SCIENCE SYMPOSIUM & MEDICAL IMAGING CONFERENCE, (2014).

[4] C.B. Wunderer et al., "The PERCIVAL soft X-ray imager", 16TH INTERNATIONAL WORKSHOP ON RADIATION IMAGING DETECTORS, Trieste, Italy (2014).

[5] B. Marsh et al., "PERCIVAL: Design and Characterisation of a CMOS Image Sensor for Direct Detection of Low-Energy X-Rays", PSD10: 10th International Conference on Position Sensitive Detectors, University of Surrey, UK (2014).

[6] Q, Gu, "High Performance Detector Software for PERCIVAL Detector", *Diamond Light Source Summer Internship Program Report*, Unpublished, (2015).

[7] Blosc website: `http://www.blosc.org`, "Blosc, an extremely fast, multi-threaded, meta-compressor library", (22 September 2015).

[8] The HDF Group website, Single-Writer/Multiple-Reader (SWMR) Documentation: `https://www.hdfgroup.org/HDF5/docNewFeatures/NewFeaturesSwmrDocs.html`, (22 September 2015).

[9] The HDF Group website, Virtual DataSet Documentation: `http://www.bigdata.org/HDF5/docNewFeatures/NewFeaturesVirtualDatasetDocs.html`, (22 September 2015).

[1] contracted by DLS and Dectris

[2] funded by Dectris

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SODIUM LASER GUIDE STAR EMULATION

I. Price, RSAA, Australian National University, Australia
R. Conan, GMTO Corporation, Pasadena, CA 91101, USA

Abstract

In the era of extremely large telescopes (ELT) an adaptive optics (AO) system with artificial guide stars is an essential part of the optics between the source and the scientific instrument. In the case of the Giant Magellan Telescope these guide stars are formed by stimulating emission from Sodium atoms in the upper atmosphere with lasers launched from the side of the telescope. Moreover, they are resolved by the adaptive optics system so Shack-Hartmann wavefront sensors record elongated spots. Cost effective proof-of-concept systems for investigating control algorithms must be built for deployment in optics labs or on small telescopes. We present a hardware and software system that mimics the propagation of a single laser guide star (LGS) through the Earth's atmosphere and the optics of the Giant Magellan Telescope, using source motion and brightness modulation to simulate the source extension. A service oriented architecture allows adaptive optics scientists to construct images from different LGS asterisms and build non-real-time closed-loop control systems in high-level languages.

SODIUM LASER GUIDE STARS

A Sodium Laser Guide Star (LGS) is an artificial source created from Sodium atoms in the upper atmosphere and used for wavefront sensing as part of Adaptive Optics systems. Laser light at the 589 nm Sodium D2 line wavelength is propagated upward from a small telescope located alongside the main telescope. The photons from the laser interact with a naturally occurring population of Sodium atoms at an altitude of around 90-100km, exciting a shell electron. When the electron returns to its original energy level it emits a photon. Although the Sodium atoms are stimulated by a laser the process is dominated by spontaneous emission events. Ideally all of the energy from the outgoing laser beam would be delivered to atoms at a single altitude. In practice photons are absorbed all along the path through the Sodium layer, based on the remaining laser energy and the Sodium density. This results in a column of point sources that radiate some light back towards the main telescope, forming a laser guide star. Adaptive Optics systems use a wavelength filter to optically separate light from the laser guide star from the sources of scientific interest and natural guide star(s), and feed it to a laser guide star wavefront sensor (LGS WFS).

Shack-Hartmann wavefront sensors are based on lenslet arrays and imaging detectors such as CCDs. In essence, the pupil of the main telescope is divided into small patches, typically 100 to 200mm across. Each lenslet transforms the tip and tilt of the wavefront over the patch into a spot on the detector. Deviations in the spot positions gives an absolute measurement of the local tip and tilt of the wavefront. All lenslets in the WFS are observing the same LGS, but they each have a slightly different point of view, as illustrated in Figure 1. In the era of extremely large telescopes the size of the primary mirror is significant compared to a 15km long source 90km away. For a 25m diameter telescope, the spot elongation for the lenslet the furthest from the laser launch location will be 9.5". In contrast, a natural guide star spot has a typical size of 1".

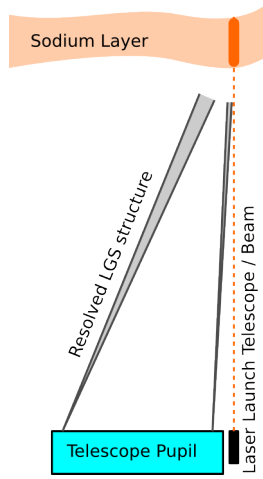


Figure 1: Column structure of the laser guide start source in the atmosphere and the variation of the angular size of the source with position in the telescope pupil.

Lenslets that sample the pupil far from the location of the laser launch telescope resolve the vertical structure of the LGS producing an elongated spot on the WFS detector, as shown in Figure 2. This distributes the signal from the LGS over more pixels than would be the case for an unresolved spot, increasing the error in spot position measurements and limiting the range of tip and tilt that can be measured without clipping of the input. Clearly the elongation of the spots has implications for the algorithms applied to WFS data for wavefront correction.

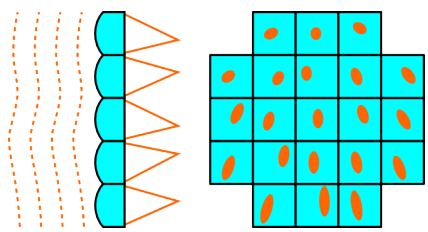


Figure 2: A distorted wavefront incident on the lenslet array of a Shack-Hartmann wavefront sensor (left) and the elongated spots in the image plane resulting from a laser launch telescope effectively located near the top edge of the pupil (right).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

EMULATING ELONGATION

In order to emulate laser guide stars in a lab, an optical system must be designed that scales the telescope-atmosphere system down to fit on an optical table and match the physical pixel size of available detectors. The telescope in this case is the Giant Magellan Telescope (GMT). The details of the optical design are not presented here, but suffice it to say a 10km LGS column maps to a small fraction of a millimeter in the lab. While no lab light source has the intrinsic physical structure required, an effective source can be generated by moving a point source during an exposure. For the range of travel and precision required a laser LED fed fibre is mounted on a precision piezo XYZ-stage. The position is set by an analogue input voltage for each of the stage axes. As well as moving the stage, the intensity profile of the source can also be controlled by modulation of the laser diode drive current. Many COTS laser LED controllers support current modulation via analogue input voltage signals. This scheme was used as it consolidated the LGS control scheme into four analogue voltages.

The point of adaptive optics is to correct wavefront errors introduced by atmospheric turbulence. The time variation in this tubulence is dominated by movement of layers of the atmosphere, driven by winds. In a lab experiment turbulence is usually introduced by movement of phase plates. These optical elements have specifically structured optical aberations. In the GMT emulation system three separate phase plates are used, each mounted on a rotary stage. Suitable placement of the stage along the optical axis sets the equivalent altitude of the turbulent layer. The rate of rotation of the stage sets the windspeed.

An adaptive optics system for the GMT will need to have a control loop rate of at least 500Hz. With an emulated LGS the loop rate is limited to about 1Hz. In each cycle an exposure is taken for each LGS. During the exposure the position of the source and the source intensity are varied to emulate a LGS. When the exposure is complete the source is moved to the starting point of another LGS. At the end of each cycle the phase plates are rotated through the small angle corresponsing to the wind speed and the control loop rate under test. This changes the wavefront at the WFS which affects the location of the spots on the image detector. The images recorded at each cycle are processed to compute the wavefront error and ultimately drive wavefront correction for closed loop control.

ARCHITECTURE

The LGS emulation system is one part of a general optomechanical control system. The primary user requirements driving the software design were to enable simple integration with AO algorithms written in high-level languages, have the flexibilty to redeploy the components for other experiments, and to be able to operate the system remotely. This naturally leads to a layered service-oriented architecture with minimal coupling between components at the low layers. It was also highly desirable to implement the lower-level control software for the Linux OS and utilize open source components. This guided selection of the hardware that interfaces with the lower layer control computer(s). An overview of the architecture is given in Figure 3.

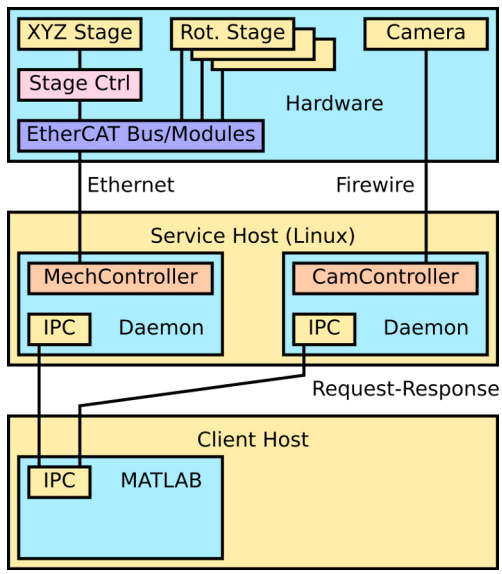


Figure 3: An overview of the hardware and software architecture of the control system.

System

From a control perspective, the rotary stages carrying phase plates are simply bipolar stepper motors and a home position indicator. Similarly, generation of an LGS profile amounts to generation of a four channel analogue output voltage signal. The EtherCAT fieldbus technology was chosen to implement all of these hardware controls, with all modules connected to a single bus coupler. This allow utilization of the open source Etherlabs IgH EtherCAT master and low level control from a Linux host. A digital output module was also included to allow the exposure of multiple cameras to be synchronized via the external trigger mechanism, and to implement laser safety interlocks.

The imaging devices at the core of the LGS WFS, and at other places in the optics, were selected based on pixel size, array size, readout noise, the maximum exposure time and cost. Although many camera interfaces are supported under Linux, cameras from Point-Grey Research with firewire interfaces were chosen. The open source dc1394 library was utilized to control and acquire data with these cameras.

Software

The software architecture follows object oriented design principles within the service and hardware interfacing layers. Each significant hardware component is encapsulated into a single service. All services utilize a common architectural framework. The service level is structured around a multi-channel request-response communication pattern. Each channel services requests sequentially and blocks while a request is being serviced, but the channels themselves are

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

independent and operate asynchronously. The framework requires each service be classified as a specific type of abstract controller, derived from the pure virtual Controller base class. Image devices are classified as CameraControllers and the LGS and phases plates are bundled together as a MechanismController. Communication channels accept request messages, parse them, and instantiate concrete classes derived from the Request class. The double-dispatch mechanism underpinning the *Visitor* design pattern [1] is used to deliver specific requests to specific controllers. The controller services the request and delivers a concrete instance derived from the Response class back to the communication channel. Responses are loosely coupled to the request type but strongly coupled to the nature of the data that needs to be delivered back to the client. The Request, Controller and Response class hierarchies and the communicaton channels provide the software infrastructure of each service. Each of the abstract controller classes declare a virtual interface that has a one-to-one mapping to a subset of the concrete request types. The request types encapsulate a single logical operation applicable to the hardware involved. For example, the camera controller has methods that commence an exposure, retrieve the image data from a completed exposure and attempt to abort an active exposure. The mechanism controller class is more generic. It's interface supports homing and positioning stepper motors and analogue signals.

A concrete class derived from one of the daugther classes of Controller is implemented for each type of hardware interface. It is at this layer in the software that thread-safe access to hardware is implemented. For example, the firewire cameras are all supported by the Dc1394Controller class via the dc1394 API, with pthread primitives utilized to ensure asynchronous requests from different channel have well defined behaviour. A simplified class diagram for the core classes and concrete classes utilised for LGS emulation are given in Figure 4.

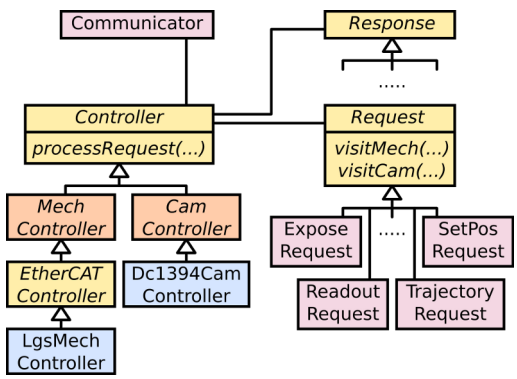


Figure 4: A compact class diagram for the laser guide star emulation services. Concrete classes from the framework are shaded in pink. The hardware specific classes that apply to a single service are shaded in blue.

Each service is implemented as a traditional unix daemon, instanciating a single concrete controller class and a single multi-channel communication class. The communication class provides the client-server support that facilitates remote operation. A simple message protocol has been implemented on top of the 0MQ (ZeroMQ) API and its REQ-REP pattern. A unique port number is used for each communication channel.

Integration of one or more services into high level languages is based on a thin client C-API and the ZeroMQ library. Each function in the C-API constructs a request message, sends this to the appropriate service, and decodes the response. The interface for MATLAB is based on the MEX function mechanism and is built on the C-API. The MEX interface was utilized from the outset for aligment of the optical system, calibration and general instrument commissioning. An example WFS image verifying the instrument and control system has the capabilities required to emulate a laser guide star is shown in Figure 5.

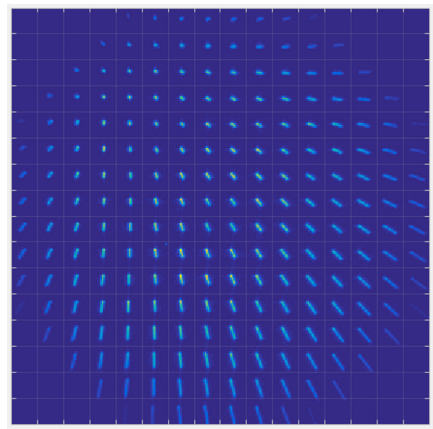


Figure 5: Data obtained from a WFS camera showing elongated spots as a result of moving the source and modulating its intensity during the exposure.

CONCLUSION

A modular service-oriented control system has been designed and developed to emulate the physical properties of laser guide stars. The hardware interfacing layer runs on the Linux operating system and leverages COTS hardware and open source software. A thin-client request-response communication model facilitates integration into high-level applications with a single third-party library, the only added dependancy for the client. Initial results from a Shack-Hartmann wavefront sensor demonstrate that the extended physical structure of a laser guide in the upper atmosphere can be sucessfully emulated in the lab.

ACKNOWLEDGMENT

This work was funded by the Australian Federal Government's Education Investment Fund (EIF) and used equipment located in the Advanced Instrumentation Technology Centre at the Research School of Astronomy & Astrophysics, ANU.

REFERENCES

[1] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns*, Addison-Wesley, ISBN 0-201-63361-2 (1994).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

HOT CHECKOUT FOR 12 GeV AT JEFFERSON LAB*

R. Slominski, JLab, Newport News, VA 23606, USA
T. Larrieu, JLab, Newport News, VA 23606, USA

Abstract

A new hot checkout process was implemented at Jefferson Lab for the upgraded 12 GeV accelerator. The previous process proved insufficient in the fall of 2011 when a fire broke out in a septa magnet along the beam line due to a lack of communication about the status of systems. The improved process provides rigorous verification of system readiness thus protecting property while minimizing program delays. To achieve these goals, a database and web application were created to maintain an accurate list of machine components and coordinate and record verification checks by each responsible group. The process requires groups to publish checklists detailing each system check to encourage good work practice. Within groups, the process encourages two independent checks of each component: the first by a technician, and a second by the group leader. Finally, the application provides a dashboard display of checkout progress for each system and beam destination of the machine allowing for informed management decisions. Successful deployment of the new process has led to safe and efficient machine commissioning.

INTRODUCTION

Hot checkout (HCO) is the process by which all required accelerator hardware and software systems are tested for safe and effective accelerator operation prior to beam startup. Checks are performed while the machine is locked up and energized. The goal of HCO is to reduce program interruptions due to faulty or misconfigured hardware and software and to minimize the likelihood of a failure that damages equipment. Such an event occurred at Jefferson Lab (JLab) on October 14 2011 after a small number of preliminary 12 GeV upgrade changes had been put into place. Two septa magnets overheated and ignited a fire causing both program delays and property damage (Fig. 1). The low conductivity water valves for the magnets were closed and the thermal switches on the box supply that normally would trip on an over-temperature fault failed because the switch was disabled via a jumper. The root cause was determined to be inadequate communication of machine readiness and poor work control practices [1].

More extensive changes to the machine, and thus more complicated hot checkouts during the upcoming 12 GeV era were going to need strict oversight. At the time of the magnet fire there was no formally documented procedure for HCO. The previous approach relied on a single individual, the so-called Restoration Coordinator (RECO) to obtain verbal confirmation from the multiple group leaders of the readiness of their systems. This approach was appropriate for shorter downs where few changes were made to the machine as the RECO did not have the authority or resources to ensure the groups responsible for the machine were meticulously verifying component status. A committee was formed to address the shortcomings of the RECO and produce a more stringent process and associated software tool [2].

Figure 1: Septa magnet MYR7S03 fire damage.

PROCESS OVERVIEW

The new HCO process can be organized into three stages: setup, upkeep, and signoff. The initial setup stage involves documenting components and their properties and only has to occur once. The upkeep stage is on-going and ensures the data put in place during setup remains up-to-date. Upkeep tasks often coincide with accelerator maintenance periods and include preparation for each signoff period. The signoff stage entails checking component readiness and occurs immediately before resuming beam operations after an extended accelerator maintenance period. In practice the signoff stage is usually done twice a year: once before the fall run and again before the spring run.

Scope

The rigorous HCO is reserved for use following extended maintenance periods of a week or more The HCO process is not used during machine running to track downtime as the HCO process is not designed to be a repair log. There is a separate process and tool for tracking downtime at JLab during machine operation [3]. The HCO process is analogous to an aircraft pre-flight checklist process: once the plane is in the air the checklist is no longer useful.

*Authored by Jefferson Science Associates, LLC under U.S. DOE Contract No. DE-AC05-06OR23177

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Furthermore, the HCO process focusses primarily on protecting equipment and reducing operational downtime; it coexists with the personnel safety system (PSS) certification process which remains the primary means for protecting people.

Setup

During setup all of the components in the machine that require checkout must be identified and classified. Components are organized into collections labeled as systems. Systems are further organized into a hierarchical tree of categories based on type (Fig. 2).

Components are placed together in a system if they are of a common type, share the same set of responsible groups, and use the same verification checklist. Placing components into systems helps responsible groups search and filter the large list of components, allows tracking of checkout progress based on systems, and avoids explicitly assigning an often redundant set of responsible groups to each component.

Two other properties of each component are established during setup: the region the component is located in and the set of beam destinations the component participates in. Both of these properties provide additional filters for searching for components and tracking progress. Providing the ability to determine readiness based on beam destination is particularly important since one or more of the experimental halls are often not included in the program and therefore are not needed to be ready for operations when other halls are.

Every group must publish a checklist for each system of components that it is responsible for. The checklist documents how the group determines that each component in the system is ready for beam operations. The checklists are web-accessible by all staff members.

Upkeep

One of the biggest upkeep tasks is ensuring that signoffs are downgraded when needed. Downgrades can occur at any time during the hot checkout process and by any staff member. When one group downgrades a component, the action will cascade to subsequent groups and require them to re-validate their prior checkout. Downgrades are also done administratively during maintenance periods when components are replaced, moved, or repaired. The JLab task tracking system, ATLis, is reviewed by management prior to a signoff period to determine what needs to be downgraded and re-checked.

On-going upkeep of component lists, component beam destination participation, responsible group assignments, checklists, and system and category organization is also necessary. Responsible groups must notify administrators if components are added, removed, or modified. Geographic integrators also play an important role in ensuring the list of components and their attributes is up to-date.

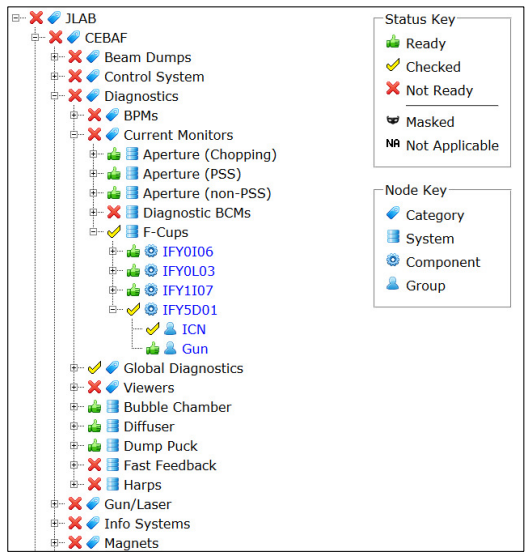


Figure 2: The component hierarchy as viewed from the HCO web application.

Signoff

Signoffs are done by responsible groups and require at least two checks per group. Each group indicates whether a component is ready, checked, or not ready. The typical flow is for a technician to verify a component and mark it as checked. A group leader then only marks a component as ready after conferring with the technician or after putting a second set of hands on the component. Only a group leader is authorized to mark a component as ready. Signoffs are recorded with a digital signature, timestamp, and optional user comment. Any user can make a comment without changing the signoff status. The history of all signoffs is recorded in the database and exposed via a web-interface for easy auditing.

Category readiness is determined by aggregating the readiness of the systems and components it contains. A component is ready once all responsible groups have signed off on the component as being ready. A system is ready once all of the components which it contains are ready. A category is ready only once all of the systems which it contains are ready. Given ready = 1, Checked = 2, and Not Ready = 3; the status calculation can be generalized as the following recursive algorithm: the status of a given node in the tree is the maximum status of its child nodes and in the base case of a component leaf node the status is set by the maximum of the responsible group signoffs. This hierarchical status roll-up allows users to quickly drill down to components which are not ready. Component attributes can be used to filter readiness for a particular beam destination, region, or responsible group.

There are two special signoff statuses: masked and not applicable (N/A). The operations director is given the administrative authority to mark as masked certain components which are not needed for safe machine operation. This is useful for example when a diagnostic that is not required for the upcoming run is non-

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

functional. The N/A signoff is used in situations where subsets of components in a system belong to different responsible groups. The non-responsible group signs N/A on the components it does not check. Masked components and N/A signoffs are ignored when computing the system/category status roll-up.

DATABASE

The new HCO process relies on readiness information stored in a centralized relational database. The initial list of component names was obtained from the JLab CEBAF Element Database (CED) [4]. The CED contains approximately 86% of the HCO components (7,403 of 8,644 as of September 2015). The remaining 14% (1,241) of components were obtained by soliciting group leaders and geographic integrators to verify the list of components and fill in any gaps. The components which require checkout, but are not in the CED include items that are not part of the EPICS control system, such as HVAC and other facilities items. Scripts were written to keep the databases in sync and are run periodically, especially before a checkout period.

WEB APPLICATION

The HCO web application provides a readiness dashboard for operators, management, and responsible groups to track checkout progress. The components in the machine are presented using a hierarchical tree widget that allows filtering by beam destination, region, and responsible group.

The application also provides graphical reports, a checklist repository, and a form for groups to sign off on individual components or collections of components. The application requires a published checklist for every system from each responsible group. The software will not permit a group to signoff components until the checklist for the systems is published. Further, the application maintains a list of group leaders and enforces that only they are authorized to signoff as ready.

A setup page provides forms for administrators to add, remove, and edit components and their properties, as well as manage beam destinations, regions, group leaders, and systems/categories.

INITIAL USAGE

The HCO process and software were utilized for the first time in September 2013 with the commencement of the 12 GeV upgrade commissioning. Figure 3 illustrates how as the HCO process progressed and new regions of the machine were verified as ready, project milestones were rapidly achieved. In fact, the accelerator division program goals for 2014 were completed five months ahead of schedule [5].

PROCESS PARTICIPATION

The HCO process has been a demonstrated success at involving a wide swathe of the lab personnel in a cooperative effort. To date, there have been 19,789 signoffs by 132 users across 43 groups. A common goal of achieving 12 GeV upgrade project milestones coupled with the recent collective memory of the YR magnet fire helped imbue staff with an early sense of common purpose. The early involvement of as many people as possible in the process design helped in creating buy-in and cooperation. Encouraging continuous feedback and improvement kept everyone engaged and has led to a better process.

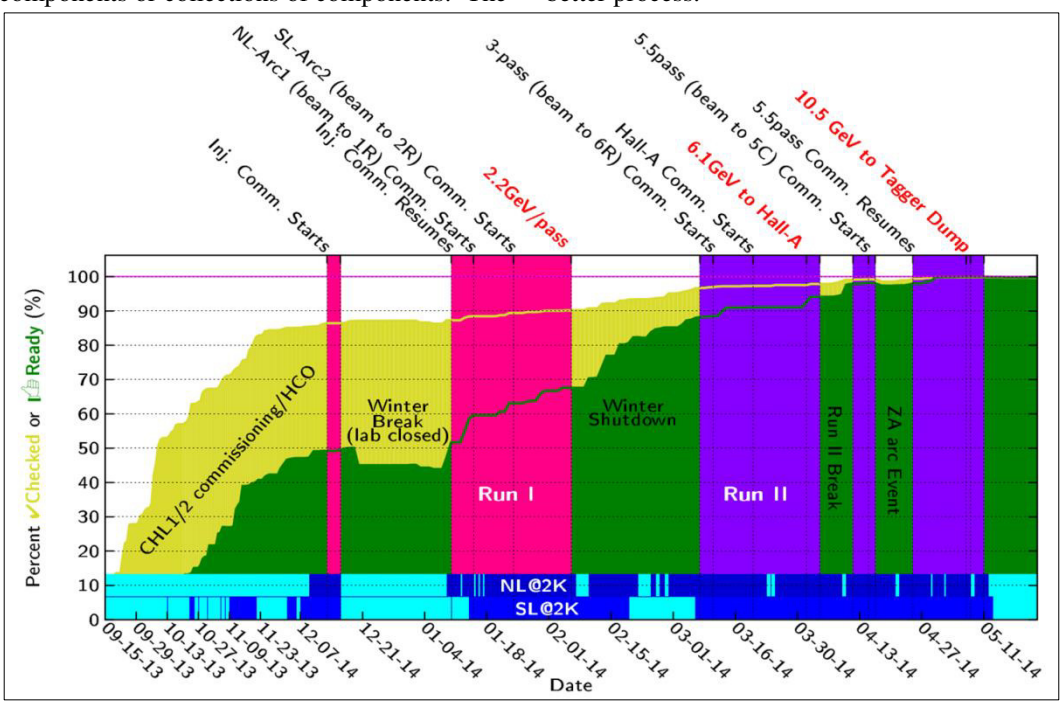


Figure 3: 12 GeV Commissioning Run I and II Progression. The yellow and green areas trace progress of verifying each component as checked and then ready. (Illustration reproduced with permission courtesy of A. Freyberger).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SUPPORTING PROJECTS

Several supporting projects were developed in parallel with the hot checkout process. The JLab wireless network infrastructure was upgraded to cover nearly the entire underground tunnel and a half dozen tablet computers were made available for technician use. The mobile computers allow technicians to view checklists and sign off on components while standing directly in front of the component in question. The Accelerator Bypassed-Interlocks Log (ABIL) project helps operators and technicians communicate bypassed component status using a digital log of bypassed components and physical tags placed directly on bypassed components.

CONCLUSION

We have developed a systematic method of checking, recording, and sharing machine readiness. The new process and software tool guides communication among operators, technicians, group leaders, geographic integrators, and management decision makers. The web-based tool provides a dashboard with real-time status reporting. The new process and tool contributed to a smooth 12 GeV upgrade project commissioning.

ACKNOWLEDGMENT

The JLab hot checkout committee is led by K. Baggett and comprised of T. Larrieu, R. Lauze, R. Michaud, R. Slominski, and P. Vasilauskis.

REFERENCES

[1] D. Green et al., "MYR7S03 and MYR9S04 Fire Incident Investigation," Jefferson Lab Repair Assessment Report, November 2011.

[2] K. Baggett, "New and Improved: The JLab (state-of-the-art) HCO System," Workshop on Accelerator Operations Proceedings (WAO 2014), Mainz, Germany, October 2014.

[3] R. Michaud, "System Downtime Management at 12 GeV CEBAF", the Accelerator Reliability Workshop (ARW 2015), Knoxville, TN, April 2015.

[4] T. Larrieu, M. Joyce, M. Keesee, C. Slominski, D. Turner, R. Slominski., "The CEBAF Element Database and Related Operational Software", 6th International Particle Accelerator Conference (IPAC 2015), Richmond VA, May 2015. MOPWI045.

[5] A. Freyberger., "Commissioning and Operation of the 12 GeV CEBAF", 6th International Particle Accelerator Conference (IPAC 2015), Richmond VA, May 2015. MOXGB2.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DRIFT CONTROL ENGINES STABILIZE TOP-UP OPERATION AT BESSY II*

T. Birke†, F. Falkenstern, R. Müller, A. Schälicke
Helmholtz-Zentrum Berlin für Materialien und Energie, Berlin, Germany

Abstract

Full stability of orbit and bunch-by-bunch-feedback controlled top-up operation becomes available to the experimental users only if the remaining slow drifts of essential operational parameters are properly compensated. At the light source BESSY II these are the transversal tunes as well as the path length and energy. These compensations are realized using feedback control loops together with supervising state machines. Key to the tune control is a multi source tune determination algorithm. For the path length correction, maintenance of beam energy and reduction of orbit deviations empirical findings are utilized.

INTRODUCTION

At lightsources, capabilities for user experiments rely on source point position, pointing stability as well as stable beam size and shape. These parameters transform to electron beam orbit and tune stability, typically controlled by slow (SOFB) and fast (FOFB) orbit correction feedback installations and by insertion device (ID) focusing feedforward compensations.

The intrinsic stability of the storage ring is greatly enhanced by the thermal equilibrium provided by top-up operation. In consequence correction amplitudes of beam guiding and shaping systems are drastically reduced, and beam quality can be better maintained by the described means.

Nevertheless, new constraints are imposed by the top-up injection permission, the technical solution implemented for FOFB and the stability requirement for efficiency of the Bunch by Bunch FeedBack (BBFB) [1] and for the amplitude of the resonantly excited bunch used for pseudo single bunch pulse picking. At BESSY II, additional control tools are needed to keep tunes, orbit and energy stable within required limits.

MOTIVATION

The ID-tune-feedforward system compensates the baseline tune shifts produced by gap and shift movements of any ID. This feedforward system interpolates empirically produced tables generated and iterated during measurements in dedicated ID-commissioning runs. Nevertheless imperfections and multiple IDs moving gaps and shifts at the same time would still cause significant tunes-shifts if not taken care of.

The purity of the camshaft bunch in the ion-clearing gap is controlled by a vertical knock out kicker. The resonant vertical beam excitation pulse during the gap around the camshaft bunch (see range marked blue in Figure 1) has to be very short to minimize perturbation of the camshaft bunch. So the kicker is active only for a few damping times at the injection shot. It has to be very efficient to prevent accumulation of charge in bunches to be cleared: If accumulated charge is cleared later when exact resonance condition is reestablished this might result in a shot injection efficiency below the allowed limits. Thus perfect matching of the excitation frequency and the vertical tune is needed to ensure excellent knockout efficiency.

Damping capability of the BBFB systems and the appropriate operational safety margin are well guaranteed as long as the tunes are constant.

Frequency and amplitude of the incoherent excitation for Pulse Picking by Resonant Excitation (PPRE) [2] have to match a specific frequency at the slope of the horizontal tune synchrotron sideband. Even tune shifts of about 500 Hz have severe impact on excitation effectiveness and stability of the PPRE bunch (see Figure 2).

Path length drifts were compensated by the slow orbit feedback (SOFB) by integrating the RF frequency into the horizontal response matrix and minimizing average horizontal corrector strength in the SVD calculation at each step. However, a straight forward implementation of this scheme was not possible with the fast orbit feedback system (FOFB).

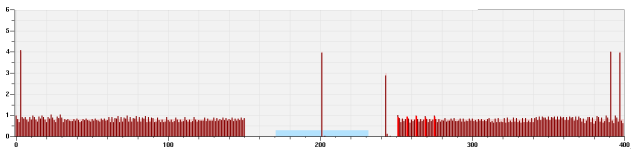


Figure 1: Fillpattern in standard multi- and single bunch hybrid mode, serving both average and timing mode experiments. A multibunch fill of 300 bunches including 3 bunches for fs-slicing experiments with an ion-clearing gap of 200 ns (100 buckets). A camshaft bunch at a fixed position in the center and a PPRE bunch close to the end of the gap - both for pseudo single bunch experiments. The blue region on both sides of the camshaft bunch is the cleared by resonant RF knockout during every injection shot.

TRANSVERSAL TUNE MEASUREMENT

Input Signals

Beammotion over a wide frequency range up to 1250 kHz is measured at a crossed stripline with a fast ADC/PXI system running LabVIEW. A processed FFT is provided to the control system in 3 partial ranges for each horizontal, ver-

* Work funded by BMBF and Land Berlin
† thomas.birke@helmholtz-berlin.de

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

tical and longitudinal tune with a resolution of 80 Hz. The measurement system delivers a sliding average of FFT data within each partial range of the spectrum, at a 10 Hz rate.

For standard optics as well as low-α optics, the coarse region of interest of the measurement system has to be set up to properly contain the main tune as well as the synchrotron sidebands of the desired transversal tunes.

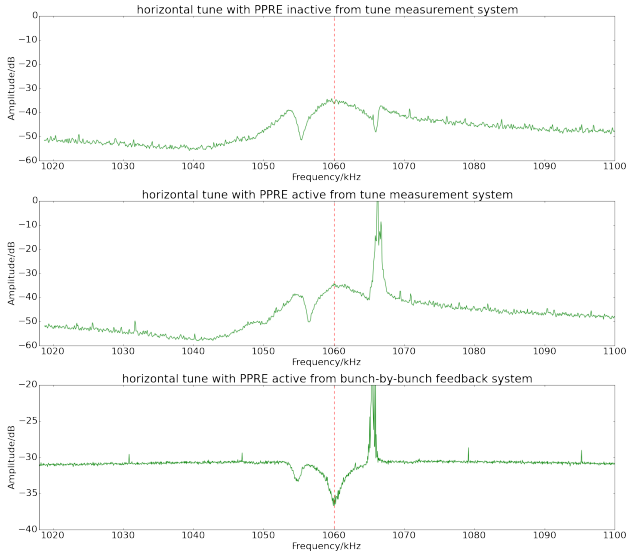


Figure 2: Typical horizontal tune spectrum - top: without PPRE excitation; center: with PPRE enabled; bottom: from bunch by bunch feedback with PPRE enabled.

The regular tune spectrum (top plot in Figure 2) has a slight asymmetry because the shifted tunes of high current bunches add a tilt to the spectrum.

Besides any possible unwanted distortions of the tune spectrum, there are inevitable additional types of beammotion e.g. due to the horizontal resonant excitation of the PPRE bunch (see Figure 1 and center plot in Figure 2). This excitation can make the actual horizontal tune impossible to determine by only analyzing the measured spectrum.

Tune-finder Strategies

A regular undistorted tune spectrum shows a peak at the mean tune of all bunches as well as two typically smaller peaks of synchrotron sidebands. In many situations, a simple peak finder is sufficient to identify the correct tune, but other non-tune-related beam motion may distort the spectrum and hence make the main tune hard to detect. Narrow-band distortions can easily be eliminated by removing outliers applying a smoothing algorithm to the spectrum. Since the main tune is not guaranteed to be the highest peak in the spectrum, a simple peak-finder often fails to identify the correct tune.

During pseudo single bunch experiments using PPRE, the motion of the excited bunch adds a major peak to the horizontal tune spectrum (see center plot in Figure 2). This may render a successful identification of the correct tune impossible, using the standard mechanism.

The horizontal BBFB system detects the motion of this bunch and delivers a spectrum showing the horizontal tune and sideband frequencies as notches and the excited PPRE bunch as a peak (see bottom plot in Figure 2) as the PPRE bunch is excluded from active damping. Consequently, switching tune-measurement to use the BBFB system as a source provides a stable and reliable determination of the tune during runs with PPRE enabled by performing simple notch-find on the selected region of interest within the horizontal BBFB system.

Depending on the scheduled optics and fillpattern, one of four algorithms is selected for each plane to determine the main transversal tunes from raw tune measurement FFT data. All these tune-finder strategies are working on a configured region of interest of the spectrum (see Figure 3).

Use BBFB peak- or notch-finder Used is if spectrum is distorted by active excitation e.g. during PPRE runs.

Simple peak search Finds the maximum value.

Peak search after smoothing While searching for the maximum value, this algorithm eliminates outliers by applying a simple smoothing algorithm with configurable window-width.

Area above threshold A simple pattern search (three equidistant peaks with the center one being the main tune) turned out to be a too simple approach to provide stable results. Instead, finding all identifiable peaks and selecting the one with biggest area below has proven to be a more reliable method. The threshold in this case is a minimum attenuation level att_0 where all relevant peaks are $att > att_0$.

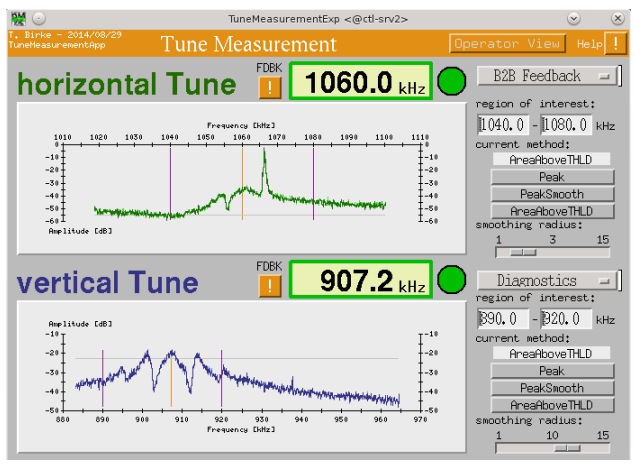


Figure 3: Tune measurement control panel - purple lines mark the regions of interest and a red line marks the identified tune for easy verification by operations staff.

Since injections appear as a major distortion in the measured beammotion, evaluation of FFT data and determination of tune values is paused for a configurable time during and after injections until the distortion is no more present in the smoothed spectrum.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

TUNE FEEDBACK

Once a reliably working tune measurement is established, it opens the possibility to setup a feedback system to keep the measured tune at its desired frequency. This way slow thermal drifts as well as residual tune shifts from ID movements can be compensated.

The transversal tune feedbacks are implemented to permanently monitor the motion of the measured tune as well as the difference from the configured target tune. Once, the difference exceeds a defined limit and tune motion is below a certain threshold, a configurable partial correction of the tune is applied. Subsequently, the resulting measured tune is checked for successful correction. If a correction was successful, further partial corrections are applied if necessary until the measured tune matches the target tune (see Figure 4).

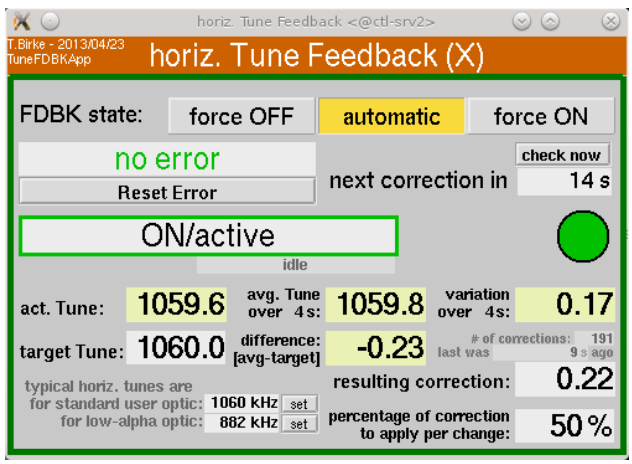


Figure 4: Tune feedback control panel.

User Level Configuration

The basic configuration parameter of the tune feedbacks is the target tune.

Some low-level configuration values control if, when and how any correction steps are performed. Tune corrections are inhibited temporarily

- For a specified time before and after injection shots, to not risk degradation of efficiency by applying a correction during an injection shot.
- When ring current does not exceed a minimum limit.
- If the measured tune varies more than a certain amount within the last few seconds, to not add additional noise.

Also configurable is the time delay between a adjustment step and the check for a successful correction.

To avoid applying additional noise, tune adjustments are performed at a maximum rate and only if a minimum step width is exceeded.

Error Handling

The following conditions lead to an error and the tune-feedback loop stops processing until error conditions are fixed and the feedback is reset manually. This measure is taken because distortions of the tune spectrum may cause the tune measurement to fail in identifying the correct tune. In this case, excess corrections may move the real tune to areas where subsystems relying on a stable tune won't work properly anymore. The tune may even approach a resonance, which could cause a partial or complete beamloss.

Error conditions are:

- Measured tune identified outside allowed window around target tune triggers a "*too much to correct*" error. The allowed window is small enough to not include the synchrotron sidebands.
- Correction step did not result in tune shift towards target tune causes a "*bad correction*" error.
- Variation of measured tune permanently exceeds a defined limit and hence leads to a "*measurement error*". The tune determinating algorithm could not identify a stable and reliable tune.

All these error conditions cause alarms to be signaled and an appropriate message sent to operators. Manual intervention and re-activation of TuneFDBK is necessary.

PATH LENGTH CORRECTION

Path length drifts are to be compensated by adjusting the RF master oscillator accordingly and hence keep the stred beam energy stable at a desired point.

Since FOFB cannot adjust the master oscillator directly, it instead uses corrector magnets to compensate for path length drifts and to keep the beam centered and stable. Over time excess strength in two corrector families accumulates.

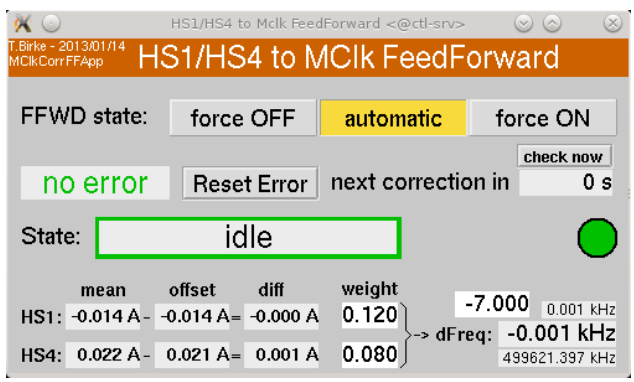


Figure 5: master oscillator FeedForward control panel.

A companion system for FOFB has been implemented, that transfers these excess corrector settings into RF frequency changes according to previously observed relations, and applies these changes "cautiously" (Figure 5). The relation between these corrector settings and the corresponding

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

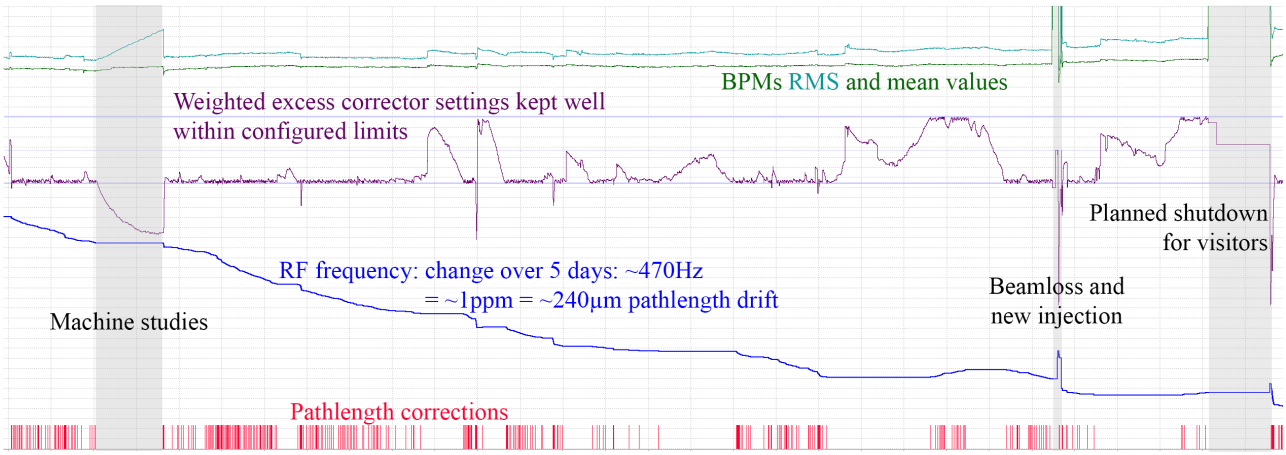


Figure 6: Plot of ~400 path length corrections over 5 days after one week of low current low-α operation. Shown are sum of the weighted excess corrector settings and the resulting RF frequency. BPM RMS and mean values are stable within ~1 µm.

RF change necessary, have been found empirically and is documented in [3, 4].

This companion system is implemented as a feedforward mechanism, that calculates the difference of the average corrector settings per family to a reference offset value. The results are weighted and transformed into a corresponding RF master oscillator change.

As the changes are applied, FOFB takes back the excess corrector settings applied previously.

Besides the influence of slow thermal drifts, driving undulators to small gaps and driving orbit bumps for PPRE experiments have the biggest share on path length changes (Figure 6).

Error Handling

After every adjustment of the master oscillator frequency, the corrector settings are checked. If FOFB did not respond to the RF change by taking back corrector settings, a "bad correction" error is raised, operators are notified by the global alarm handler and further corrections are stopped until manual reactivation.

Run Conditions

Some preconditions have to be met to allow path length corrections to be performed.

- FOFB has to be active and running
- ring current has to exceed 50 mA during multibunch and 11 mA during singlebunch runs
- the last injection shot was more than 3 seconds ago

Cautios application of changes is realized by scaling down the determined necessary frequency change (currently by a factor of 7).

A correction step is initiated as soon as the necessary change of the master oscillator frequency exceeds 1 Hz (RF frequency: 500 MHz). The real window width of ±7 Hz corresponds to a path length window of ~±3.5 µm.

CONCLUSION

The sketched setup of beammotion measurement, tune identification and tune-drift compensation has reliably stabilized operation at BESSY II. All other systems, that need stable tunes to work properly, benefit from basic beam properties.

Path length corrections have been applied flawlessly since FOFB is in use at BESSYII, and the resulting changes applied to RF master oscillator well reflect e.g. thermal drifts during accelerator startup and the seasonal and weekly changes of the outside temperature.

Performance and stability of top-up operation benefits from both drift control engines, and specialized setups for time resolved experiments heavily depend on these stabilized beam conditions.

REFERENCES

[1] A. Schälicke, P. Goslawski, M. Ries, M. Ruprecht, "Status and Performance of Bunch-by-Bunch Feedback at BESSY II and MLS", IPAC2014, Dresden, Germany (2014).

[2] K. Holldack, R. Ovsyannikov, P. Kuske, R. Müller, A. Schälicke, M. Scheer, M. Gorgoi, D. Kühn, T. Leitner, S. Svensson, N. Mårtensson, A. Föhlisch, “Single bunch X-ray pulses on demand from a multi-bunch synchrotron radiation source”, Nature Communications 5, Article number 4010 (2014).

[3] R. Bakker, K. Holldack, P. Kuske, R. Müller, "Orbit Control at BESSY II: Present Performance and Plans", EPAC 2000, Vienna, Austria (2000).

[4] R. Müller, T. Birke, M. Diehn, D. Engel, B. Franksen, R. Görgen, P. Kuske, R. Lange, I. Müller, A. Schälicke, G. Schindhelm, “Fast Orbit Feedback at BESSY-II: Performance and Operational Experiences”, IPAC 2013, Shanghai, China (2013).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

EUROPEAN XFEL CAVITIES PIEZOELECTRIC TUNERS CONTROL RANGE OPTIMIZATION

W. Cichalewski, A.Napieralski, LUT-DMCS, Lodz, Poland
J. Branlard, C. Schmidt, DESY, Hamburg, Germany

Abstract

The piezo based control of the superconducting cavity tuning has been under the development over last years. Automated compensation of Lorentz force detuning of FLASH [1] and European XFEL [2] resonators allowed to maintain cavities in resonance operation even for high acceleration gradients (in range of 30 MV/m). It should be emphasized that cavity resonance control consists of two independent subsystems. First of all the slow motor tuner based system can be used for slow, wide range mechanical tuning (range of hundreds of kHz). Additionally the piezo tuning system allows for fine, dynamic compensation in a range of 1 kHz. In mentioned pulse mode experiments (like FLASH), the piezo regulation budget should be preserved for in-pulse detuning control. In order to maintain optimal cavity frequency adjustment capabilities slow motor tuners should automatically act on the static detuning component at the same time. This paper presents work concerning development, implementation and evaluation of automatic superconducting cavity frequency control towards piezo range optimization. FLASH and XFEL dedicated cavities tuning control experiences are also summarized.

INTRODUCTION

Both FLASH and European XFEL are free electron lasers facilities that build up accelerated beam energy using superconducting linacs. Superconducting cavities are or will be operated in pulse mode with 10 Hz repetition range and field gradients up to 30 MV/m. This work conditions causes extensive Lorentz force based reaction on the structure walls [3]. This cause mechanical deformation in range of few micrometers. For around 1 meter long, 1,3GHz resonant frequency, niobium resonator such a length change induces dynamic detuning modification in the range of couple hundreds of hertz. As cavities loaded quality factor is high (range from 3e6 for FLASH to 4.6e6 for XFEL) such misalignment results in significant accelerating field gradient drop. This have to be compensated by increase of supplying RF power - in order to maintain constant beam energy level. The other possibility to minimize this effect is external mechanical excitation provided by slow (step motors) and fast (piezo elements) cavity tuners.

MOTIVATION

Since fast tuners are foreseen for precise, in-pulse reaction to cavity dimension deviation theirs parameters have to be well recognized and adequately used.

Cavity Tuning Components Characterization

The European XFEL linac installation preparation includes also cryo-module testing phase [4]. Among the others also piezoelectric components are being characterized. One of test goal is to determine coefficient that describes relationship between piezo supply voltage and superconducting resonator frequency shift. The same study allows to verify both piezo elements (actuator and sensor) tuning range and voltage polarity (see Figure 1). Other (LLRF related) test

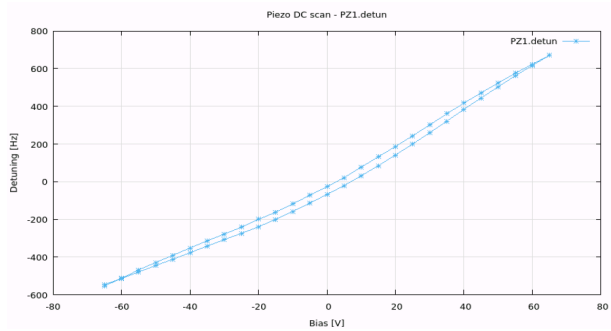


Figure 1: Piezo element tuning range characterization example (XFEL module XM51, cavity 3).

is dedicated to cavity slow motor tuners examination. This devices tuning range is not being verified (about hundreds of kHz). The cavity detuning in function of the motor position change is determined in order to evaluate transfer coefficient (like in previous test). Moreover motor backlash and movement hysteresis is determined. Exemplary measurement results can be found in Figure 2. All measurements results

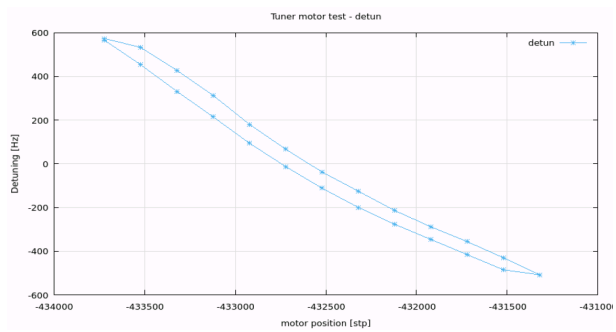


Figure 2: Slow tuner motor behavior characterization (XFEL module XM51, cavity 3).

are stored in dedicated (PostgreSQL [5] based) database. This data will be used during the XFEL linac commissioning for optimal LLRF system configuration. Additionally this information is also vital for described piezo relaxation algorithm configuration.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Detuning Caused by LFD

Superconducting niobium cavity mechanical deformation can be caused (among the others) by high Lorentz force (LFD) acting on the resonator walls in presence of high accelerating field. In case of short pulse operation of TESLA based structures detuning caused by this phenomenon is often described by linear detuning change during RF pulse flat-top phase. Dependency between cavity field gradient and linear detuning is determined also in the scope of modules characterization (see Figure 3). As it is known the LFD

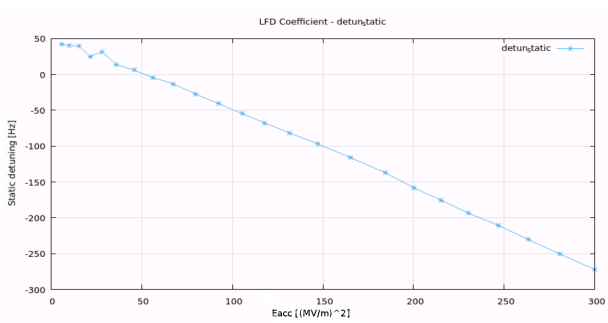


Figure 3: Static detuning change measured in function of cavity gradient.

is proportional to square of cavity gradient. The proportionality coefficient is determined during this tests. Typically this parameter has value in the range of -(0.8-0.9).

Resonators Tuning Aproaches

The cavity detuning is being controlled during RF pulse operation in order to minimize system power consumption. In case of resonator misalignment the accelerating field gradient loss is compensated by the RF power increase. The motor tuner can provide wide range of tuning while its reaction is a time consuming. That is why it can be used for static superconducting structure tuning. The piezo elements provide fast mechanical response (reaction time within RF pulse duration). On the other hand its range is often limited (typically around 1,2kHz). This system can be used both for static and dynamic tuning. Since LFD increases rapidly with accelerating gradient it is highly desired to preserver piezo system frequency shift capabilities to compensate for in-pulse detuning. LFD influence on the cavity operation is minimized by means of piezo components excitation that counteracts resonator dimension change in presence of high accelerating field. In this compensation approach the DC component of piezo voltage excitation is being used for static tuning while AC excitation (sin waveform shape) is used for dynamic misalignment handling. Piezo operation has been automated (for FLASH and AMTF facilities). The algorithm provides both static and dynamic frequency shift reduction

PIEZO RANGE OPTIMIZATION ALGORITHM

The piezo relaxation algorithm main goal is to minimize tuner driver DC stimulus to provide full voltage range for AC signals (to maximize linear tuning boundaries). This requirement can be fulfilled by combined piezo and motor actions. Main idea of this application has been presented in Figure 4.

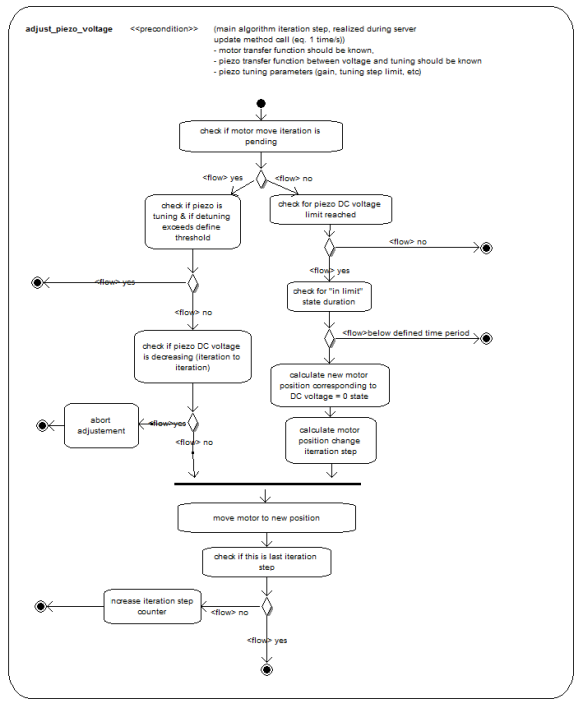


Figure 4: Piezo dynamical range optimization algorihtm overview.

In the scope of this application DC voltage level is monitored. In case its value exceeds predefined threshold (near to the driver saturation) required cavity frequency correction is evaluated. Then the motor adequate position change is determined. New slow tuner settings are calculated according to the piezo tuning resolution and motor steps to cavity frequency modification factor (evaluated during module examination). Overall position change is then divided to several steps iterations. Individual iteration size is determined taking into account piezo automation algorithm ability of resonator tuning. One of the application requirement defines maximal deviation from resonance which is acceptable during relaxation actions. Constant monitoring of cavity behaviour allows for algorithm adjustment or termination in case of operation conditions change or exception occurrence. Application concludes after last iteration step execution. Afterwards system continues piezo operation status monitoring and performs range optimization as soon as DC voltage threshold is being exceeded.

ALGORITHM IMPLEMENTATION

The piezo range optimization algorithm has been developed and implemented as DOOCS server [6]. Application has been realized basing on the DOOCS framework. Piezo and motor operations are relatively slow process in comparison to fast LLRF feedback loop. That is why the server

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

has been prepared as a middle layer process. The server communicates with the LLRF diagnostics server in order to receive cavity detuning calculations readout (see Figure 5). Additionally it connects to the LLRF controller server that provides information about current piezo components drive settings. Since slow motor tuner acts as a actuator in this slow feedback loop also communication with motor management server is provided.

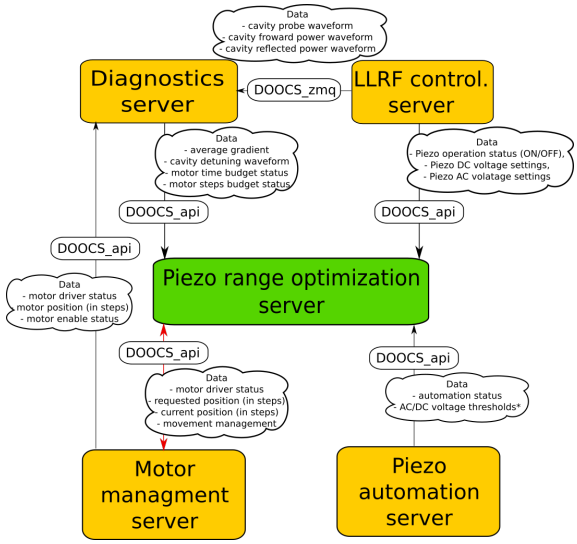


Figure 5: Server interconnections overview.

Server parameters configuration and management has to be performed by RF system users. In order to facilitate server management dedicated GUI (based on JDDD [7]) has been provided (see Figure 6).

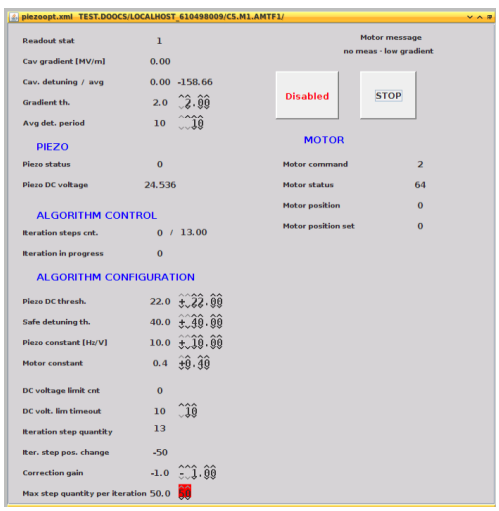


Figure 6: Example of expert GUI.

INITIAL ALGORITHM EVALUATION

Algorithm has been evaluated in the accelerator environment. The FLASH facility is not only FEL research centre but it is also commonly treated as a pilot project to XFEL. That is why it has been perfect place to conduct study concerning future operation automation software component. First test have been performed for single cavity in accelerating module (ACC3). Initially (before algorithm execution) this resonator has been tuned and operated with gradient of 22 MV/m. The piezo automatic tuning algorithm have had been enabled to maintain resonance conditions. Test began with motor tuner position change which corresponded to 300Hz cavity frequency shift. As automatic tuning process caused piezo driver DC voltage increase and threshold violation the algorithm started motor readjustment. In case of FLASH and AMTF study it was observed that application moved step motor to the position corresponding to minimal DC voltage settings. At the same time boundary detuning deviation (configured by user) had been not violated. At the time of tests the algorithm has been extended by some exception handling mechanisms. The most important were:

- range optimization temporary blocking due to the motor steps budget defined for specific period of time,
- range optimization temporary blocking due to the motor operation time budget defined for specific period of time,

AMTF hosts temporally all cryomodules dedicated for XFEL for components examination. That is why piezo range optimization is integrated with the LLRF systems operating in this facility. Scenario of constant cavities gradient increase (during the quench level studies) is the best candidate for piezo range optimization algorithm usage.

CONCLUSION

Piezo based cavity tuning system is widely used during operation of TESLA cavities in high gradient conditions. Tuner range optimization for Lorentz force detuning suppression is a must in case of variable energy settings for the linac. Presented algorithm optimizes fast tuners dynamic range by means of slow motor system readjustment. Cavities characterization provide necessary data for best application configuration. Initial tests performed in accelerator environment proofs algorithm usefulness. That is why the decision has been taken concerning application integration in overall software framework for automatic tuners systems operation.

REFERENCES

[1] Official Free Electron Laser in Hamburg website: `http://flash.desy.de`

[2] Official European X-ray Free Electron Laser website: `http://www.xfel.eu`

[3] T. Schilcher, *Vector Sum Control of Pulsed Accelerating Fields in Lorentz Force Detuned Superconducting Cavities*, (Hamburg: Universitat Hamburg, PhD thesis,1998).

[4] J. Branlard, V. Ayvazyan, M. Grecki, H. Schlarb, C. Schmidt, W. Cichalewski, K. Gnidzinska, A. Piotrowski, K. Przygoda, W. Jalmuzna, "LLRF tests of XFEL cryomodules at AMTF: first experimental results ", THP087, SRF2013, Paris, France (2013).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

[5] PostgreSQL website: https://www.postgresql.org/

[6] Distributed Object Oriented Control System (DOOCS) website: http://doocs.desy.de/

[7] Java DOOCS Data Display website: http://jddd.desy.de/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONTROL SYSTEM OF RF STATIONS FOR NICA BOOSTER

G. A. Fatkin, A. M. Batrakov, I. V. Ilyin, M.Yu. Vasilyev, BINP SB RAS, Novosibirsk, Russia
G. A. Fatkin, NSU, Novosibirsk, Russia

Abstract

NICA (Nuclotron based Ion Collider fAcility) is an accelerator complex, which is being built in JINR (Dubna, Russia). The system described in this paper is controlling the RF stations of Booster, the first element of the NICA complex. Control System consists of two devices: Intellectual Controller and Tester module. The first one is designed for precise measurement of magnetic field, generation of the acceleration frequency in accordance with measured field and control of RF power and pre-amplifiers. Intellectual Controller is a real-time feed-forward system with 40 μs loop time. It is based on ARM microcontroller and bare-metal control programs are used to reach maximum performance. Approaches that were used to achieve maximum performance are elaborated and presented in this paper. The second part of system - Tester is a simulator for tuning and checking the RF stations before start of operations or in absence of real accelerator. The achieved accuracy in chain 'magnetic field' – 'acceleration frequency' is better than $5 \cdot 10^{-5}$. Plans on feedback incorporation to stabilize ion beam behavior via frequency and phase tuning are discussed.

INTRODUCTION

The superconducting accelerator complex NICA (Nuclortron based Ion Collider fAcility) is constructed in JINR, Dubna. Project aims to provide collisions of heavy ion beams in the energy range from 1 to 4.5 GeV/u at the luminosity level of $1 \cdot 10^{27} cm^{-2} \cdot s^{-1}$. The Booster is a cycling accelerator of 197Au32+ ions and serves as one of the elements in the injection chain. It accepts particles with energy 6.2 MeV/u, accelerates them and extracts with energy 600MeV/u to the next acceleration stage - Nuclotron [1].

RF subsystem of the Booster consists of: two cavities, two RF stations, intellectual controller and tester module. RF stations include power amplification cascades and low-voltage electronics. They provide 10 kV acceleration voltage in required frequency range (0.5-5 MHz) on cavity gaps [2]. Primary function of the controller is to generate master frequency depending on the current value of magnetic field. Inaccuracy of frequency setting must not be worse than $\pm 2 \cdot 10^{-5}$.

RF CONTROL STRUCTURE

RF control structure is shown on Fig. 1. Signals from induction sensor and synchronization pulses are passed through the Tester module to Controller. Tester module allows to substitute real signals with imitated scenario or to through-pass these signals. Frequency, phase and cavity voltages are adjusted according to measured magnetic field and acceleration stage. Other technological parameters are measured and set, they are presented at table 1. Both RF Controller and Tester modules use telnet over Ethernet interface for interaction.

Table 1: Input and Output Signals of Controller

Signal	Channels	Rate	Resolution
Output signals			
Master frequency	2	25 kHz	24
V cavity	2	25 kHz	12
I anode	2	100 Hz	10
Input signals			
Field sensor	1	25 kHz	18
V cavity	2	25 kHz	12
Phase difference	1	25 kHz	12
Phase bias	1	25 kHz	12
Frequency bias	1	25 kHz	12
V preamplifier	2	100 Hz	12
I anode	4	1 kHz	12
V rectifier	6	1 Hz	12
V filament	2	1Hz	12
Synchronization	7	N/A	N/A

Controller Structure

Observing table 1 one may conclude that controller module must supply 4 fast (25 kHz) output channels and measure 6 signals with the same rate. Most calculations and data conversions will require floating-point arithmetic. It is also necessary to arrange data interchange between several independent measurement and control devices. E.g. to generate master frequency controller must receive data from measuring ADC, convert, correct and integrate the resulting value to attain field value then calculate frequency, apply necessary corrections, then convert it again and finally send this data to DDS. At the same time other signals must be changed and measured: cavity voltages, cathode currents, preamplifier voltages etc.

Probably betatron and synchrotron oscillations damping would be necessary for Booster. It is very hard to predict feedback parameters before Booster starts operation. Therefore we decided to implement feedback later using separate module which will provide frequency and phase bias analog signals.

Nowadays there are two main approaches for creation of such complex high rate systems: FPGA and high-end microcontrollers. Our reasons for choosing micro-controller were following: the need of floating point arithmetic, complex logic with different modes of operation that probably will require expansion and finally the need to provide TCP/IP interface for the interaction with Booster control system. At

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

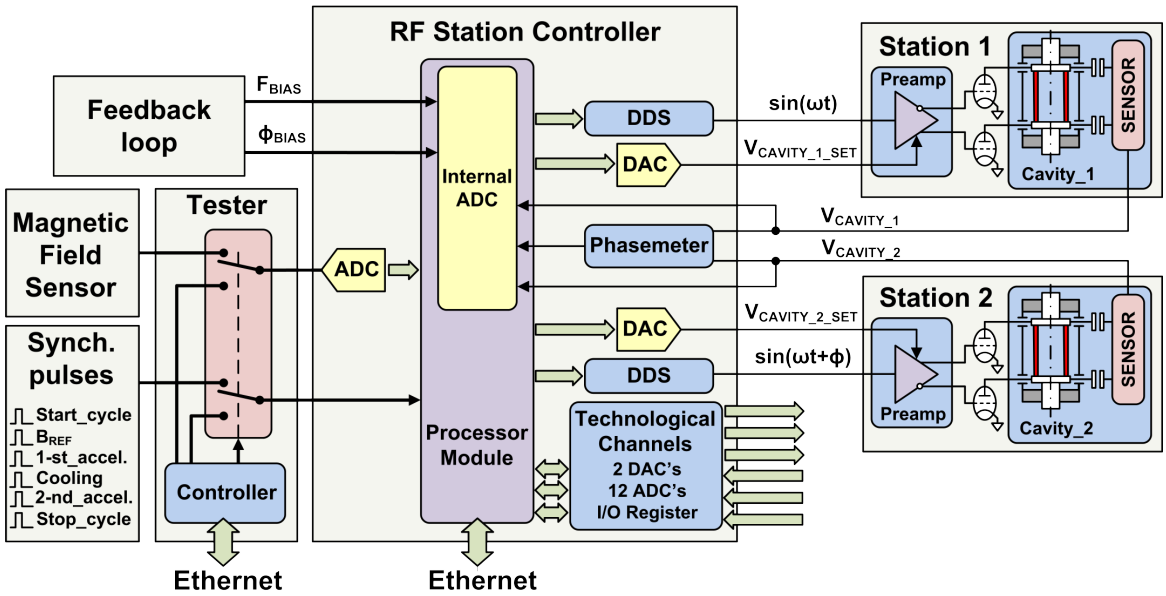


Figure 1: Radio frequency control structure.

the beginning of our work we planned to have Linux with tango-server on-board. Unfortunately we couldn't achive necessary performance with any variation of Linux, so bare-metal approach was used.

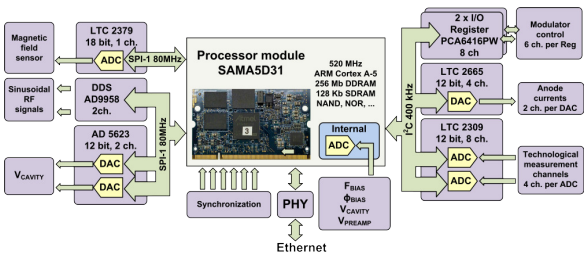


Figure 2: Internal structure of the controller.

Structure of the controller is shown on Fig. 2. We used Atmel SAMA5D31 micro-controller. Availability of on-board 1 MHz 12 bit 8-channel ADC was a significant advantage. Usage of SAMA5D31-CK SO-DIMM board allowed us to simplify PCB design and accelerate production. Availability of different interfaces (I2C, SPI, UART, Ethernet) on this controller was a huge plus. It must be noted though that we had to give up usage of any (even real-time) operational system to provide necessary reaction rate with 40μs cycle. Apart from such bare-metal approach we had to thoroughly optimize our program code. Resulting algorithm is shown on Fig. 3.

Main cycle is realized using three routines. One of them is called "Fast cycle". It is initiated every 40 μs by high-priority timer IRQ. Signal from inductive sensor is measured, derivative of frequency is calculated and passed to DDS. Fast signals (look at Tab. 1) are measured by ADC and generated by DAC's. Another one is "Synchronization" state machine. This state machine is operated by synchronisation I/O that generates low-priority IRQ. Last one is a "Slow cycle" which

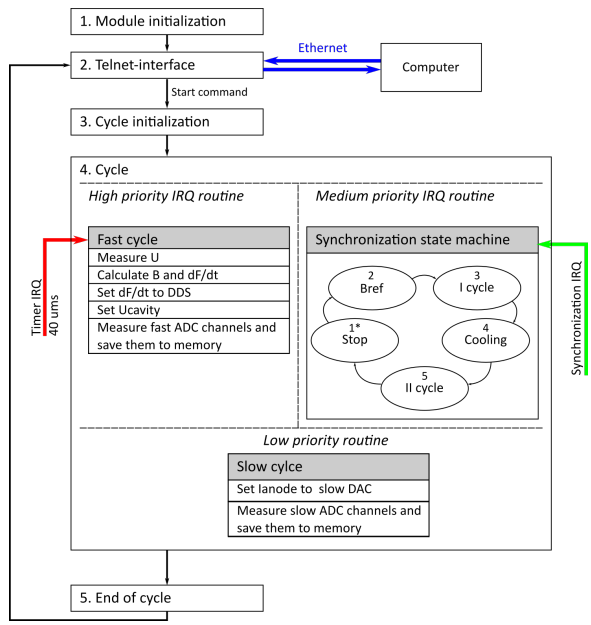


Figure 3: Program structure.

is a loop that measures and generates slow signals. This code works between the other two and thus strict synchronization couldn't be achieved, fortunately it isn't necessary.

Another restriction we had to establish is following: interaction with Booster control system using Ehternet protocol is available only between cycles, while in-cycle Ethernet controller is turned off. Apparently an ideal approach in such case would be the usage of 2-core microcontroller, or SOC with FPGA and ARM kernel e. g. CYCLONE V SOC. But when we started this work, boards like MITYSOM-5CSX on SODIMM were still unavailable. And PCB layout of CYCLONE V SOC is quite problematic in itself and was considered more tedious than program optimization.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Both tester and controller modules are managed over the Ethernet interface using text-based command protocol over telnet. RS-232 interface is provided for reprogramming and debug. Apart from adequate embedded programs, testing software was developed. This software will also serve as a reference for implementation of corresponding modules in NICA Booster control system which is based on TANGO [3].

Low Level Frequency Generation

Main task of the controller is to provide low-level sinusoidal signal with frequency in accordance with the magnetic field. The dependency can be presented in the following form:

$$f(B) = K\frac{aB}{\sqrt{b + CB^2}}, \quad (1)$$

where K - is the harmonic number, B - is magnetic field, and a,b and C are coefficients. We'll use Eq. 1 from now on.

Most common approach for the low-lever RF control systems nowadays is to generate sinusoidal signal using DDS (direct digital synthesis). Both specialized IC's and FPGA with DAC implementations are used. We decided to use latter because our system is based on microcontroller. To alleviate possible difficulties with inter-channel synchronisation two-channel AD9958 chip was chosen.

Actual signal from the inductive sensor is proportional to the change of magnetic flux over time. If the sensor is stationary, then the area is constant and signal U is proportional to the derivative of the magnetic field:

$$U = \frac{1}{\alpha}\frac{dB}{dt}, \quad (2)$$

where $\frac{1}{\alpha}$ is a proportionality constant.

Therefore to calculate B and for Eq. 1, one has to measure the following integral:

$$B(t) = \alpha \int_0^t U dt. \quad (3)$$

We can use several mehods to generate master-frequency that varies according to the law Eq. 1. First of them is not to measure integral at all and work by a table instead. Actually if you have perfectly repeatable magnetic field source and perfect synchronisation, you can calculate values of $f(t)$ beforehead, put them in a table and generate them using DDS at appropriate moments of time. We can note that this approach doesn't require measurement of magnetic field at all. The downside of this approach is the complexity of creating sufficiently stable magnet power supply and providing necessary synchronisation accuracy [4].

Rather wide-spread method to measure integral at Eq. 3 is based on the use of voltage to frequency converters (VFC) followed by counter [5].

Other way to determine integral is to measure inductive sensor output voltage (i.e. derivative dB/dt) with the help of fast, precision ADC and then to sum the ADC samples in order to calculate the integral value. This method may be called "direct digital integration" [6]. Usage of the fast precision ADC allowed us to use a specific method of frequency tuning to achieve higher accuracy. This method is described below.

Our Method of Frequency Tuning

As described previously, one of the approaches is to acquire field value, substitute it to the Eq. 1, integrate it, apply necessary corrections (e.g. feedback signals, etc) and set calculated frequency value to DDS generator. This was the approach that we realized in our first controller prototype.

The main consideration on this approach (besides accuracy of measurement for which appropriate means were taken) is it's "stepping" nature, which introduces a quantization error. We can estimate this error in the following way. Highest frequency change rate is 2.25 MHz/sec. With practically achievable cycle time $\Delta T = 40\mu s$ we get $25Hz$ quantization error. That is around $5 \cdot 10^{-5}$ at the starting frequency 500 kHz.

Now we would like to describe the method which allows to simplify calculations, lower quantization error and loosen cycle time requirements. This method relies on the ability of DDS to perform linear sweep of frequency with variable sweep rate over time. This ability is implemented as shown at Fig. 4. Internal down δt counter operates at DDS clock (25 MHz in our case), when it reaches zero a value written in δf register is added to current frequency register and counter is reset to initial value. Current frequency register is used by sinusoidal waveform generator to provide output signal. Essentially this ability means that we are able to provide DDS with frequency derivative $\frac{df}{dt}$ modifying both difference at each step δf and stepping rate δt. If δt is lower than the measurement/integration cycle time ΔT, we get an advantage in precision.

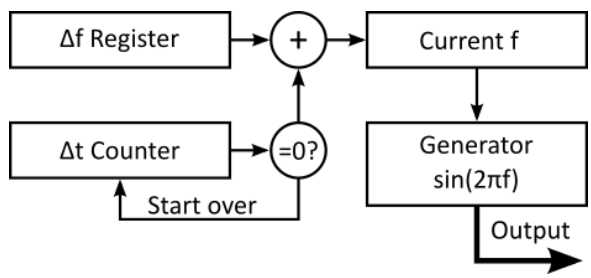


Figure 4: Simplified DDS frequency sweep implementation.

The illustration to this fact is provided on Fig. 5. $U(t)$ is a typical signal from induction sensor, $B(t)$ is an appropriate integral. Let's assume that our cycle period is ΔT, therefore individual readings are taken at moments of time t_0,t_1,...t_i, with $\Delta T = t_i - t_{i-1}$. Graph $f(t)$ represents the approach when on each step we measure $U(t)$, calculate $f(\alpha \int U(t_i))$ and provide DDS with new frequency value. Graph $f_{sweep}(t)$ represents an approach with derivatives which will be described in detail below. Apparently this approach allows to significantly lower quantization error.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

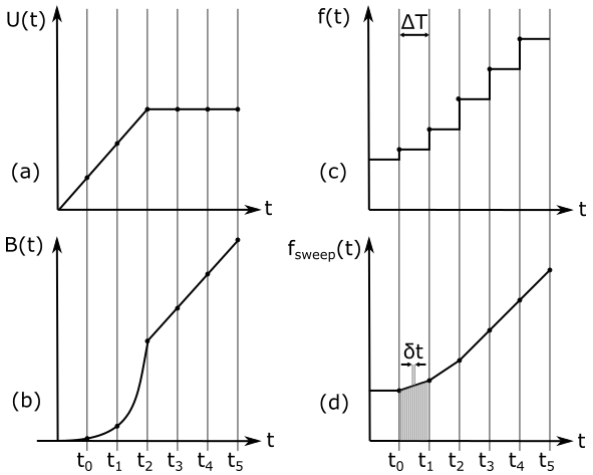


Figure 5: Demonstration of sweep approach superiority. (a) – inductive sensor signal U(t); (b) – magnetic fiel B(t); (c) – frequency form for "stepping" approach f(t); (d) – frequency form for "sweep" approach.

To calculate $\frac{df}{dt}$ we start from the Eq. 1 and take it's derivative over dB:

$$\frac{df}{dB} = \frac{Ka}{\sqrt{b + CB^2}} - \frac{KaCB^2}{(b + CB^2)^{3/2}}, \quad (4)$$

after examination of Eq. 4 taking Equations 1 and 3 into account we get the following:

$$\frac{df}{dt} = \frac{df}{dB}\frac{dB}{dt} = \frac{f}{B}(1 - \frac{C}{K^2a^2}f^2)\alpha U. \quad (5)$$

For the actual calculations following discrete form is used:

$$B_i = B_{i-1} + \alpha U \Delta T \quad (6)$$

$$\delta f_i = \frac{f_{i-1}}{B_{i-1}}(1 - \frac{C}{a^2K^2}f_{i-1}^2) \quad (7)$$

$$f_i = f_{i-1} + \delta f_i \delta t \quad (8)$$

Notable feature of these equations is the absence of square root operation. Values f_0 and B_0 could be calculated before the start of an acceleration cycle.

Currently we are using the described method with $\delta t = 4\mu s$, therefore achieving theoretical frequency error level of $5 \cdot 10^{-6}$ in the worst case. Error measured using a frequency meter is $1.4 \cdot 10^{-5}$ and is determined by different measurement and synchronisation inaccuracies.

TESTER

Tester Module consists of digital control unit and analog Front-End. Analog Front-End is based on DAC8831 IC with 16 bit resolution. The core of digital control unit is LPC2478 processor, which provides DAC control of analog Front-End, synchronization pulses generation, multiplexer control and Ethernet interface for inter-operation with Booster control system. Particular attention was paid to analog Front-End circuit design. The block diagram of analog Front-End is shown in Fig. 6. Front-End is a combination of two OPA2277 operational amplifiers. First one provides gain and offset for DAC output signal. Second generates ±1 V signal on 120 Ohm input resistance of the Controller module. Using offset adjustment circuit, high precision resistors 0.1% with $5ppm/°C$ temperature coefficient and precision voltage references (LTC6655) allow to achieve high accuracy of the magnetic field derivative. Resulting noise integral of magnetic field simulation is less then 10^-5.

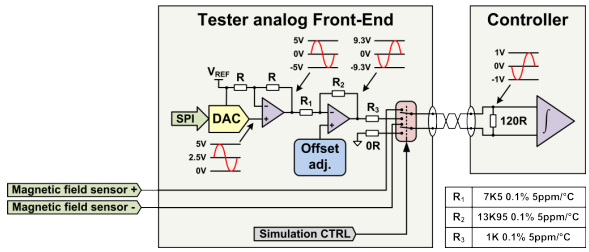


Figure 6: Tester structure.

Ethernet interface for inter-operation with booster control system.

CONCLUSION

Controller and Tester modules were designed manufactured and tested in working conditions with RF stations generating high voltage. Usage of SAMA5D31-CK board allowed to significantly reduce prototyping time. New approach for frequency tuning was devised. Overall measured innaccuracy using frequency meter is better than $1.4 \cdot 10^{-5}$. After NICA Booster comissioning beam feedback module would be produced and incorporated to the LLRF control system. All devices provide simple telnet interface for integration in NICA control system.

REFERENCES

[1] Eliseyev A.V., Meshkov I.N., et. al., "Longtitudal Dynamics of AU-197**(32+) and AU-197**(79+) Beams in the NICA Collider Injection Chain", PEPAN Letters, vol. 78, No7, 2010, p. 774-780.

[2] G. Ya. Kurkin, A. M. Batrakov, et al., "RF System of the Booster of NICA Facility", in Proc. of RuPAC2014, Obninsk, October 2014, TUCB02 .

[3] E. V. Gorbachev, A. Kirichenko, et al., "Upgrade of the NUCLOTRON Injection Control and Diagnostic System" in Proc. of ICALEPCS2013, San Francisco, October 2013, THPPC048.

[4] V. S. Arbuzov, A. A. Bushuev, et al. "Accelerating RF Station for HIRFL-CSR, Lanzhou, China", in Proc. of RuPAC XIX, Dubna, October 2004, TUGP08.

[5] E. Feldmeier, T. Haberer, "Development of a high precision integrator for analog signals to measure magnetic fields in real-time", in Proc. of IPAC2013, Shanghai, May 2013, MOPWA001.

[6] Batrakov A.M., Il'yin I.V., Pavlenko A.V., "Precision digital signal integrators with accurate synchronization", Optoelectronics, Instrumentation and Data Processing, V.51, No1, p. 51-57.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

TPS BOOSTER TUNE MEASURENET SYSTEM

P. C. Chiu, K. H. Hu, H. P. Hsueh, Y. S. Cheng, Demi Lee, K. T. Hsu
NSRRC, Hsinchu 30076, Taiwan

Abstract

The Taiwan Photon Source (TPS) is a state-of-the-art synchrotron radiation facility featuring ultra-high photon brightness [1]. Its Booster has 6 FODO cells which include 7 BD dipoles with 1.6 m long and 2 BH dipoles with 0.8 m long in each cell. After magnetization of stainless steel vacuum chamber of the booster were identified and then dismantled, annealed, and re-installed, the electron beam energy of the TPS booster ring has ramped to 3 GeV in a week. The booster tune correction during ramping is one of the main reasons why the booster commissioning progress so fast. This report will be summarized the booster tune monitor system.

INTRODUCTION

The TPS is a state-of-the-art synchrotron radiation facility featuring ultra-high photon brightness with low emittance. The TPS accelerator complex consists of a 150 MeV S-band linac, linac to booster transfer line (LTB), 0.15–3 GeV booster synchrotron, booster to storage ring transfer line (BTS), and 3 GeV storage ring. The booster has 6 FODO cells and its circumference is 496.8 meter; the storage ring's circumference is 518.4 meters with 24 DBA lattice and 6-fold symmetry. The booster and the storage ring share the same tunnel in a concentric fashion. During 4 years of construction period, civil constructions had been completed in early 2013. The accelerator installation then had taken another one year.

At September 2014, booster tune measurement commissioning had committed with beam commissioning. First-turn was easily obtained by beam steering and then multi-turn was also observed after optimization of Linac and transfer line. However, the beam could not be stored. After some hardware improvement such as power supply tuning, chamber and magnet re-alignment, kicker and septum improving and etc., the key setback was finally found on November 12: the pipes had a relative high permeability (ranging from 1.2 to 2.0) induced from the lack of proper annealing process. These un-annealed chambers were uninstalled and treated in vacuum oven up to 1050 ℃, and then re-installed. Booster then had stored beam in DC mode and ramped to 3 GeV successfully at December 16 2014 after several times of tune compensation. The tune measurement system provides precise tune measurement to correct tune variation during energy ramping and avoid across the resonant line which would cause a lot of beam loss. This report will summarize the infrastructure of booster tune measurement system measurement results and tune compensation during TPS booster commissioning.

TUNE MONITOR SETUP

Originally during booster commissioning beginning in Sep. 2014, the magnetic shakers where two multi-turn coils are mounted on vacuum chamber in horizontal and vertical plane are applied to excite beam. The kickers with 50Ω terminated load have calibration factor of 3 mG/A and are driven by a 50W amplifiers. Later, the stripline electrodes are adopted to replace magnet shakers on the booster synchrotron considering more power strength to excite beam. The TBT data provided by BPM electronics would be acquired to extract tune by FFT. Agilent arbitrary signal generator would provide band-limited, strength-adjustable excite signal. The functional block diagram of this new tune monitor system is shown in Fig.1. Figure 2 shows the stripline kicker installed in June 2015 to provide more efficient power to excite beam.

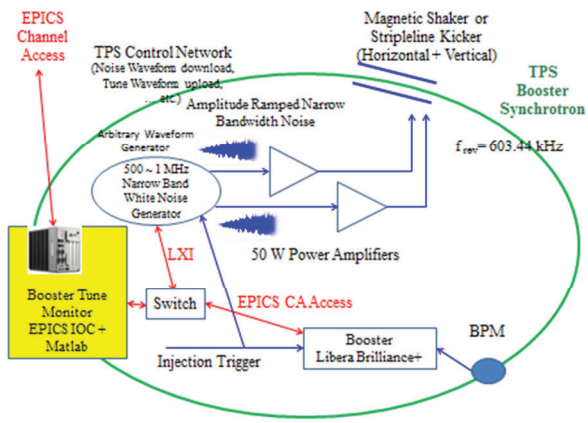


Figure 1: Functional block diagram of the tune monitor for TPS booster.

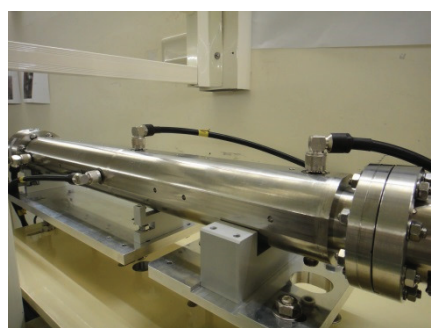


Figure 2: Stripline kickers installed in June 2015.

Figure 3 shows the booster tune measurement GUI. The spectrogram of the horizontal and vertical of BPM turn-by-turn data could identify tune variation clearly. Peak identification from the spectrogram could extract the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

varying tunes during one ramping cycle. Excitation waveform would be generated from white noise with limited bandwidth and strength could be adjusted as energy increased. The generated waveform would be load to signal generator via EPICS channel access.

Besides, the reproduction of ramping power-supplies stability was critical for it could affect the working point and injection efficiency at 150 MeV at the beginning. The real-time injection tune display is also provided for monitoring as Fig. 4 shown.

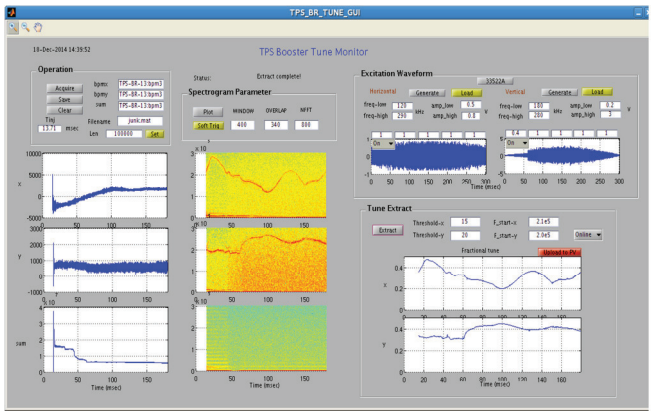


Figure 3: Tune measurement GUI during Booster ramping. Excitation waveform could be set to the proper strength and spectrum.

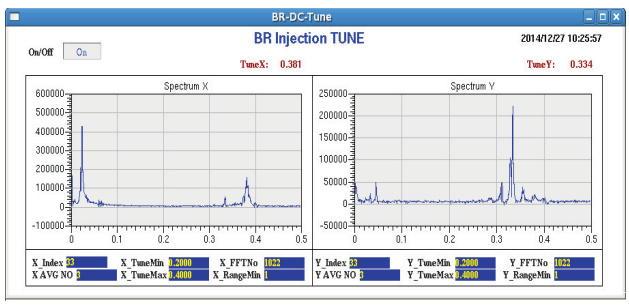


Figure 4: Injection tune extracted from BPM TBT data at injection time.

TUNE MEASUREMENT AND COMPENSATION DURING BOOSTER COMMISSIONING

After chamber demagnetized, we had a stored beam soon after the RF system was activated. Then the first test of energy ramping at AC mode was found that the beam loss occurred when beam ramping to 2.3 GeV as Fig. 4 shown because the vertical tune across 1/3 the resonance line. After tunes compensation scheme applied during ramping, a 3 GeV beam was attained in two days.

The tune variation was as large as 0.22 for horizontal and 0.1 for vertical as Fig. 6 (a) shown. The working points across the resonance line would cause beam loss and decrease the injection efficiency. It could be inferred that reference waveform generated from the measured I-B table provide by the magnet lab could be deviated from the actual machine. To improve the injection efficiency, Q1 and Q2 are chosen and measured the tune response for tune compensation. After tune correction, horizontal and vertical tune variations are both reduced to 0.05 as Fig. 6(b) shown.

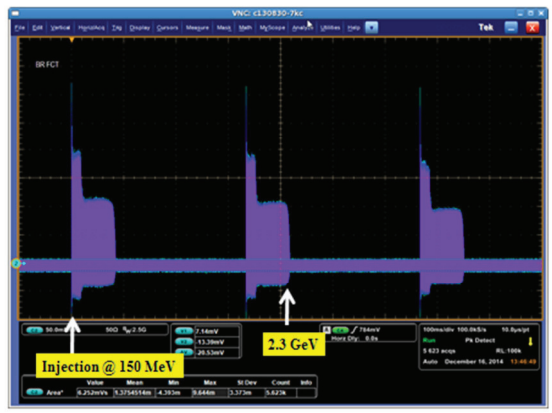
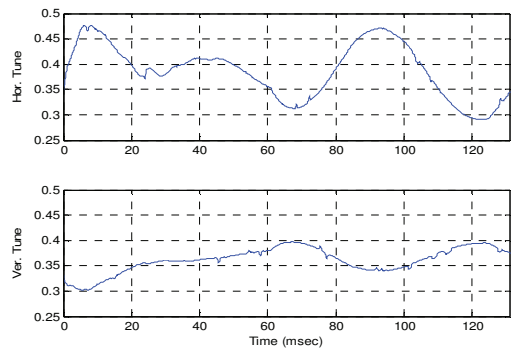
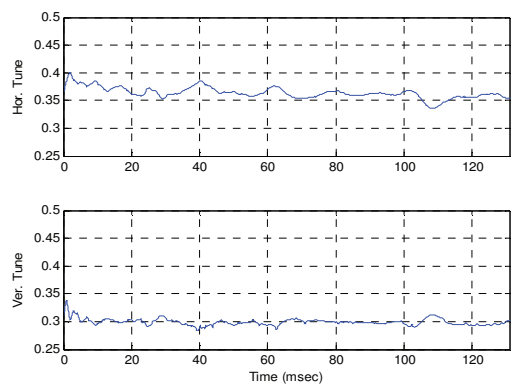


Figure 5: The beam loss occurred when beam ramping to 2.3 GeV observed on the first day of booster AC mode test.



(a)



(b)

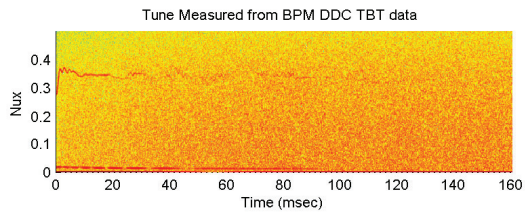
Figure 6: (a) Tune variation during ramping before tune compensation. Horizontal tune variation was as large as 0.22; vertical tune variation was 0.1. (b) Horizontal and vertical tune variation are both reduced to 0.05 after tune compensation.

ISBN 978-3-95450-148-9

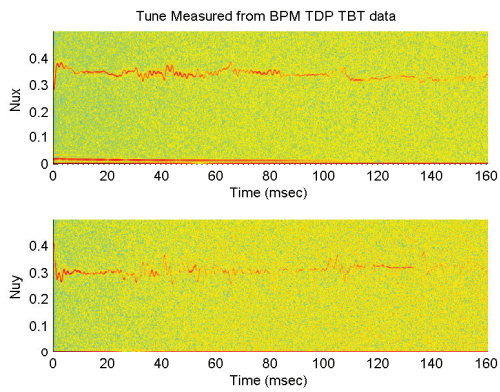
Copyright © 2015 CC-BY-3.0 and by the respective authors

BPM TDP AND DDC TBT DATA FOR TUNE MEASUREMENT

DDC (Digital Down Converter) and TDP (Time Domain Processing) Turn-by-turn data are both provided by BPM electronics and the resolution could achieve around 150 um at 0.5 mA. To use TDP properly, phase offset and mask window should be set correctly [1]. Both of these two TBT data chains could be applied to extract tune. Nevertheless, compared to DDC, TDP could well resolve beam loss status and tune extraction especially in lower beam current or smaller beam motion. It is due to clear and no smear TBT data as Fig. 7 which shows the spectrogram of DDC and TDP data respectively.



(a)



(b)

Figure 7: (a) Tune extracted from BPM DDC TBT data during booster ramping (b) Tune extracted from BPM TDP TBT data. TDP had better signal qualities than DDC due to its clear and no smear TBT data.

BOOSTER TUNE STABILITY

Initially, the booster beam current was not stable. It was found that one of the reasons was the stability/reproducibility of booster main power supply not good enough, especially at lower current. Figure 8 shows tune shifts for different injection. At lower energy, the tune reproducibility is worse than high energy. It is because quadruple to dipole relative err would affect tune and when tune drift to the improper working point, it would have lower beam current. Figure 8 shows the variations of QF to Dipole err, injection tune and beam current. After stability of power supply improved from 0.5% to 0.2%, stability of beam current was also improved [2].

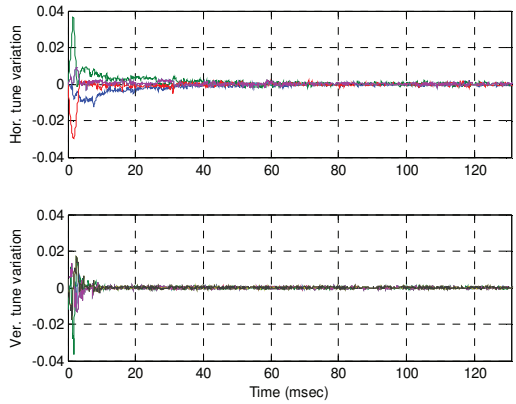


Figure 8: The tune reproducibility observed from 10 times injection. The reproducibility becomes better as energy increased for the reproducibility of power supply also becomes better as current increased.

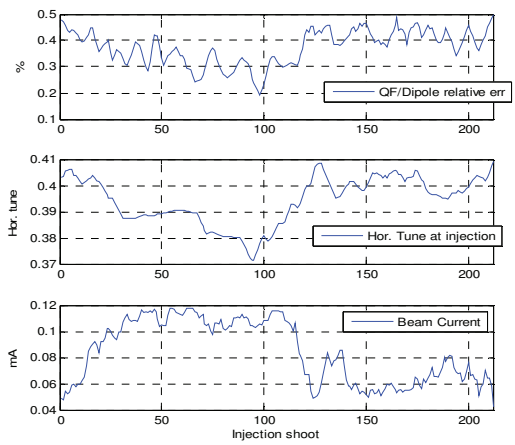


Figure 9: The variations of QF to Dipole relative err, tune and beam current.

SUMMARY

In this report, we summarized the TPS booster tune measurement system and perform tune compensation to reduce tune variation and improve injection efficiency.

REFERENCES

[1] TPS Design Handbook, version 16, June 2009.
[2] H. J. Tsai et al., "Hardware Improvements and Beam Commissioning of the Booster Ring in Taiwan Photon Source", IPAC'15, Richmond, VA, USA, May 2015.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INTEGRATING THE MEASURING SYSTEM OF VIBRATION AND BEAM POSITION MONITOR TO STUDY THE BEAM STABILITY

C. H. Huang, C.Y. Liao, Y.S. Cheng, P. C. Chiu, K. H. Hu, K. T. Hsu
NSRRC, Hsinchu 30076, Taiwan

Abstract

For a low emittance, third-generation light source such as Taiwan Photon Source (TPS), beam orbit motion needs to be controlled within submicron for obtaining a high quality light source. Magnets vibration especially quadruples will be one of the main sources to destroy the beam stability. In order to study the relationship between vibration and beam motion, a synchronous data acquisition system which integrates the measurement of vibration and beam position monitoring system is highly desirable. For a larger vibration such as earthquakes are also deleterious to beam stability or even make the beam trip due to the quench of superconducting RF cavity. A data acquisition system integrated with an earthquake P-wave detector is also quite necessary to show and archive the data on the control system. First, the data acquisition systems of vibration and earthquake measurement system are summarized in this report. The preliminary study of the relationship between the beam motion and magnets vibration will also be presented here.

INTRODUTION

For a low emittance, third-generation light source such as TPS, it imposes a stringent requirement for the orbit stability. To obtain a high quality light source, the beam orbit motion needs to be controlled at least within 0.5 μm. There are many sources [1] which may destroy the beam stability such as vibration, ill function of power supply, septum/kicker field leakage, etc. The ground vibration and the technical noise such as vacuum pumps and cooling water flow will cause the magnet and vacuum chamber vibration. It leads to the distortion of the close orbit [2]. The dominant effect for the beam motion caused by magnet vibration is produced by the quadrupole and it introduces a kick angle to the beam as the quadrupole has an offset to the beam due to the vibration.

To achieve submicron stability, various efforts and studies should be continuously performed such as power supply performance, RF system performance, cooling water, mechanical vibration, orbit feedback system and bunch-by-bunch feedbacks. Besides, it is also important to identify various sources and to minimize their effects to the beam instability. In this paper, the data acquisition system for measuring the vibration and beam position is introduced first. The setup of beam position monitoring and earthquake monitoring system are described in the third and fourth parts. In the fifth and sixth sections, the preliminary results to identify the vibration caused by turbo molecular pump (TMP) system and to study the beam stability are shown. Finally is the short summary.

VIBRATION DETECTOR AND DATA ACQUISITION SYSTEM

Low noise three-component seismometers, LE-3Dlite Mark II, with frequency range 1- 100 Hz is used in this study. The data acquisition unit (Data Translation DT8837), which is complied with LXI class C standard, provides Ethernet accesses via SCPI command to acquire data. Multiple DT8837s are synchronized by wired trigger bus (WTB) interface. To extend length limit of WTB cable, an in-house made small interface adapter from RJ-45 to Micro D is installed at the WTB connector of the DT8837 side. It allows unshielded twisted pair (UTP) cables to replace WTB cables and to send the trigger, sync and clock single from the timing system adapter to DT8837 more than 100 m. Coherent data acquisition can also be achieved by the aid of global timing system for long rang or by LXI WTB cable for short range application. A Matlab script can be running in the server for the long-term measurement or in the laptop for the short-term measurement, shown in Fig. 1. The synchronous data acquisition between vibration measurement and beam position monitoring system can also be through the timing system.

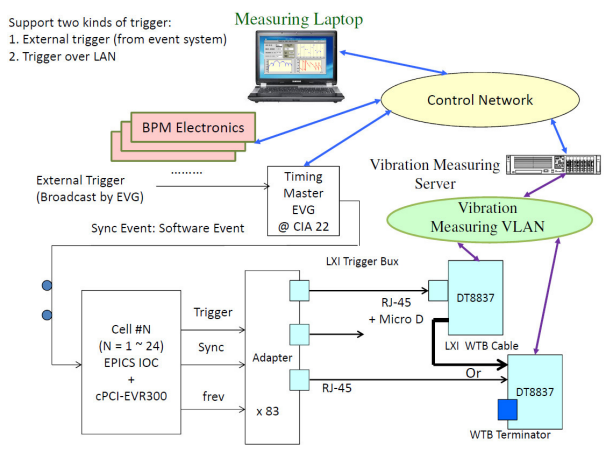


Figure 1: Configuration of vibration measuring system integrated with beam position monitoring system.

BEAM POSITION MONITORING SYSTEM

The TPS storage ring is divided into 24 cells and 7 beam position monitors (BPMs) are installed in each cell.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Another six BPMs are installed at three long straight lines with inserting devices for the local measurement. Two kinds of BPM are installed in the storage ring: one is standard button BPM shape at arc section; the other is primary BPM shape with racetrack at the insertion device straight.

The BPM electronics provide several data types for various application. Analog-to-digital converter (ADC) and turn-by-turn (TBT) data are acquired for first turn and betatron tune analysis; 10 Hz slow acquisition data (SA) is for DC average orbit and 10 kHz fast acquisition data (FA) could be applied for fast orbit feedback application and beam stability analysis. An experimental physics and industrial control system (EPICS) input/output controller (IOC) is embedded in the electronics platform to control, monitor and configure the BPM system in the control network [3]. Here, FA is used to be integrated with vibration measurement system and to study beam stability.

EARTHQUAKE MONITORING SYSTEM

In order to monitor the earthquake information which is happened in NSRRC campus and to study the beam stability as an earthquake is happened, an earthquake P-wave detector which equips with MEMS accelerometers and an ADC with 16-bit output resolution is installed in the TPS building. A complied Matlab program is used to communicate with the earthquake detector through Modus TCP protocol, and transfer the useful information into the created IOC through the soft process variables (PVs), shown in Fig. 2. The EPICS IOC publish PVs into the TPS control network and earthquake information detected by the detector can be shown on the console on line.

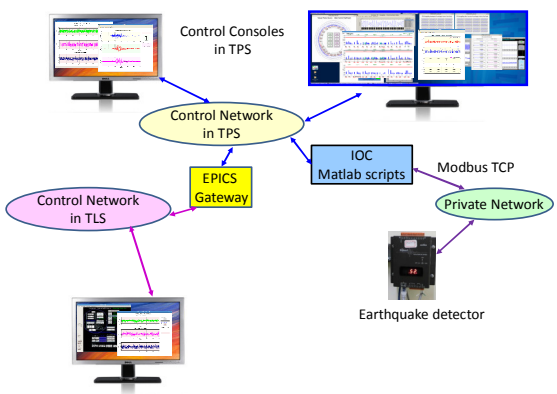


Figure 2: The configuration of the earthquake monitoring system.

As detecting an earthquake, the number of event flag in the data stream will be changed. When the Matlab program detects this flag, the earthquake information with 30 seconds before trigger and 90 seconds after trigger is recorded in the archived file for further analysis. A graphical user interface is used to show the events in the control console, shown in Fig. 3. To show the earthquake information in the Taiwan Light Source (TLS) control system, an EPICS gateway is used to transfer the earthquake related PVs from TPS control network into the TLS control network. Therefore, the earthquake monitoring system can also be launched in the TLS control system.

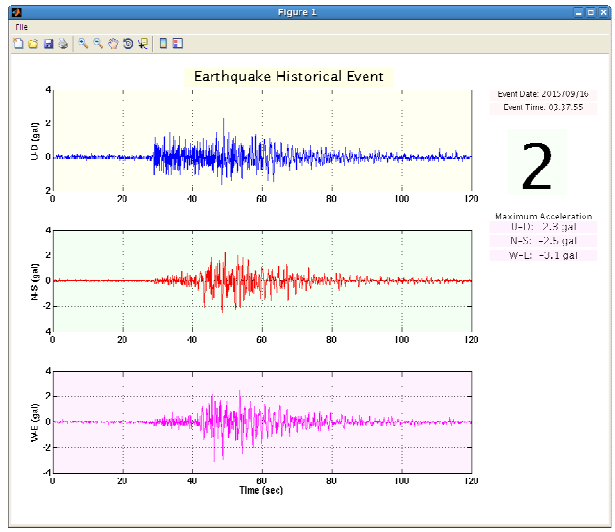


Figure 3: The earthquake historical event loaded from the archived files.

VIBRATION CAUSED BY TURBO MOLECULAR PUMP SYSTEM

In the first phase commissioning during December 2014 to March 2015 [4], an obvious beam motion around 29 Hz is observed. A sharp peak around 29 Hz can also be found in the magnet vibration [5]. As the TMP system, which contains a TMP, a dry pump and a mechanical pump, is turn off, the beam motion and magnets vibration around 29 Hz is highly eliminated. In order to investigate the vibration propagation path, a serious study is performed during the long shutdown from April to August. It was identified that the vibration source is from the mechanical pump, shown in Fig. 4. As a seismometer is put near the mechanical pump, a sharp peak in Fig. 5 can be found in the displacement spectrum.

The electric mains frequency in Taiwan is 60 Hz. All of these mechanical pumps are driven by induction motor. There are four poles in this motor so the synchronous speed should be 30 Hz [6]. The slip of rotor and stator makes the revolution frequency less than 30 Hz and the revolution frequency depends on the load of the motor. Therefore, the measured vibration frequencies of magnets in each girder are slightly different. The TMP system is the most pumping efficient for high gas load and was planned to operate during the early commissioning and the first year operation. In order eliminate the destroying of beam stability, all the TMP systems are turned off in the later period of beam commissioning.

The propagating path from the mechanical pump to the TMP is by the vacuum pipe connected between TMP and dry pump. The TMP is combined with the vacuum chamber of the storage ring so vibration of mechanical

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

pump would also propagate into the vacuum chamber. There are no direct contact between magnets and vacuum chambers so the vibration propagating from the vacuum chambers to the magnets should be through the girder.

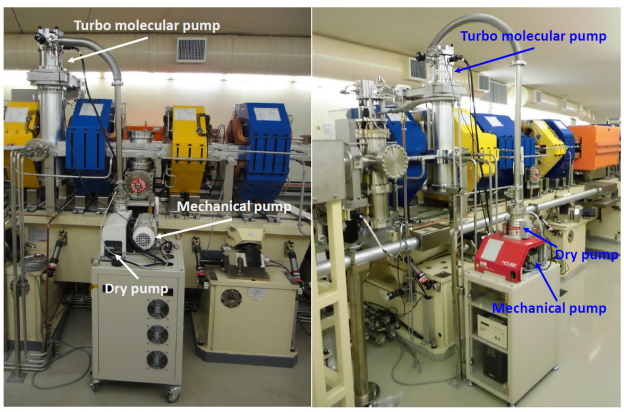


Figure 4: The setup of the turbo molecular pump systems.

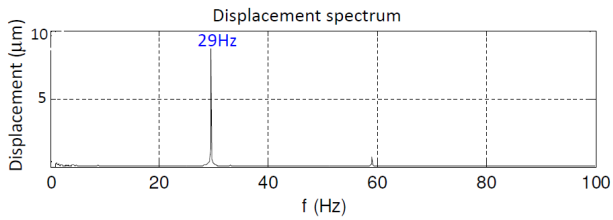


Figure 5: The displacement spectrum near the mechanical pump.

As comparing the vibration of quadruples in the third cell, it can be observed that the peak around 29 Hz is disappeared as the TMP system is turned off. When the cooling water of magnets and vacuum chamber is turned off, the vibration around 20 - 40 Hz is greatly reduced, shown in Fig. 6. Therefore, the source of vibration around this frequency should be the water flow. However, the measuring results in the magnets is not only corresponding to the amplitude of the vibration sources but also the transform function or resonance frequency of supporting mechanical structures such as the girder so advanced study is necessary to figure out a complete conclusion.

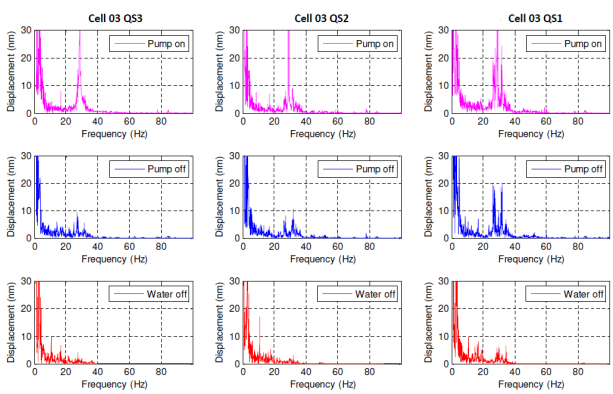


Figure 6: The vertical displacement spectra of the quadruples in the third girder of the cell #3 as TMP is turned on, TMP is turned off, and cooling water is turned off.

PRELIMINARY ANALYSIS OF BEAM STABILITY

The phase II commissioning with SRF cavities and inserting devices starts at September, 2015. All the TMP systems are turned off. In the beginning of the commissioning, we found there is a 3 Hz oscillation in horizontal axis plane from the 10 Hz BPM data. As the ramping the booster power supply is turned off, the 3 Hz peak is disappear, shown in Fig. 7. From the power spectral density (PSD) of the 4th BPM in the cell #5, the variation contribution is mainly from 20 - 50 Hz in the horizontal axis plane

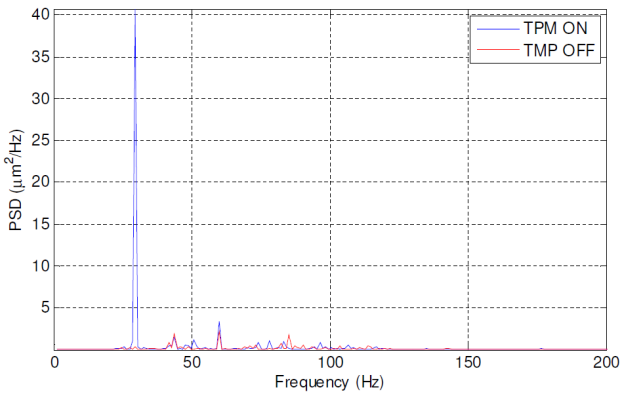


Figure 7: Power spectral density (PSD) in horizontal axis plane of the 4th BPM in the cell #5 as the ramping power supply (PS) in the boostering ring (BR) is turned on or turned off.

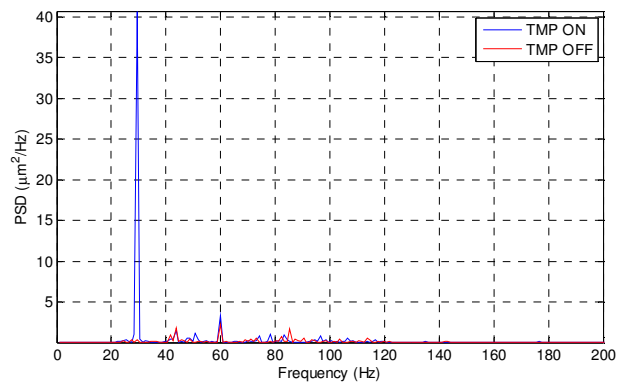


Figure 8: The comparison of vertical beam motion in the 5th BPM of cell #7 as the TMP in the 3rd girder of the cell #7 is turned on and turned off. The beam motion around 29 Hz disappears as TMP is turned off.

In order to test the data acquisition between vibration and BPM system, the TMP system in the 3rd girder of cell #7 is turned on. Comparing with the results as all TMP systems are turned off, a strong beam motion can be observed around 29 Hz in the vertical direction, shown in Fig. 8. Note that the beam motion around 29 Hz could be

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

observed in all of the BPMs with various amplitudes due to the different beta function and phase advance. As measuring the coherence between the quadruple vibration and beam motion, the coherence around 29 Hz is almost 1 and the coherence in the other frequencies is lower, shown in Fig. 9.

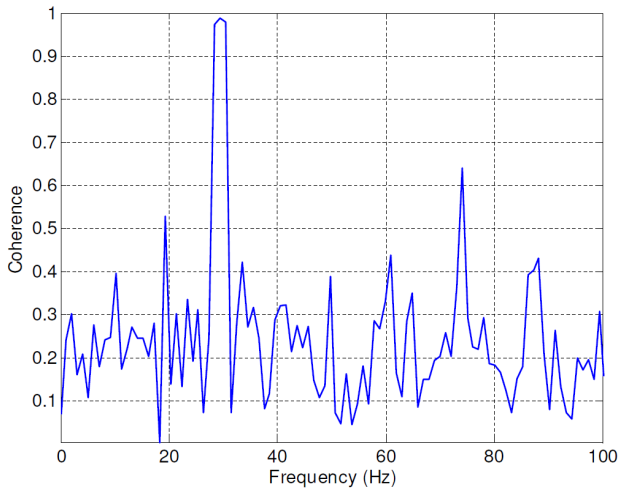


Figure 9: The vertical coherence between the 8th quadruple and 5th BPM in the cell #7.

SUMMARY

The phase II commissioning with SRF cavities and inserting devices is in process. Data acquisition which integrates with the beam position monitoring system and vibration measurement is setup for studying the relationship between beam stability and vibration. The earthquake monitoring system is also setup not only to monitor the earthquake but also try to investigate the relation between the earthquake and beam motion. By the vibration measurement and beam position monitoring system, the source and propagation path of the beam motion around the 29 Hz is identified and the strong coherence is found. However, the beam motion around 20 – 50 Hz is still not low enough. A lot of efforts should be done to increase the beam stability.

REFERENCE

[1] P. C. Chiu et al., "Application of BPM Data to Locate Noise Source", IPAC 2010, Kyoto, Japan.
[2] S. Matsui et al., "Orbit Fluctuation of Electron Beam due to Vibration of Vacuum Chamber in Quadrupole Magnets", Jpn. J. Appl. Phys. 42 (2003), L338-L341.
[3] P. C. Chiu et al., "Commissioning of BPM System for the TPS Project", IBIC 2015, Melbourne, Australia.
[4] C. C. Kuo, et al., "Commissioning of the Taiwan Photon Source", IPAC 2015, Richmond, USA.
[5] C. H. Huang et al., "Vibration Measurement of the Magnets in the Storage Ring of TPS", IPAC 2015, Richmond, USA.
[6] https://en.wikipedia.org/wiki/Induction_motor

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONTROL OF FAST-PULSED POWER CONVERTERS AT CERN USING A FUNCTION GENERATOR CONTROLLER

R. Murillo-Garcia, Q. King, M. Magrans de Abril, CERN, Geneva, Switzerland

Abstract

The electrical power converter group at CERN is responsible for the design of fast-pulsed power converters. These generate a flat-top pulse of the order of a few milliseconds. Control of these power converters is orchestrated by an embedded computer, known as the Function Generator/Controller (FGC). The FGC is the main component in the so-called RegFGC3 chassis, which also houses a variety of purpose-built cards. Ensuring the generation of the pulse at a precise moment, typically when the beam passes, is paramount to the correct behaviour of the accelerator. To that end, the timing distribution and posterior handling by the FGC must be well defined. Also important is the ability to provide operational feedback, and to configure the FGC, the converter, and the pulse characteristics. This paper presents an overview of the system architecture as well as the results obtained during the commissioning of this control solution in CERN's new Linac4.

INTRODUCTION

CERN has nine accelerators, numerous experiments and various test areas. One commonality of such infrastructure is the need of a power system to energize the magnets (dipoles, quadruples, septums, etc.) and the various radio frequency cavities. In total, CERN has around 5,600 power converters, accounting for a typical consumption during exploitation of 1.2 TWh per year [1].

From an operational point of view, power converters fall into three categories:

- Cycling: the field, current or voltage reference value can synchronously be modified on a cycle-by-cycle basis. At CERN each cycle is a multiple of a basic period, which has a fixed duration of 1.2 seconds.
- DC: the reference is asynchronously modified as required during operation to reach a desired field, current or voltage.
- Aperiodic: operators trigger aperiodic cycles to ramp the current to maintain the required beam stability. This is the case of the LHC.

Fast-pulsed converters are used with cycling circuits to generate a pulse with a flat-top duration of only a few milliseconds and synchronous to an event such as beam injection or extraction.

This paper describes the use of the third generation of the CERN-designed Function Generator/ Controller embedded platform (FGC3) to control the four types of fast-pulsed power converters currently being commissioned in CERN's new Linac4 (see Table 1).

The Linac4 beam is 400 µs long, whilst the flat-top pulse is stable solely during a few milliseconds. As these two events must be synchronized to ensure the correct functioning of the accelerator, the FGC3 must guarantee time accuracy and inter-FGC precision better than 10 µs.

Table 1: Fast-pulsed Power Converters in Linac4

Name	Peak Power	Pulse	Number*
Mididiscap	30 kW	5 ms	50
Maxidiscap	4 kW	2 ms	48
Modulator	5.5 MW	1.8 ms	14
HMinus	150 kW	1.2 ms	3

FUNCTION GENERATION/CONTROLLER

Most power converters currently being installed at CERN are controlled with FGC3s, shown in Fig. 1. This embedded device includes a mainboard, an Ethernet-based communication card† and an analog card with four high precision ADC channels (ADS1274) and two 16-bit DACs (MAX5541) [2]. The mainboard includes a Renasas RX610 microcontroller (running a small footprint real-time operating system called NanOS), a TI TMS320C6727 floating point DSP with a 10 kHz interrupt-driven task and a Xilinx FPGA for glue logic and peripheral handling.

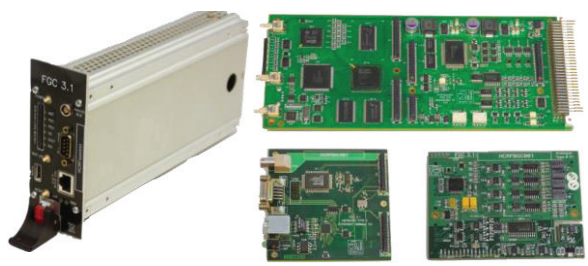


Figure 1: Function Generator Controller: mainboard, Ethernet and analog cards.

The FGC3 is integrated within a CERN-designed chassis called RegFGC3 [3], shown in Fig. 2. A backplane links the FGC3 with a variety of cards dedicated to state control, analog interlock, digital interlock, measurement conditioning, etc. Communication buses over this backplane include:

- SPIVS: serial bus used by the FGC3 to send the reference value in digital form.
- SCIVS: serial bus used by the FGC3 to exchange parameters with the cards.

* Number of converters on completion of Linac4.

† The FGC3 can also be fitted with a WorldFIP-based communication card.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- QSPI: a serial bus supporting detailed diagnostics.

Although the number and type of cards connected to a RegFGC3 chassis depends on the needs and specifications of a given power converter, the state control card is always present. It implements the state machine for the particular power converter. It is partially driven by the FGC3 output commands such as power on, power off, reset and timing pulses (described in the section Timing Pulses), and it relays back to the FGC3 status and fault information.

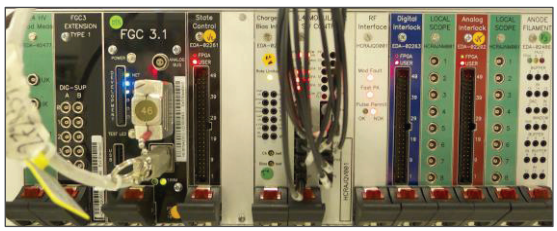


Figure 2: RegFGC3 chassis for the Modulator converter.

CONTROL OF OPERATIONAL POWER CONVERTERS

The three-tier control system of the FGC3-based power converters is illustrated in Figure 3. The equipment tier encompasses the pulsed converter, RegFGC3 crate, Ethernet switches, and sync pulse injectors, whilst the top tier includes the controls applications, alarms service, etc. mainly used by the machine operators. Between the FGC3s and these applications lie the front-end computers known as FGC_Ether gateways. The gateways[‡] is a rack-mounted Linux-based computer with two network interface cards: one connected to the CERN Technical Network for access to the upper control layers and the second connected to a gigabit Ethernet network with enough capacity to manage a cluster of up to 64 FGC3s. Each FGC3 integrates a 100 Mbps LAN chipset to manage the raw Ethernet-based communication.

Control applications can perform three operations on a device: get a property, set a property and subscribe to a property. A property is a device-specific parameter through which the underlying hardware – a power converter in this case – can be configured, controlled and monitored.

Gateways receive these commands over the Technical Network using an in-house distributed communication protocol known as Controls Middleware (CMW) [4]. The commands are reformatted and forwarded to the appropriate FGC3. A task associated with the command is then executed in the FGC3 and a packet with the command response is transmitted upstream through the gateway to the control application.

[‡] In the FGC lingo and by extension in this paper the term 'gateway' refers to the front-end computer responsible to forward to and fro commands and responses between the upper control layers and the FGCs, thus acting as a gateway to these messages and hence the name.

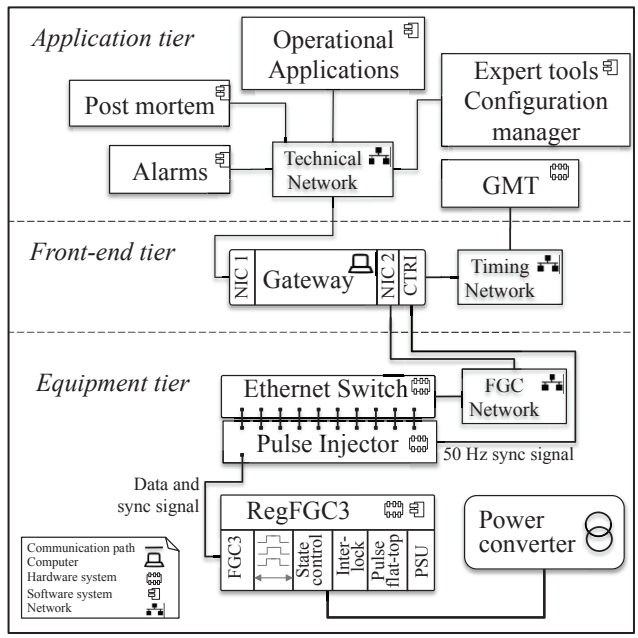


Figure 3: Power converter control.

TIMING DISTRIBUTION

The correct operation of the accelerator depends on the accurate timing of the pulse generation. To this end, the General Machine Timing (GMT) plays a preeminent role by distributing timing events and telegrams over a dedicated communication network to timing receiver cards, driven by a UTC-synchronous 40 MHz clock with small jitter and known delay [5,6]. One such card installed in all the gateways is a PCI variant known as the CTRI. These timing packets provide information on the machine state; post-mortem; the cycle selector of the next cycle within the super-cycle; and the delay in milliseconds of the next event, be it a start of cycle, beam injection or extraction.

In addition, the CTRI cards in all the gateways are configured to generate a 50 Hz synchronization signal on one of its four outputs. This is linked through a coax cable to the first Pulse Injector. This CERN-designed 24-port box injects the sync signal onto a spare pair of wires of the 100 Mbps Ethernet cable. Thus, a single Cat7 cable from the Ethernet switch to the FGC3 device coalesces the data communication link and the synchronization signal (Fig. 3), which saves significantly on cables and connectors. The combination of an Ethernet switch and a Pulse Injector is known as an FGC_Ether star point [7]. Up to three star points can be daisy chained using a backbone of Gigabit Ethernet and a coax cable to enable one gateway to support up to 64 FGC3s.

In the FGC3, the 50 Hz sync signal is utilized to discipline the PI-based Phase Locked Loop (PLL), which synchronizes a 25 MHz Voltage Controlled Crystal Oscillator (VCXO) through a 14-bit DAC. The resulting clock has a jitter below 40 ns and accuracy better than 2.5 µs [8]. The clock is fanned out to the various integrated circuits in the FGC3 (MCU, DSP, FPGA,

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ADCs, and DACs) making the firmware and the interrupt-driven software tasks in the MCU and DSP fully synchronous with the GMT.

FGC3 SOFTWARE

The first software class written for the FGC3 devices provides function generation and regulation [9], property management, event and analog signals logging, ADCs and DACs calibration, monitoring and diagnostics of the converter, and an interface with the interlock system.

This code base has been reused as the foundation for a new class of software used to control fast-pulsed power converters. Function generation and regulation is not needed and thus have been removed. Instead a mechanism to synchronize the converter with the GMT has been developed to achieve the required time constraints. Due to size restrictions, this section only presents two of the features specific to this software.

Timing Pulses

The FGC3 commands the electronics of the fast-pulsed power converter with timing pulses. These control the use of the capacitors stocking the energy needed to generate the current or voltage pulses.

Figure 4 provides a typical sequence of events starting with the GMT transmitting a timing event over the timing network to the CTRI in a gateway. The software running in the gateway formats the event and broadcasts it to all the FGC3s connected to the FGC Ethernet network. This information includes the current UTC and millisecond time, the type of event and the remaining time in milliseconds until the event will occur. The FGC3 retrieves this last value and writes it into the FPGA register TIME_TILL_EVENT. This register implements a free-running counter that starts down counting at 1 MHz at the start of the next millisecond. Thus, the exact event time coincides with the register reaching the value zero.

The FPGA also provides a peripheral consisting of a set of GPIOs that can generate timing pulses. Each output channel has two associated registers: PULSE_ETIME defines the time of the rising edge of the pulse with respect to the TIME_TILL_EVENT and PULSE_WIDTH specifies the pulse duration in microseconds.

As represented in the example of Fig. 4, the timing event is received 875 ms in advance. The timing pulse A0 controls the capacitor charging time, which starts 410 ms prior to the arrival of the beam and lasts for 400 ms. The timing pulse A1 controls the capacitor discharging time spanning from 4 ms before the event to 1 ms after the event. Finally, A2 is used to synchronize the acquisition measurement sampled precisely at the event time. This measurement is then made available to the operators.

To complete the above description it should be noted that the Modulator converter, employed to power the radio-frequency klystrons, requires a fourth timing signal to trigger an active bouncer used to stabilize the pulse.

The FPGA registers PULSE_ETIME and PULSE_WIDTH for each output channel (A0, A1, A2) are initialized from FGC3 properties, which are

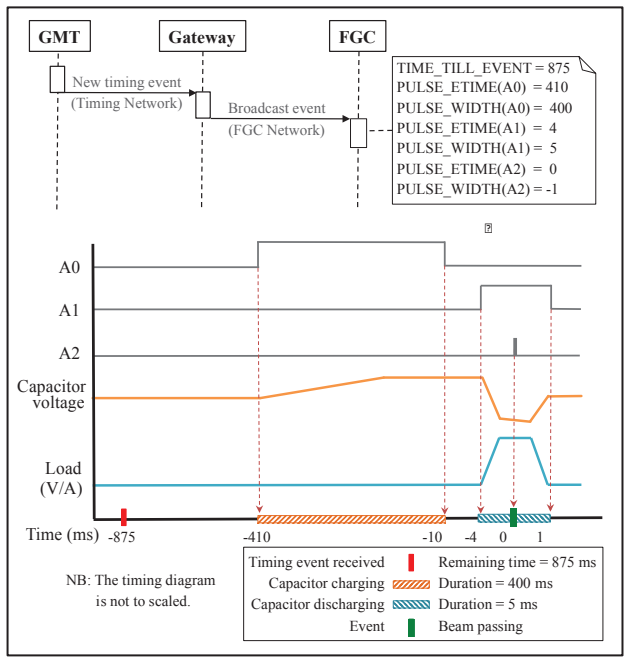


Figure 4: Simplified chronogram of timing pulses for fast-pulsed converters.

configured based on the requirements of the fast-pulsed power converter. With this strategy, converters with different topologies can be controlled homogenously.

Pulse State Machine

The main power converter state machine encoded in the FGC3 [10] is driven by external stimuli (inputs, commands, etc.) and accounts for various working modes: direct, cycling, economy, etc. Fast-pulsed converters operate in the cycling state, allowing operators to specify a different pulse reference for each cycle. This state has been extended with a pulse state machine (Fig. 5) to sequence the generation of pulses. The following functionality is implemented within each pulse state:

- Waiting: waits for a timing event.
- Preparing: verifies if the pulse for the current cycle is enabled and latches the reference and timing parameters.
- Setting: configures the timing pulses as described in the previous subsection; sends the reference value to the converter; and if a polarity switch is present and its position does not match the sign of the reference, the output command to change the polarity switch is activated.
- Reporting: gathers the current and voltage measurements and verifies if a fault condition has occurred. This information is published and made available to the operators.
- Fault: logs the error condition.

Some of these states have an associated timeout. If the timeout expires before the onwards transition is asserted, the state machine defaults into the Fault state, where the appropriate information is logged. This is then followed by a transition to the Waiting state to handle the next

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

timing event. This prevents the state machine from locking up, allowing subsequent timing events to be handled and thus improving the software resilience.

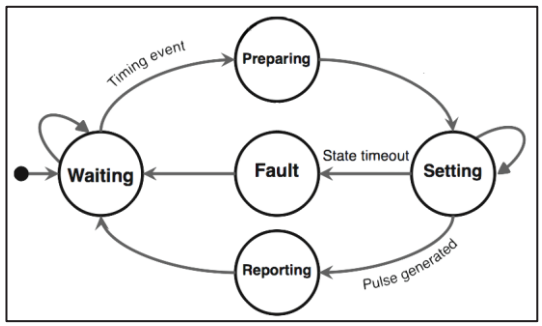


Figure 5: Pulse state machine.

PERFORMANCE

Since the FGC3s are synchronized with the General Machine Timing, the timing pulses and consequently the current or voltage pulses are generated synchronously. This can be observed in Figure 6, which shows pulses with various references sampled from six circuits installed in Linac4. Note how the overlapping flat-top stability is of merely 1.5 millisecond. Longer pulses would require an over-dimensioning of the power system, resulting in higher costs.

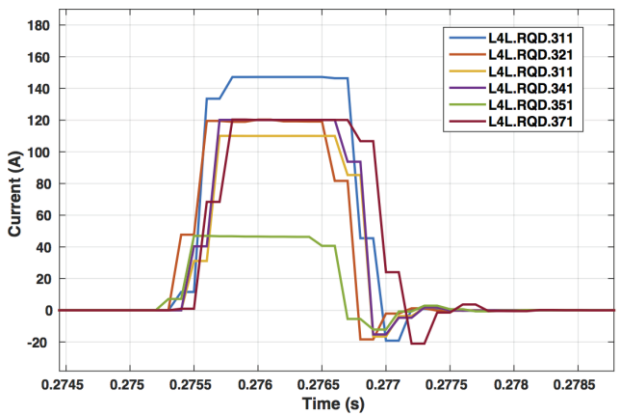


Figure 6: Six Maxidiscaps pulsing synchronously.

The FGC3 timing accuracy is demonstrated in Fig. 7, which includes three oscilloscope snapshots. The left and middle ones were taken in the laboratory. The signals probed comprise: the pulse; the exact event time as output by the CTRI card in the gateway; and the acquisition timing pulse (A2) defining the event time from the FGC3's perspective. It can be observed how the difference between these latter two signals is less than 200 ns, well below the requirement of 10 µs. The right image shows the acquisition timing pulses (A2) of three different Maxidiscap converters connected to one gateway and a Modulator connected to a different gateway in Linac4. The inter-FGC timing precision among the three Maxidiscaps is negligible whilst between these and the Modulator is only 186 ns, due to differences in cable length. In any case these values are also below the 10 µs requirement.

FUTURE WORK AND CONCLUSIONS

Two important software features are yet to be developed:

- Fast sampling: the software must be adapted to provide support for a new analog interface which is based on four LTC2378 SAR ADCs sampling at a rate of 500 ksps. This will allow acquiring the pulsed signals with a better time resolution.
- Slow regulation: the Modulator converter works in open loop, i.e. the flat-top pulse is not regulated. The FGC3 shall implement an algorithm that trims the reference value on a pulse by pulse basis to minimize the error.

Despite the above, the core functionality of the software is fully operational and has been integrated with the tools in CERN's new Linac4 accelerator to control the currently installed. When complete, there will be 115 circuits in Linac4 controlled by this software.

Working in collaboration with the converter designers has proved essential to successfully reach this goal. Their feedback was crucial to provide an intuitive interface and to correctly parameterize each circuit. In return, an unprecedented level of remote diagnostics and monitoring is now available, which will shorten the intervention times during machine exploitation.

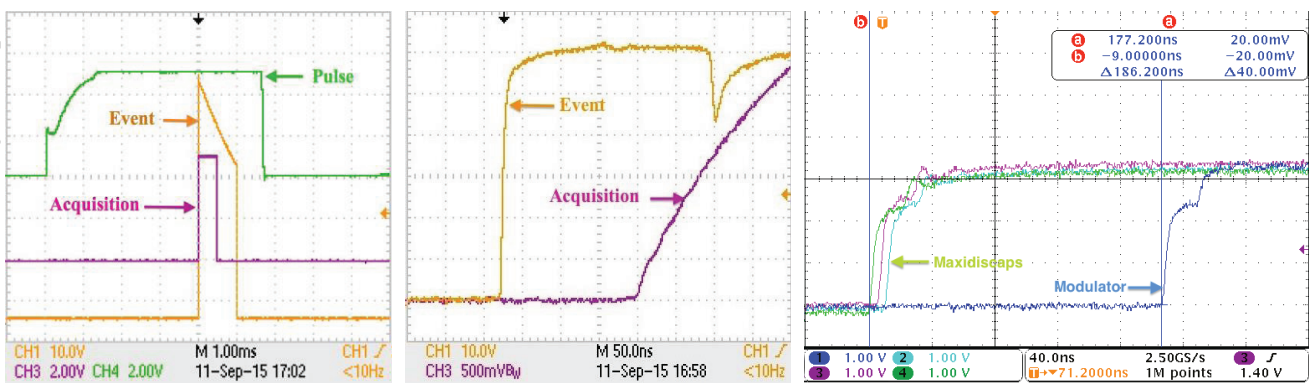


Figure 7: Precise synchronization between the pulse and the event proving time accuracy and precision.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REFERENCES

[1] CERN Engineering Department: http://ern.ch/en-dep/CERN_Energy_Consumption/2012/ElectricityFlyer2012.pdf

[2] D. Calcoen et al., "Evolution of the CERN power converter function generator/controller for operation in fast cycling accelerators", ICALEPCS, Grenoble, France (2011).

[3] M. Di Cosmo et al., "The new modular control system for power converter at CERN", ICALEPCS, Melbourne, Australia (2015).

[4] K. Kostro et al, "The Controls Middleware (CMW) at CERN – Status and Usage", ICALEPCS, Gyeongiu, Korea (2003).

[5] J. Serrano et al., "Nanosecond level UTC timing generation and stamping in CERN's LHC", Proceedings of ICALEPCS, Gyeongju, Korea (2003).

[6] J.C.Bau et al., "Managing the real-time behavior of a particle beam factory: the CERN Proton Synchrotron complex and its timing system principles", IEEE Transactions on Nuclear Science 45, 1998.

[7] S. Page et al., "Migration from WorldFIP to a Low-Cost Ethernet Fieldbus for Power Converter Control at CERN", ICALEPCS, San Francisco, USA (2013).

[8] M. Magrans et al., "Clock and Event Synchronization of the Function Generation/ Controller", ICALEPCS, Melbourne, Australia (2015).

[9] Q. King et al., "Function generation and regulation libraries and their application to the control of the new main power converter (POPS) at the CERN CPS", ICALEPCS, Grenoble, France (2011).

[10] Q. King et al., "CCLIBS: The CERN Power Converter Control Libraries", ICALEPCS, Melbourne, Australia (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

WHITE-RABBIT BASED REVOLUTION FREQUENCY PROGRAM FOR THE LONGITUDINAL BEAM CONTROL OF THE CERN PS

D. Perrelet, Y. Brischetto, H. Damerau, D. Oberson*, M. Sundal#, A. Villanueva, CERN, Geneva, Switzerland

Abstract

The measured bending field of the CERN Proton Synchrotron (PS) is received in real-time by the longitudinal beam control system and converted into the revolution frequency used as set-point for beam phase and radial loops. With the renovation of the bending field measurement system the transmission technique is changed from a differential sequence of pulses, the so-called B-train, to a stream of Ethernet frames based on the White Rabbit protocol. The packets contain field, its derivative and auxiliary information. A new frequency program for the conversion of the bending field into the revolution frequency, depending also on parameters like radius of the accelerator and the particle type, has been developed. Instead of storing large conversion tables from field to frequency for fixed parameters, the frequencies are directly calculated in programmable logic (FPGA). To reduce development time and keep flexibility, the conversion is processed in real-time in the FPGA using Xilinx floating-point primitives mapped by a higher level tool, Simulink System Generator. Commissioning with beam of the new frequency program in the PS is progressing.

INTRODUCTION

The CERN PS accelerates protons and ions [1] for fixed target experiments and as part of the LHC injector chain. For protons, its kinetic range covers 1.4 GeV at injection to 26 GeV total energy. The longitudinal beam control system drives in total 25 cavities in a wide frequency range (2.8-10, 20, 40, 80 and 200 MHz). It requires the so-called open-loop revolution frequency to be derived from the bending field of the main dipole magnets which serves as a reference for the feedback loops closed around the beam. In addition to protons from the PS Booster (PSB) the PS accelerates a large variety of heavy and light ions such as $^{208}Pb^{54+}$ [2,3,4] or $^{40}Ar^{11+}$ injected from LEIR.

Motivated by the renovation of the present measurement system of the magnetic field, its distribution in the form of asynchronous pulses (so-called B-train) [5, 6] has been upgraded [7] to a stream of Ethernet frames based on the White Rabbit (WR) protocol. The reception of the magnetic field information for the longitudinal beam-control and its conversion into the open-loop revolution frequency has been redeveloped and commissioned with beam.

*HEIA-FR (Haute école d'ingénierie et d'architecture Fribourg)
#IST (Instituto Superior Técnico Lisboa)

FIELD MEASUREMENT

The existing system uses a peaking-strip (PKS) marker at 49.8 Gauss as an absolute reference at the start of each cycle. This implies to ramp down to a low field between cycles to reset the integration process, consuming time and power. Moreover, the B_{up}/B_{down} pulses distribution scheme triggered by 0.1 G changes in B at a maximum rate of 500 kHz is at the limit for ions in terms of resolution and also in case of flat zones.

In the new system, the measurement setup is referenced at each cycle by a Ferromagnetic Resonance (FMR) tunable marker and integrates a flux coil voltage, meaning the derivative of B. This kind of marker has a working point closer to the injection field, thus reducing the ramp down constraint besides improving the injection accuracy. The regulation done in B, has to cover the range from 600 G to 13000 G with a maximum changing rate of ~ 30 kG/s.

Following the acquisition process, the magnetic field and its derivative are transported via Ethernet frames using the White Rabbit protocol over optical fibres [8]. At the reception side, B is extracted and then converted into the open-loop revolution frequency.

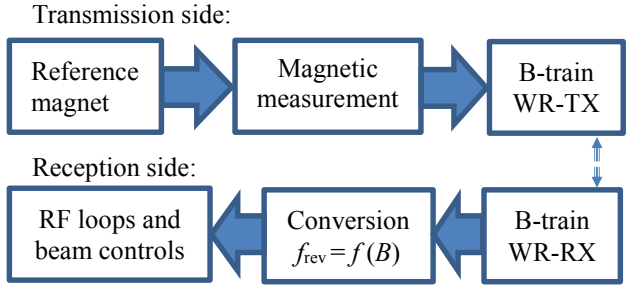


Figure 1: Overview of the whole system and its connection between reference magnet and longitudinal beam control.

OPEN-LOOP REVOLUTION FREQUENCY

In a synchrotron, the orbit of the particles can be controlled by changing the bending field in the dipole magnets or the RF frequency. A theoretical relationship between magnetic field and revolution frequency can be derived from machine parameters [9,10]. According to

$$f_{rev}(B,b) = \frac{c}{2\pi R_{nom}} \frac{1}{\sqrt{1+\left(\frac{E_0}{b \cdot B \cdot c \cdot \rho_{nom}}\right)^2}} \quad (1)$$

with the velocity of light in vacuum c, the mean radius R_{nom}

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

of the vacuum chamber and the bending radius of the bending magnet ρ_{nom} to end with the relation Eq. (1) describing the revolution frequency f_{rev} as a function of the magnetic field B. For ions, a charge-over-mass scaling factor b relative to protons (b=1.0) is introduced

$$b = Z_{ion} \frac{E_{0,proton}}{E_{0,ion}}. \quad (2)$$

requiring only the charge Z_{ion} of the species and its rest mass energy $E_{0,ion}$. This scaling factor is then directly applied as a factor to the magnetic field B. Figure 2 shows

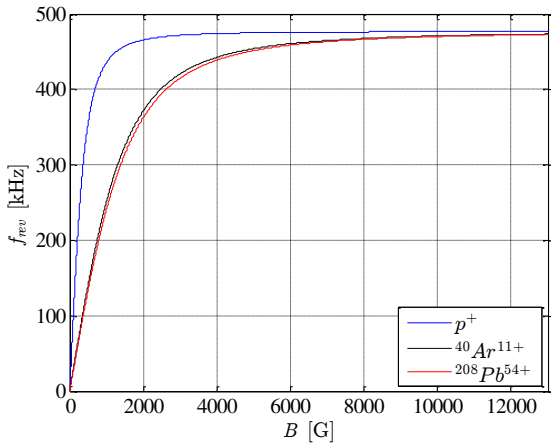


Figure 2: Revolution frequency function of B.

frequency swing for different particle accelerated in the PS.

Table 1: $f_{rev}=f(B_{inj}, B_{ej}$, particle type)

Species	Region	B [G]	f_{rev} [kHz]
p^+	Injection	1013	436
	Ejection	12550	477
$^{208}Pb^{54+}$	Injection	685	178
	Ejection	12550	473
$^{40}Ar^{11+}$	Injection	685	187
	Ejection	12550	473

Equation (1) shows that the frequency sensitivity is greater for ions than for protons and in particular at low energy.

Table 2: Δf_{rev} and ΔR for a ΔB of 0.1 G at Injection

species	b	Δf [Hz]	ΔR[mm]
p^+	1	7.0	0.32
$^{208}Pb^{54+}$	0.261571	22.3	0.40
$^{40}Ar^{11+}$	0.277303	23.1	0.40

The simple theoretical relationship (1) is not sufficiently precise, mainly due to saturation effect in the main magnet and hypothetically due to uncertainties and dispersion in the magnetic length of the magnets, non-linarites and other errors in addition to the fact that the beam over its path sees the integrated B and not exactly the measured B_{ref} at a specific location in the magnet).

Therefore a fit-based frequency model [11] has been implemented for the PS to reduce the action of the radial loop of the beam control. The polynomial coefficients are extracted using beam-based measurements of the radial loop error to perform a fit optimization of the low energy part of (Eq. 1) in order to minimize the radial loop correction. The fit-based revolution frequency model becomes

$$f_{rev}(x) = \frac{c}{2\pi R_{nom}} \frac{1}{\sqrt{1+\frac{N}{P(x)}}} \quad (3)$$

with $x=b\cdot B$ and

$$P = \left(x\left(1+a_1x+a_2x^2+a_3x^3+a_4x^4+a_5x^5\right)\right)^2. \quad (4)$$

TRANSMISSION/RECEPTION

The WR technology [8] developed at CERN as an extension of the IEEE 1588 protocol, a time precise Ethernet, has been chosen as a transmission standard to distribute the magnetic information in the accelerator complex. There are two ways of using it, the pulse mode where the reference timing is given by the grand master and all slaves align to it allowing generation of grouped precise pulses or simply in Ethernet packet streaming mode, the one currently used.

For this application, a dedicated frame (Fig. 3) with specific type (0x42, ASCII code of 'B' for B_{field} measurement frame) and content has been defined as a standard to transmit magnetic information. The latency of the link is monitored by extracting and differentiating the time included in each frame. To guarantee a transmission rate of the existing B-train, the frame rate is constant and currently set to 250 kHz, with a maximum rate defined by the size of the payload and number of users in a standard Ethernet 1 Gbps network.

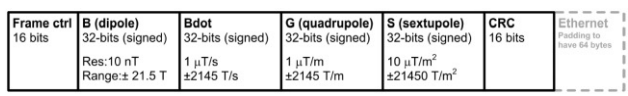


Figure 3: Layout of B field frame.

IMPLEMENTATION (CORE)

The new frequency program uses mainly standard hardware as open hardware designs (SPEC and SVEC) [8,12] related to the WR project. New FMC daughter boards hardware designs (ADC, CVORB, SERIAL40M)

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

[10] and specific firmware to adapt the data format from the WR core to the transmission standard of the RF beam control have been developed. A new approach has been explored and validated for the conversion of B to f_{rev} implemented in the FPGA of the SVEC board. The conventional way might have been to pass through the development of a fixed-point implementation and iterative algorithms to perform elaborated mathematics like the reciprocal square root or via huge and static conversions tables present in the existing system to implement Eq. (3). The approach chosen was to calculate the revolution frequency on the fly in floating point algorithms in the FPGA. The calculated f_{rev} digital word is serially Manchester encoded for being finally distributed through the daughter board FMC-DTX-4CHA as a reference revolution frequency to the beam control. This so-called open-loop revolution frequency is then multiplied by the harmonic number. Some corrections such as a frequency steering offset and weighted radial and phase loop errors are applied to finally become the closed loop revolution frequency. The better the frequency program quality, the smaller is the difference between open and closed loop, which translates in a well centered orbit in the machine and less action of the radial loop.

To reduce the floating point development time, Matlab System Generator associated with the ISE Xilinx Primitives library has been used. The library contains optimized versions of advanced mathematic operations whereas the simulation and verification process is convenient and simple. Then the generation process ends up with a mapped and configured netlist directly targeting the project FPGA. These generated files have to be included in the Register Transfer to Logic design entry tools or directly in ISE where it is simply interfaced by port mapping to the main part of the design. This way of designing gives a higher abstraction level to the FPGA and allows to build complex blocks more rapidly. However the commercial tool is vendor specific and requires a license.

The complete project is a versatile assembly of different languages (VHDL2008, Verilog2010 and native C) and philosophy (generic code and use of specific IP cores). In the WR Core, an instantiated LatticeMico32 soft core processor described in Verilog is initialized by a compiled and linked C code. After the compilation and generation processes, a unique file is given to the final tool Xilinx ISE to perform the fitting for the Spartan6 FGPA.

EXPERIMENTAL RESULTS AND COMMISIONNING

The transmission and reception of magnetic bending field frames over WR has been validated and commissioned with beam on operational cycles without a significant or notable difference on the radial loop signal (Fig. 7), confirming as well that the conversion part is fulfilling the requirements. This is a trustable indication about the whole chain validity from this complete new setup. The small difference between both systems is explained by the two independent measurement setups of the magnetic field which slightly deviate from each other. In both sides of the WR link, relevant signals are directly stored in the DDR memories(Fig.8-10), remotely available for acquisition and diagnostic. On top of this, drivers and FESA classes have been developed and deployed for the integration in the CERN infrastructure.

Measurements with beam (Fig. 4-7) are made with the operational user for LHC-type beam with 25ns spacing on 4 consecutive cycles with the existing setup and then switched to the new WR system for the 4 next cycles. They shows the existing B-field still measured with the PKS+B_{up}/B_{down} setup available in the samplers, the new Bfield FMR+WR accessed in the DDR memory through the fesa class, the closed loop revolution frequency analogue signal resampled (Fig. 5) and the radial loop with arbitrary units also resampled from the beam analogue signal (Fig. 6) of the LHC beam control. Figure 5 confirms that the cycle is properly accelerated through transition crossing up-to flattop. The only notable difference is during the ejection synchronization process and is visible at the end of the radial loop signal (Fig. 6 and 7). The cycle-to-cycle dependent start of the locking of the loop explain this behavior.

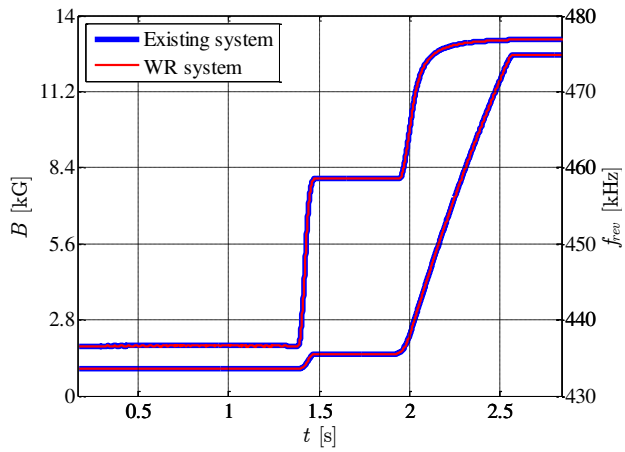


Figure 4: Measured magnetic field and associated closed loop revolution frequency of a LHC-type proton cycle.

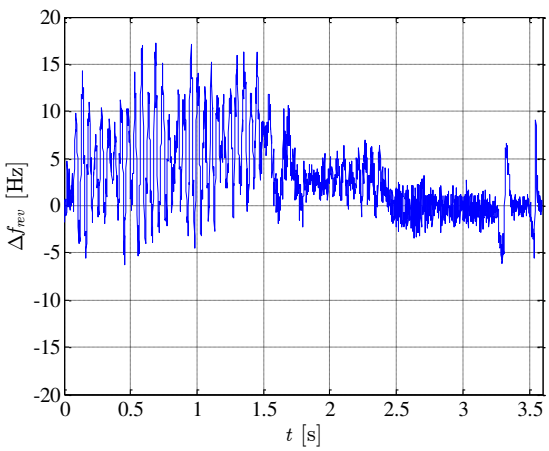


Figure 5: Difference Δf_{rev} between closed loop revolution frequency of existing and WR-system.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

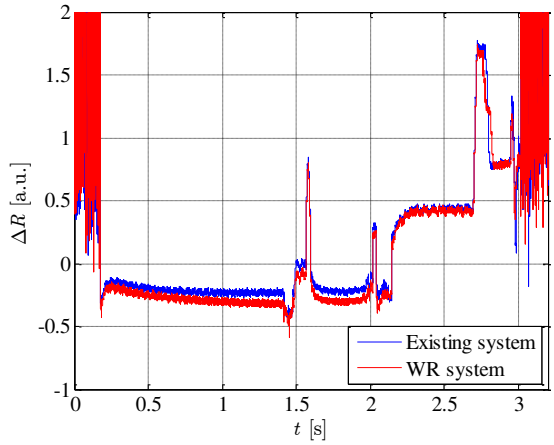


Figure 6: Measurement of the beam control radial loop error signal Δ*R* averaged on 4 cycles (LHC proton beam).

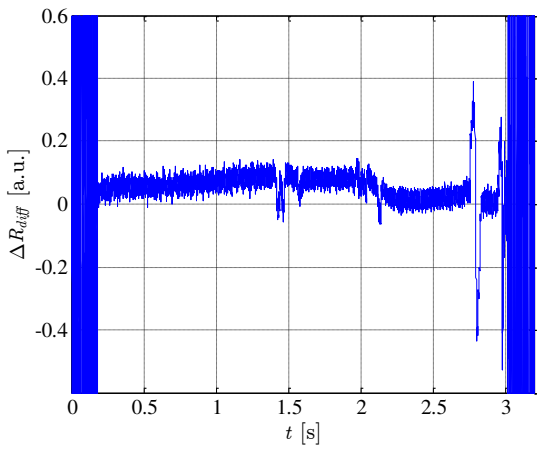


Figure 7: Difference performed on averaged radial loop errors Δ*R* of figure 5 for existing and WR-system. Spikes show blow-up, then transition crossing and ejection synchronization in the right.

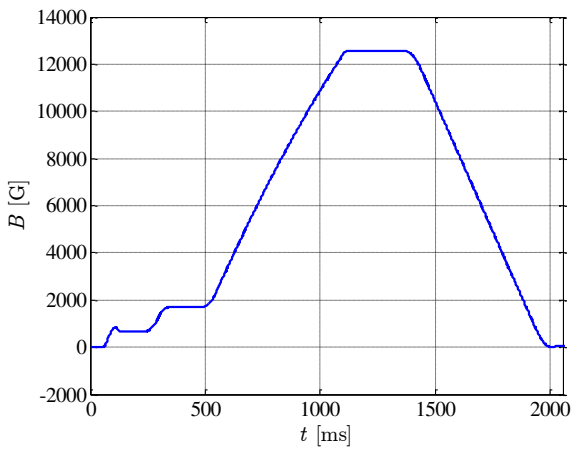


Figure 8: Received magnetic field available from on-board DDR memory of a single bunch ion cycle.

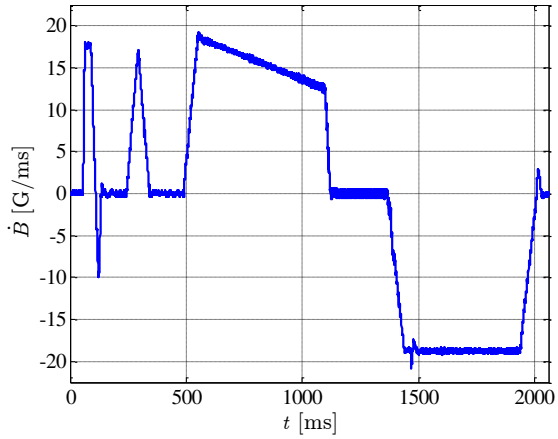


Figure 9: Received magnetic field derivative of ion cycle from Fig. 8.

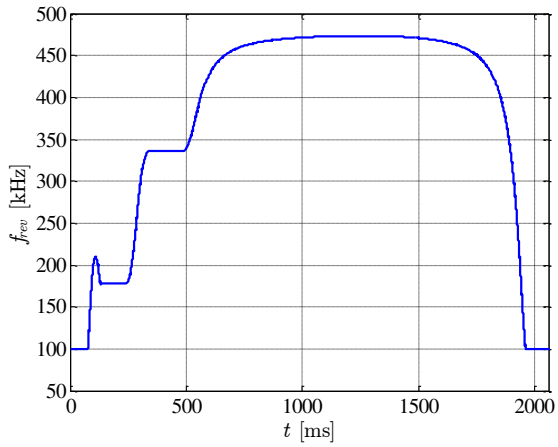


Figure 10:Real-time open-loop revolution frequency f_{rev} converted by WR system using the $^{40}Ar^{11+}$ scaling factor, b=0.2773.

CONCLUSION

New hardware and firmware developments have been completed to adapt the PS frequency program according to this new distribution scheme of the bending field over White Rabbit. Measurements with beam shows a minor difference between the existing and the new WR system (Fig. 5, 6 and 7) essentially because the bending field measurement setups are different and also the regulation is still performed using the existing field measurement setup. The new revolution frequency is real-time processed entirely in the FPGA with a floating-point signal chain giving accurate values.

Furthermore, acceleration of LHC type proton beam has been demonstrated successfully, confirming the functionality and capability of the new system on its own. As a last step, the new system will be put in operation after the completion of its integration in the CERN control system in addition to a dedicated commissioning phase with ions. Finally, some further development work is planned, notably to offer more advanced diagnostics signals in addition to a monitoring of the link status.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

ACKNOWLEDGMENT

The authors would like to acknowledge the support of colleagues from the CERN Radio-Frequency, Controls, Magnetic Measurement, and Operation Groups. In particular we thank A. Beaumont, S. Hancock, W. Höfle, M. Jaussi, G. Sterbini and T. Włostowski for their contributions to the project.

REFERENCES

[1] J. D. Simon, The CERN PS complex: a versatile particle factory, CERN-PS-96-019-DI, CERN, Geneva, Switzerland, 1996.

[2] M. Benedikt (ed.), The PS complex as proton pre-injector for the LHC: design and implementation report, CERN-2000-003, CERN, Geneva, Switzerland, 2000.

[3] LHC Design report III, Chapter 37, The PS and transfer line to SPS, CERN, Geneva, Switzerland, 2004.

[4] D. Manglunki, M. E. Angoletta, P. Baudrenghien, et al., Ions for LHC: Performance of the injector chain, WEPS022, IPAC11, San Sebastián, Spain, 2011.

[5] P. Dreesen, I. Garcia-Alfonso, A new B-train system for the PS accelerator, unpublished PS/PO technical note, 2002, CERN, Geneva, Switzerland.

[6] D. Cornuet, Présentation des trains B des accélérateurs, unpublished presentation EDMS document 844310 v.2, CERN, Geneva, Switzerland, 2007.

[7] M. Buzio, P. Galbraith, G. Golluccio, et al., Development of upgraded magnetic instrumentation for CERN real-time reference field measurement systems, MOPEB016, IPAC10 Kyoto, Japan, 2010.

[8] J. Serrano, M. Cattin, E. Gousiou, et al., White Rabbit status and prospects, THCOCA02, ICALEPCS2013 San Francisco, CA, USA, 2013.

[9] C. Bovet, R. Gouiran, I. Gumowski, et al., A selection of formulae and data useful for the design of A.G. synchrotrons, CERN-MPS-SI-INT-DL-68-3-Rev-1, CERN, Geneva, Switzerland, 1970.

[10] M. Sundal, H. Damerau, L. Ragnhild. et al., Development of a new Frequency Program in the CERN Proton Synchrotron, CERN-THESIS-2015-003, CERN, Geneva, Switzerland, 2015.

[11] S. Hancock, A fit-based frequency programme for the PS, AB-Note-2007-036 MD, CERN, Geneva, Switzerland, 2007.

[12] M. Cattin, E. Gousiou, J. Serrano, et al., CERN's fmc kit, WECOCB01, ICALEPCS2013 San Francisco, CA, USA, 2013.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INTEGRATION OF THE TRACK BEAM DYNAMICS MODEL TO DECREASE LINAC TUNING TIMES

Christopher E. Peters#, Clayton Dickerson, Francisco Garcia, Maria A. Power
Argonne National Laboratory, Argonne, IL, USA

Abstract

The Accelerator R&D Group within the Argonne National Laboratory (ANL) Physics Division maintains a beam dynamics model named TRACK. This simulation code has the potential to assist operators in visualizing key performance parameters of the Argonne Tandem Linear Accelerating System (ATLAS). By having real-time access to visual and animated models of the particle beam transverse and longitudinal phase spaces, operators can more quickly iterate to a final machine tune. However, this effort requires a seamless integration into the control system, both to extract initial run-time information from the accelerator, and to present the simulation results back to the users. This paper presents efforts to pre-process, batch execute, and visualize TRACK particle beam physics simulations in real-time via the ATLAS Control System.

INTRODUCTION TO ATLAS AND TRACK

The ATLAS accelerator is located at the United States Department of Energy's Argonne National Laboratory in the suburbs of Chicago, Illinois. It is a National User Facility capable of delivering ions from hydrogen to uranium for low energy nuclear physics research in order to perform analysis of the properties of the nucleus. As a result of the wide variation of beams delivered [1], re_ tuning of the entire machine is necessary on a near weekly basis. After a series of upgrades, ATLAS will consist of two possible source lines, a common injection and beam transport line, and 8 different target areas. This wide range of possible machine configurations combined with the thousands of individual devices which support them present a very real challenge to operators to arrive at a final tune quickly.

The TRACK beam simulation program developed at ANL-PHY is available and has been validated during the design of upgrade projects at ATLAS [2]. This code can perform individual particle simulations of a beam traveling through various types of optical and RF fields in order to predict final emittance and beam shape. The resultant plots show an animation in time of beam emittance ellipses, and an overall plot of beam size vs. distance. It should be possible to link the TRACK simulation code to the real-time running parameters of ATLAS in order to give the operators a detailed overview of the tune process. This effort requires that the program be integrated into the control system in a seamless way so that non-PhD personnel can operate and understand the behavior of the machine. The TRACK program's graphical user interface (GUI) only runs in Windows (Figure 1) while the control system is primarily developed using Vista Control Systems, Inc.'s Vsystem [3].

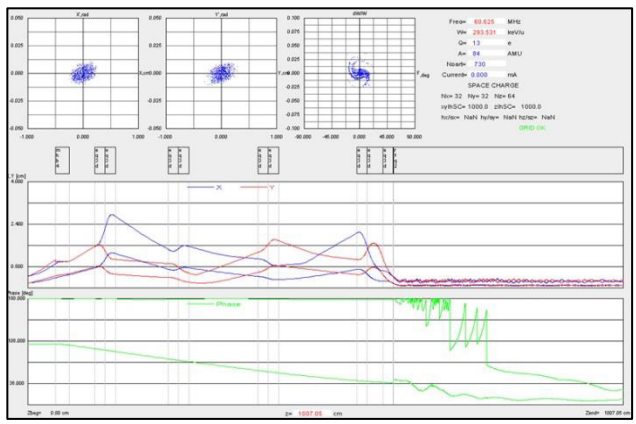


Figure 1: Existing TRACK GUI and Simulation.

USER INTERFACE & CONFIGURATIONS

In order to fully realize the potential of fast, real-time iterations on tuning time, the interface to TRACK must operate very similarly to the existing control system. Therefore, brand new interface windows were designed using native control system libraries (Figure 2). One important difference between the TRACK application GUI and the control system GUI is the presence of 'configuration' buttons. These are pre-selected input conditions, source/target line selections, and graph views which can be executed more quickly than re-running a full simulation for the entire machine.

Figures 3 and 4 represent a full TRACK simulation from current device settings by the operator. Note the black boxes which indicate corresponding quadrupole devices. By using the same interface as normal, operators can move sliders and immediately see the resultant simulation update.

USE CASES OF TRACK AT ATLAS

There are two main ways in which a pre-processed simulation of the beam could be useful to operators. The first is by importing the real-time settings of each device in the current beamline to give operators an overview of the performance of the machine. The second is by utilizing a database of previous runs to predict which machine configurations have been successful in the past. This second method will be useful before the start of a new experiment to predict future good tunes.

This work was supported by the U.S. Department of Energy, Office of Nuclear Physics, under Contract No. DE-AC02-06CH11357. This research used resources of ANL's ATLAS facility, which is a DOE Office of Science User Facility.

#ChrisPeters@anl.gov

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

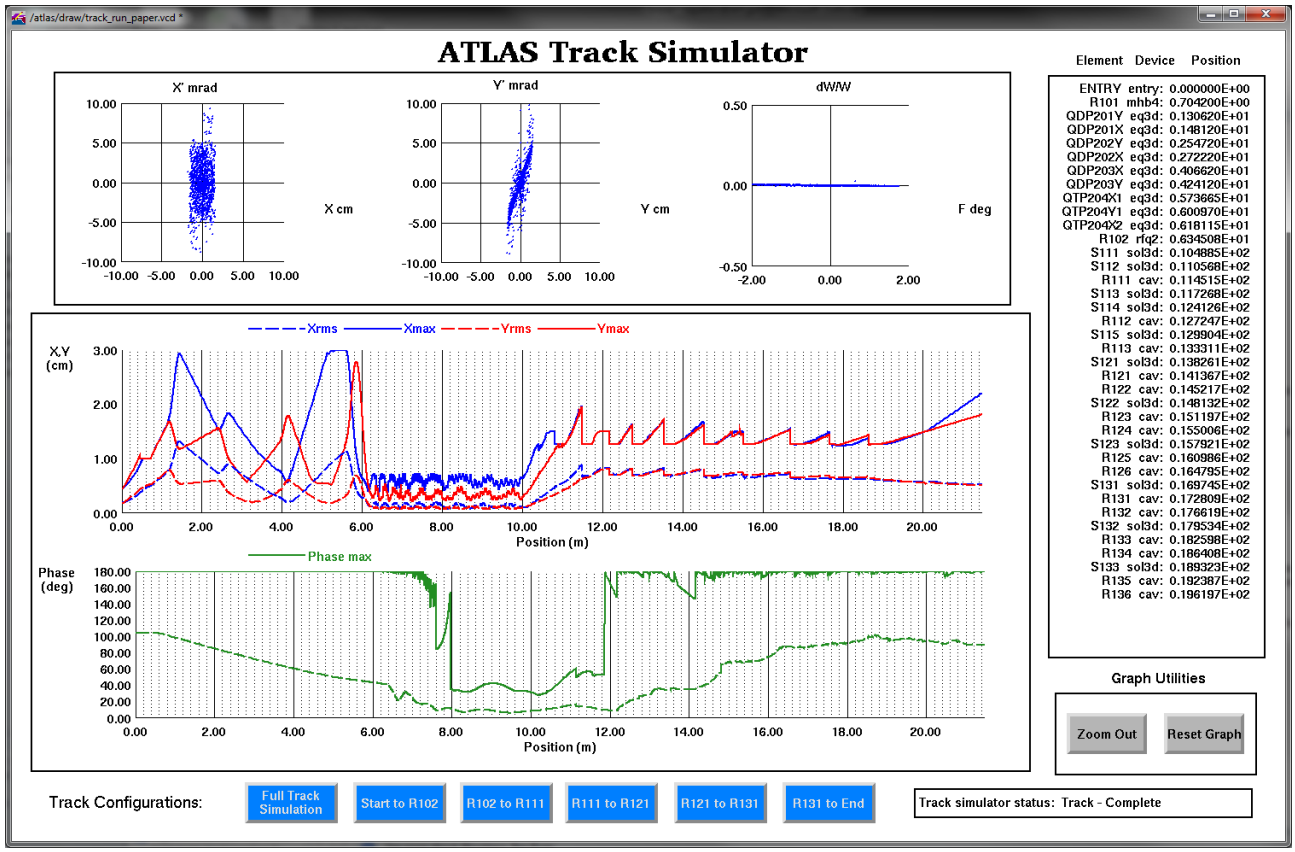


Figure 2: Control System TRACK GUI: (top) animated emittance plots; (middle) red and blue lines show the transverse X/Y size of the beam; (bottom) maximum and mean longitudinal phase plots.

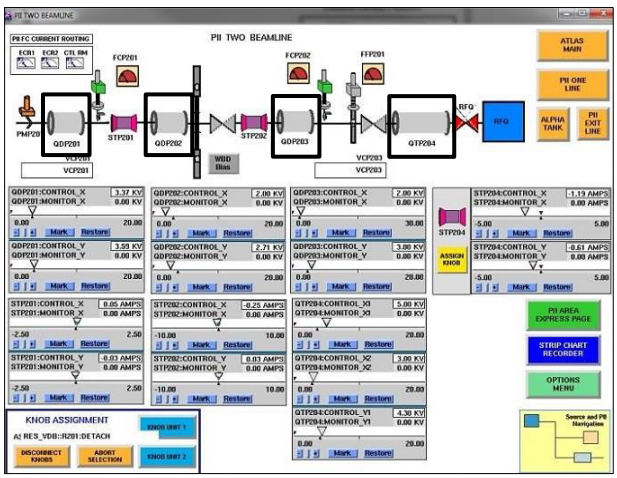


Figure 3: Simulating four quadrupole adjustments via the standard control system interface.

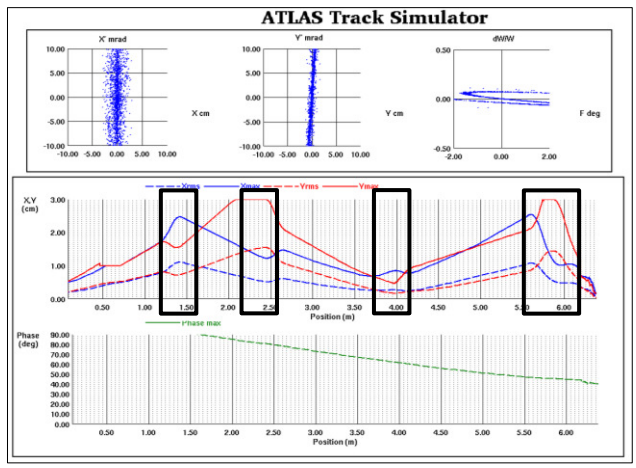


Figure 4: The four sets of brackets show the simulation results after adjusting the quadrupoles.

Simulation of Current Real-time Settings

The first integration of a beam dynamics model into the control system provides operators with a full overview of the running experiment's beam quality. In order to simulate the current experiment, device settings are extracted from the running control system and scaled according to scaling parameters stored in the real-time database, specific to each device. These scaled values are then written, along with previously stored constants like device length and radius, to specially crafted input files. These input files are fed to TRACK and the resultant view of the simulation is displayed.

Simulation of Previously Archived Settings

The ATLAS Control System maintains a database of run parameters for all previous tunes [4] which runs continuously and archives relevant parameters every four hours or whenever triggered manually. This second simulation method has been implemented by utilizing a duplicate, offline version of the control system databases.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The historical run parameters are extracted from a SQL database and loaded into a staging SQL database on the real-time control system backup machine. Then the standard "Preload Machine" functionality is used to load all the previous parameters onto the backup machine real-time databases for simulation.

PROGRAM DEVELOPMENT AND FLOW

In order to extract real-time parameters, a series of new methods and programs were developed. The first program named TRACK_VSYS_InputGenerator extracts and formats an ordered device name list from an existing relational database. Depending on which source and target lines are currently selected in the control system, the program executes a series of SQL unions and sorts in order to automatically select each device which is currently in use. Next the program reads this ordered list, and for each device extracts the current set points and scales the control value to a corresponding electrical or magnetic RF field value (Figure 5). It then writes the scaled value to a specially crafted output file containing values and formats used by TRACK.

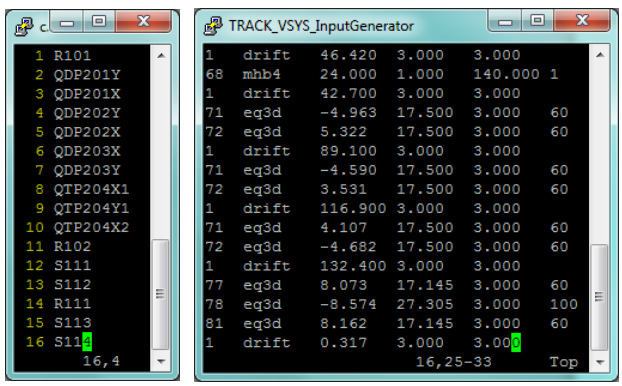


Figure 5: Example of automatically generated device list (left) and associated TRACK input file containing device parameters like radius, length, and field strength (right).

The scaling factors are stored within each database and are used to scale a control system variable, e.g. '0 – 10 volts' into a representative field value, e.g. '0 - 2.6 MV/m'. These scalers are either determined from manuals or empirically using equipment like a gauss meter. These scaler values, combined with assumed initial conditions, represent critical factors which influence the accuracy of the overall simulation.

Upon completion of the TRACK simulation, the results are in the form of a large set of text files, which defines the beam state after each device. This file is then parsed by a new program called TRACK_VSYS_Graph, which reads the data and automatically uploads it into the control system for input into the new GUI system.

EFFECT OF INITIAL CONDITIONS

TRACK uses, as part of its input files, a set of parameters which define the starting normalized emittance of the beam. There are two proposed methods to determine these initial conditions, only one of which is currently in use. The R&D group has developed a PepperPot emittance measurement program [5] written in LabVIEW [6], which can take 'snapshots' of the beam projected onto a phosphorescent plate through small holes arranged in a grid layout. The program then automatically calculates the 'Twiss' parameters [7] associated with the beam. A second method is to use a beam profile monitor (BPM) and quadrupole combination. By varying the quadrupole field and monitoring the BPM, Twiss parameters can be determined [8]. One result of this work is the ability to test the level of influence beam emittance has on the tune parameters. By varying the initial conditions, it is possible to quickly determine exactly which devices are critical to achieve proper optics (Figure 6).

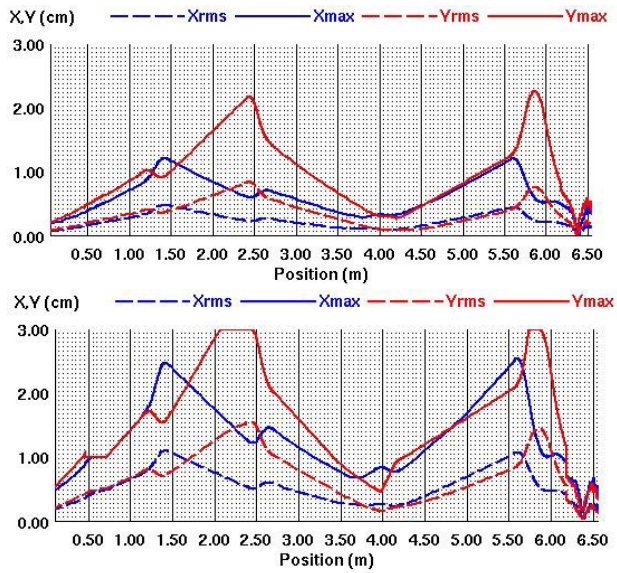


Figure 6: Effect (top to bottom) of changing only initial beam emittance parameters and keeping devices constant.

RUN TIME AND ITERATIONS

The TRACK program can be configured to perform a simulation of the entire defined beam path, or to continue from a pre-configured saved state from a previously run simulation. This 'continue' option can save a large amount of time if the user is only interested in one section of the beam path. These 'continue' options are saved as defined configurations within the buttons at the bottom of the TRACK simulation page (Figure 2). If the user pushes one of the partial beam path buttons, iterations of the tune become very rapid between changing a device set point and seeing the results on the screen.

INTERPRETATION OF RESULT PLOTS

Table 1 summarizes the execution time of the current single-threaded server setup, which is not yet optimized for a dedicated simulation machine. Note the decrease in execution time between a full simulation, and a pre-configured partial section of the beam line.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Table 1: Comparison of Full/Partial Execution Time

Program and Action	Time
Extract path info and create Device List	1.0 sec
Generate TRACK input files for devices	6.1 sec
Full Execution of injection line and linac	200 sec
Partial Execution of injection line only	11 sec
Plot & animate results on User Interface	10 sec

The graphs in Figure 7 represent one possible example of an A-B comparison example between beam shapes. In general, the top plot has more variations in beam focus and size, and also hits the beam pipe starting at the 4 meter mark. In comparison, the bottom plot has smoother trends using less focusing, and hits the pipe much less. The TRACK simulation predicts 100% transmission for the bottom plot, while the top plot loses beam at the pipe.

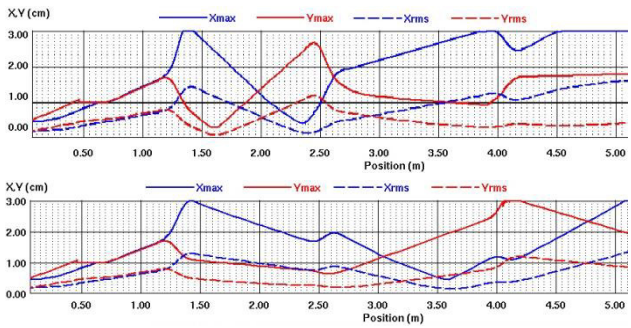


Figure 7: Comparison of good (top) & better (bottom) tunes. Note the bottom plot hits the beam pipe less.

RESULTS AND VALIDATIONS

This program is still in the early stages, and only a few beam line tunes have been simulated in advance of the actual experiment start date. The first time the plots were used, it was shown that operators had over focused the beam in several areas. By decreasing quadrupoles in key areas as suggested by the simulation, beam current immediately increased by ~5%.

Another area of validation of the TRACK simulation against machine performance is in the area of longitudinal beam size during runs with ions with relatively low mass/charge ratios. Beam transmission has historically been poor in these cases, and the simulation shows the potential for over-bunching of the beam. This is due to the configuration of the first buncher having a minimum amplitude level which is greater than the required field to bunch light ions (Figure 8).

FUTURE WORK

One of the greatest areas for improvement in iteration time is on the server which currently runs the applications. The current version runs on a standard desktop PC which is not dedicated to running simulations. A separate multi-core server with additional RAM has been ordered to speed up the simulation times.

There are several other applications going forward where this type of overall simulation can be useful. Pre-processing historical tunes would provide a catalogue of tunes which can be re-used. Operators could scroll through plots of beam sizes and determine which tunes could work for a future run. Additionally, automatic iteration of sequential simulations in order to converge on a full tune could be provided in advance of the tune day.

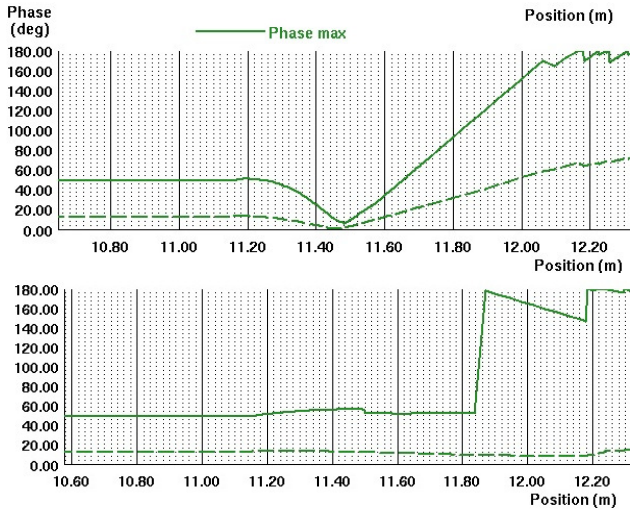


Figure 8: Top: over bunching of a 'light' beam as a result of required minimum field to run the buncher. Bottom: simulated beam at ideal 1/3 maximum amplitude.

REFERENCES

[1] "Stable Beams Available from ATLAS", www.phy.anl.gov/atlas/facility/stable_beams.html, retrieved October 12, 2015.

[2] B. Mustapha, V.N. Aseev, E.S. Lessner, and P.N. Ostroumov, "TRACK: The New Beam Dynamics Code," PAC 2005, Knoxville, Tennesee, May, 2005, TPAT028, p. 2053 (2005); http://www.JACoW.org

[3] Vista Control Systems, Inc. http://www.vista-control.com/

[4] M. Power and F. Munson, "Evolution of the Argonne Tandem Linear Accelerator System (ATLAS) Control System," ICALEPCS 2011, Grenoble, France, October 2011, MOPMS024, p. 371 (2011); http://www.JACoW.org

[5] S. Kondrashev, A. Barcikowski, A. Levand, P.N. Ostroumov, et. Al., "Emittance Measurements for Stable and Radioactive Ion Beams," LINAC 2010, Tsukuba, Japan, September, 2010, TUP086, p. 608 (2011); http://www.JACoW.org

[6] National Instruments Corporation LabVIEW, http://www.ni.com/labview/

[7] U. Raich, "Accelerator and Beam Diagnostics," U.S. Particle Accelerator School, June 23-26, 2009, uspas.fnal.gov/materials/09UNM/Emittance.pdf, retrieved Oct. 12th, 2015.

[8] K.T. McDonald, D.P. Russel, "Methods of Emittance Measurement," Lect.Notes Phys. 343 (1989) 122.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REAL-TIME BEAM LOADING COMPENSATION FOR SINGLE SRF CAVITY LLRF REGULATION

I. Rutkowski, M. Grzegrzółka WUT, Warsaw, Poland
Ł. Butkowski, C. Schmidt, DESY, Hamburg, Germany
M. Kuntzsch, HZDR, Dresden, Germany

Abstract

Stable and reproducible generation of a photon beam at Free Electron Lasers (FELs) necessitates a low energy spread of the electron beam. A low level radio frequency (LLRF) control system stabilizes the RF field inside accelerating modules. An electron beam passing through the cavity induces a voltage proportional to the charge and the cavity shunt impedance. The feedback loop tries to compensate for the perturbation after the accelerating gradient drops. The delay and high gain result in an overshoot or oscillations during transients. A feed forward signal can be applied to act on the plant simultaneously with the RF feedback. It can be generated off-line based on system characteristics and beam parameters or on-line using information obtained from the beam diagnostic systems. In the latter scheme fluctuations of the beam current are accounted for in real-time using an open-loop (feedforward) controller. The bunch charge detection scheme and its implementation is described. This paper describes results of the tests performed on the ELBE (Electron Linac for beams with high Brilliance and low Emittance) radiation source at the HZDR (Helmholtz-Zentrum Dresden-Rossendorf) facility using a MTCA.4-based LLRF control system.

INTRODUCTION

Stable and reproducible generation of a photon beam at Free Electron Lasers (FELs) necessitates a low energy spread of the electron beam. A low level radio frequency (LLRF) control system stabilizes the RF field inside accelerating modules. An electron beam passing through the cavity induces a voltage proportional to the charge and the cavity shunt impedance. For an SRF cavity with a high loaded quality factor (Q_L) compensation of the beam loading may require more RF power to be provided than is required to sustain the field gradient in the cavity, making it a major source of distortion of the RF field. Waveforms from a MTCA.4-based digital LLRF system using only a PI feedback controller are shown in Fig. 1 (gain in arbitrary units).

The doubling time of the integral controller is significantly longer than the pulse length (4 ms). Increasing the gain decreases the static error during the beam pulse, but results in transient oscillations (see Fig. 2). A feed forward signal can be applied to act on the plant simultaneously. It can be generated off-line based on system characteristics and expected beam parameters or on-line using information obtained from the beam diagnostic systems. It has been demonstrated [1] that real-time beam loading compensation

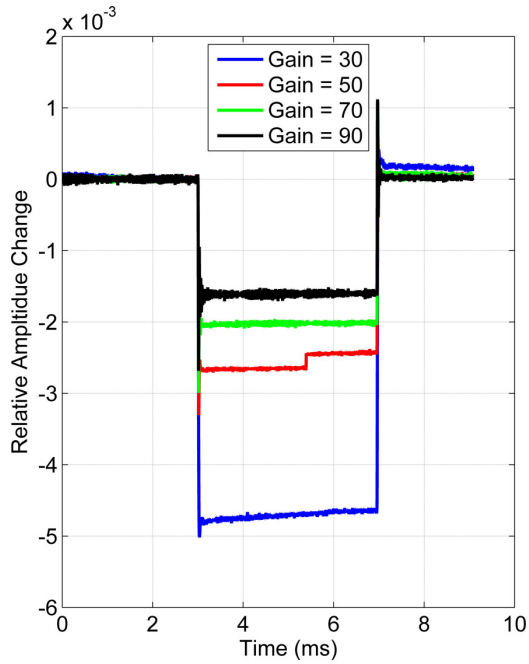


Figure 1: Response of a LLRF system using only a PI feedback controller to a beam pulse.

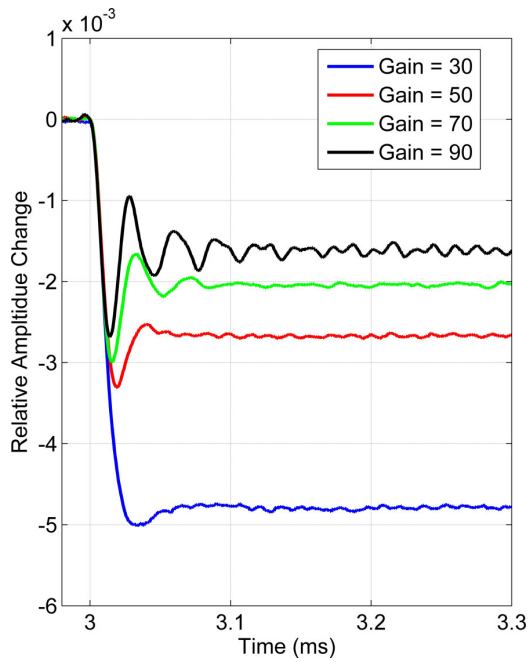


Figure 2: Response of a LLRF system using only a PI feedback controller to a beam pulse (zoom in during beam rise).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

(BLC) scheme reduces the bunch-to-bunch energy spread compared to control scheme using the PI controller only.

DETECTION HARDWARE

The bunch charge can be detected using various instruments. For real-time operation a non-destructive scheme is necessary. At ELBE facility an Integrating Current Transformer (ICT) is used. The signal at the output of the ICT is broadband and has low amplitude, resulting in low SNR. A high-gain, wideband amplifier is used to condition the signal. Very short rise and fall times make direct sampling of the signal impractical.

For beam-loading compensation only the information about signal's amplitude is necessary. A peak hold detector with a reset circuit can be used to sample the amplitude. The precise timing information is lost, but requirements for an analog-to-digital converter (such as bandwidth, maximum sampling speed, and aperture jitter) are lessened. The block diagram of the circuit used in this project is shown in Fig. 3.

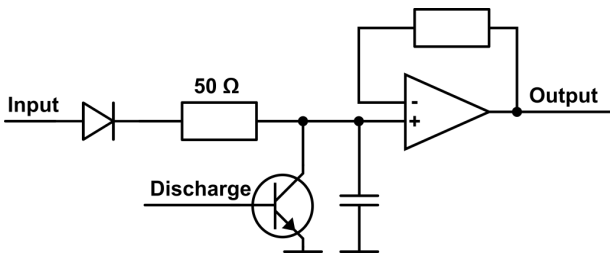


Figure 3: Block diagram of the peak detector.

The input signal charges a capacitor through a fast Schottky diode and a series 50 Ohm resistor which provides matching during signal transients. The voltage in the capacitor is buffered using a low input bias current operational amplifier. A fast bipolar junction transistor discharges the capacitor after the peak value is sampled. The discharge signal is synthesized from the accelerator's reference signal and shifted by appropriate number of cycles. Waveforms of the peak detector's output signal sampled by a high-speed real-time oscilloscope for various bunch charges are presented in Fig. 4.

For charge below 30 pC the signal is disturbed by strong oscillations during the first 15 ns after the rising edge. The transfer function of the system (voltage -> bunch charge) was characterized by relating the mean value of the signal (in time period 35 to 45 ns) to the known bunch charge using linear regression (Fig. 5).

The estimated model equals

$$Q = 0.1044 \frac{pC}{mV} * (V + 158.13)(mV) \qquad (1)$$

The charged (Q) is lineary proportional to the measured voltage (V) with a roughly 158 mV offset. The maximum

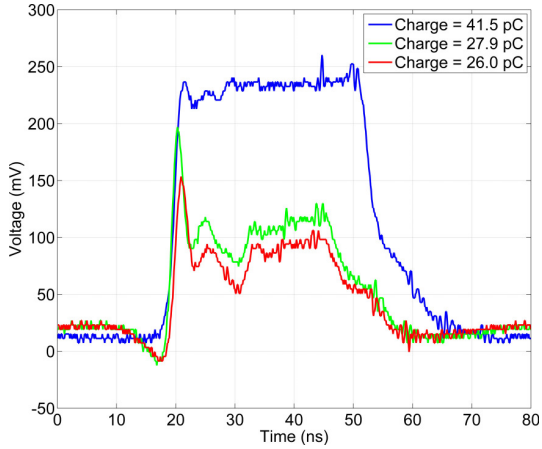


Figure 4: Detector's output signal.

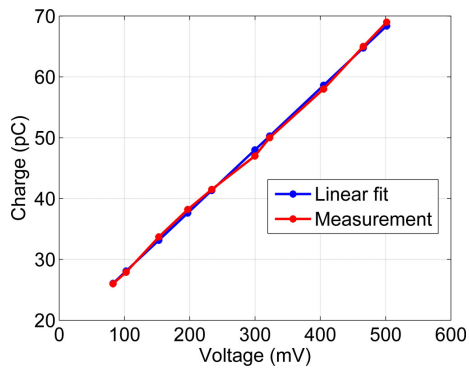


Figure 5: Characteristic of the bunch charge detection subsystem.

estimation error is 0.73 pC and mean absolute estimation error is 0.41 pC. The measurements prove the detection susbystem can be considered linear.

FIRMWARE

The charge detector output signal feeds the MTCA.4-based single cavity LLRF regulation system [2]. This system consist primarily of an Advanced Mezzanine Card digitizer SIS8300-L2[1] and an analog Rear Transition Module DRTM-DWC8VM1[2].

An outline of the controller's algorithm is given in Fig. 6. First, the RF feedback loop is described. Sampled IF signals from ADCs are fed into delay blocks, allowing coarse time alignment of different channels. In the next step IQ demodulation and calibration are performed. A programmable IIR filter is typically used as a low-pass or band-gap, limiting the detector noise and suppressing undesired spectral components like other pass-band modes of the cavity. The calculated error signal is fed to the feedback controller, the output of which is limited, before being combined with feed-forward signal. The drive signal is then scaled and rotated to compensate the slow drifts of the system, providing the controller with constant operating conditions. Finally, offset compensation is performed.

Raw ADC samples of the bunch charge detector output signal are fed into a block detecting the maximum and mini-

[1] http://www.struck.de/sis8300-l2.html
[2] http://www.struck.de/dwc8vm1.html

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

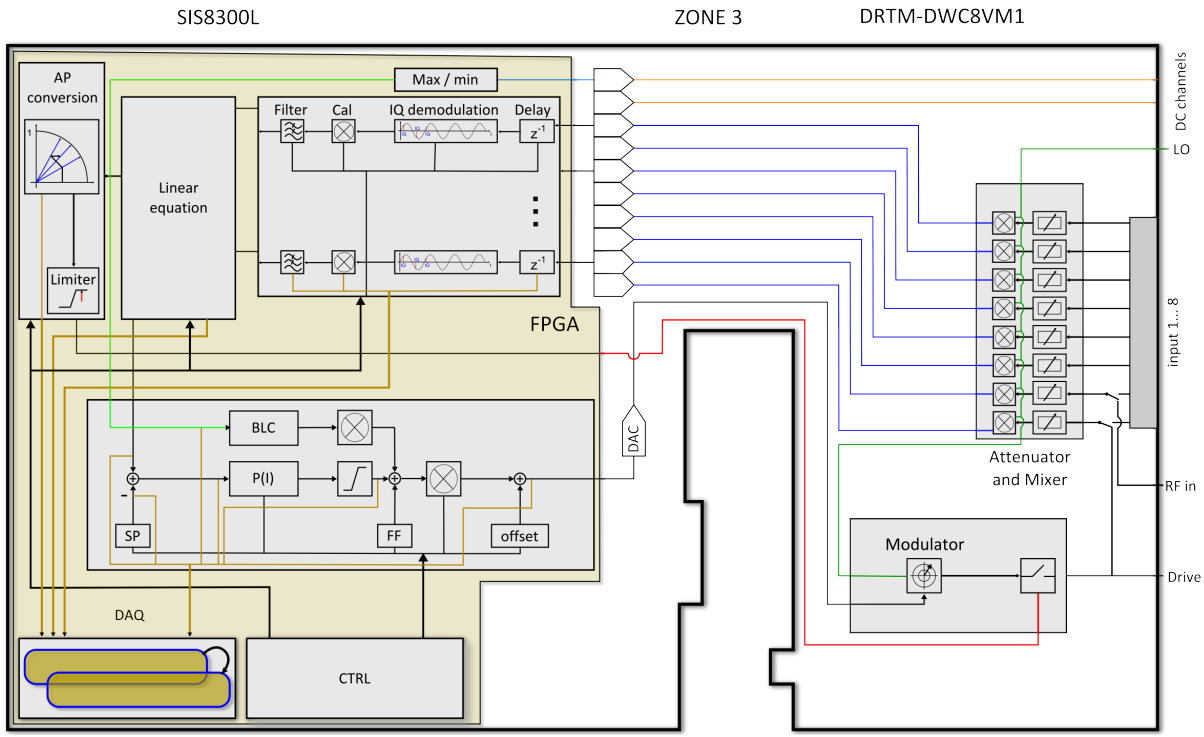


Figure 6: Hardware and firmware block diagram of the LLRF controls.

mum values. The amplitude (difference between max and min values) is multiplied by a gain coefficient. The next step is the rotation on the IQ plane. This signal is combined with feedforward and feedback signals before the output rotation.

TESTS

Measurements of the MTCA.4-based digital LLRF system using a PI feedback controller and the real-time BLC are shown in Fig. 7 (gain in arbitrary units). The beam pulse shape is similar to the one presented in Fig. 1 and 2.

Gain factor and rotation angle were chosen experimentally. The system response to beam rise is smooth, with no visible distortions. After the beam disappears, the RF field oscillates starting with the inital spike independent of the PI controller gain as visible in Fig. 7.

CONCLUSION

The addition of the real-time BLC to the LLRF system improves the system response to beam transients. Further development is needed to improve hardware (offset reduction by using different type of peak detector) and implement new algorithms in software (automated estimation of gain factor and rotation angle). The field oscillation at the end of the beam-body should be reduced.

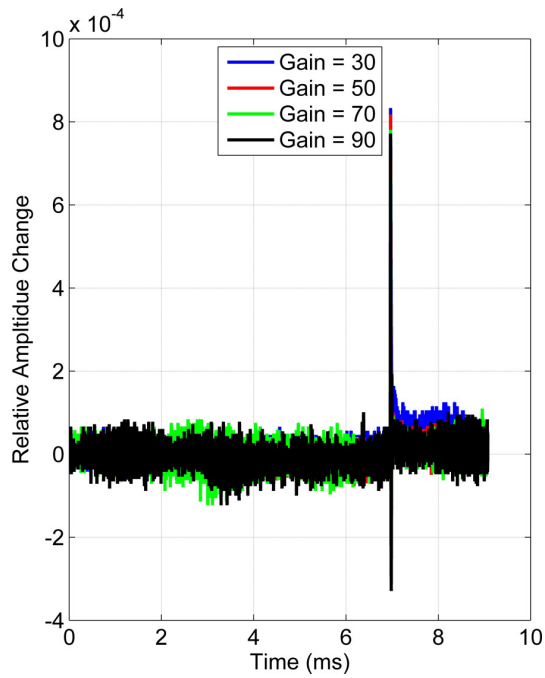


Figure 7: Response of a LLRF system using the PI feedback controller and the real-time BLC to a beam pulse.

REFERENCES

[1] E. Vogel, et al., “Beam loading compensation using real time bunch charge information from a toroid monitor at FLASH," in *Proc. of the 22nd Particle Accelerator Conf.*, Albuquerque, New Mexico, USA, pp. 2074-2076, June 2007.

[2] I. Rutkowski, et al., “MTCA.4-based digital LLRF control system for CW SRF Linacs," submitted for publication.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ARCHITECTURE OF TRANSVERSE MULTI-BUNCH FEEDBACK PROCESSOR AT DIAMOND

M.G. Abbott, G. Rehm, I.S. Uzun, Diamond Light Source, Oxfordshire, UK

Abstract

We describe the detailed internal architecture of the Transverse Multi-Bunch Feedback processor used at Diamond for control of multi-bunch instabilities and measurement of betatron tunes. Bunch by bunch selectable control over feedback filters, gain and excitation allows fine control over feedback, allowing for example the single bunch in a hybrid or camshaft fill pattern to be controlled independently from the bunch train. It is also possible to excite all bunches at a single frequency while simultaneously sweeping the excitation for tune measurement of a few selected bunches. The single frequency excitation has been used for continuous measurement of the beta-function. A simple programmable event sequencer provides support for up to 7 steps of programmable sweeps and changes to feedback and excitation, allowing a variety of complex and precisely timed beam characterisation experiments including grow-damp measurements in unstable conditions and programmed bunch cleaning. Finally input and output compensation filters allow for correction of front end and amplifier phasing at higher frequencies.

INTRODUCTION

At Diamond Light Source (DLS) we have been using a Transverse Multi-Bunch Feedback (TMBF) system based on the Libera hardware platform [1] for nearly a decade [2–5] to control multibunch instabilities. With a new bunch position every 2 ns it is appropriate for the feedback processing to be implemented in an FPGA. The original firmware was based on work done at the ESRF [6], but subsequently the firmware and control system have been rewritten to provide further extensions to the system functionality [7, 8]. This work has extended the lifetime of the original platform and increased its capability, and the DLS TMBF system has now been adopted at ALBA [9] and is being evaluated at the Australian Synchrotron.

In this paper we expand on and update the original IBIC 2013 [7] paper with many system refinements.

OVERVIEW

Figure 1 shows the TMBF processors in context. A 4-button electrode assembly together with an RF hybrid circuit picks up the horizontal and vertical position of each bunch. An RF front end converts this raw 3rd harmonic signal into a 250 MHz bandwidth bunch position signal suitable for processing by TMBF. This signal is digitised at 500 MHz (using four 125 MHz ADCs phased at 2 ns intervals), processed by the FPGA, and then reconverted to drive the striplines.

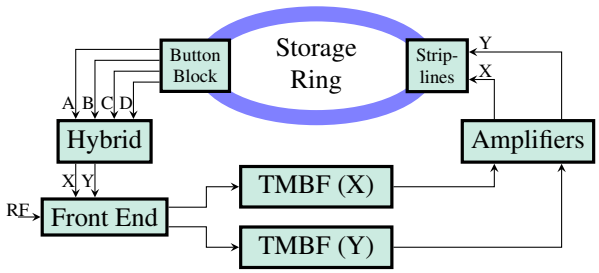


Figure 1: TMBF system in context. The Transverse Multi-Bunch Feedback system measures the position of each bunch, detects the betatron oscillations of each bunch, and generates a drive signal to suppress the oscillations.

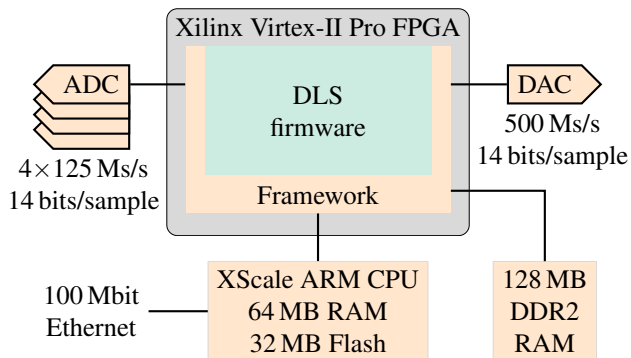


Figure 2: Libera System Platform. The DLS TMBF system is implemented on the Instrumentation Technologies Libera platform with the control system running EPICS on an ARM based embedded Single Board Controller (SBC).

The position of each bunch oscillates at the machine betatron tune, and these oscillations can normally be suppressed by feeding them back with a phase shift of 180°. The TMBF measures the horizontal or vertical position of each bunch, runs a 10-tap FIR filter on each bunch position to compute the appropriate phase adjustment, and outputs a negative feedback signal.

The TMBF processing system consists of analogue to/from digital converters connected to an FPGA for the high speed processing, together with memory for data capture and an embedded Single Board Computer as shown in Fig. 2.

As well as the core feedback function, a number of diagnostic functions are provided, both to enable easy monitoring of the system status, and to support complex experiments on the beam. System status monitoring includes detailed overflow detection and quick measurement of beam movement. Complex experiments are supported by internal oscillators, detectors, and a programmable sequencer, described in more detail below.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

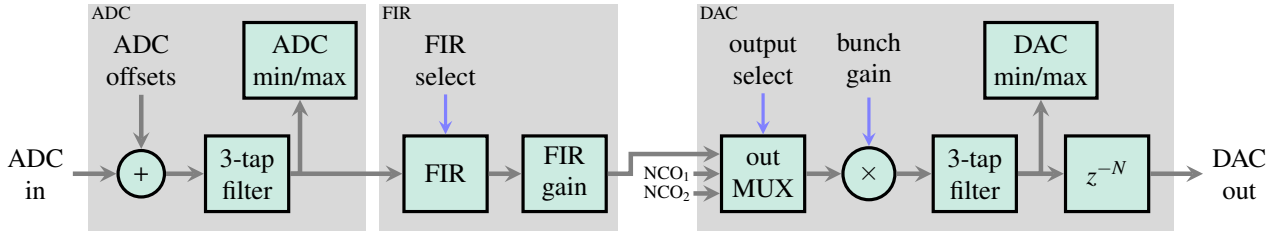


Figure 3: This is the core data processing chain in the TMBF processor. Data processing starts by adding a DC offset to each of the four ADC channels to compensate for static ADC errors, followed by a 3-tap filter to compensate for high frequency phase errors in the front end. The minimum and maximum value per bunch of both the ADC and DAC streams is captured for display. A 10-tap filter with programmable gain (in 6 dB steps) is applied in turn to each bunch in the ring. The output multiplexer adds any combination of its three inputs, which is then scaled by a bunch specific gain. Finally an output pre-emphasis filter corrects for amplifier errors and is followed by a delay line to correctly close the loop.

FEEDBACK DATA PROCESSING

The main function of TMBF is to stabilise transverse oscillations of the beam. This is done by running a separate 10-tap FIR on the position of each of the stored bunches. The core data processing chain shown in Fig. 3 combines this feedback with up to two optional Numerically Controlled Oscillator (NCO) outputs.

Note that numerical overflow can occur at any stage in this processing chain. When this is detected, a saturated output is generated and the resulting overflow event is reported to the user through the EPICS interface.

ADC Data In

The Libera platform uses four separate 125 MHz ADCs to sample the 500 MHz bunch position signal, and so the ADC input arrives as four parallel streams of 125 MHz data. The first processing step is to compensate for small DC offsets between the ADCs. These offsets are easily measured and compensated.

The next step is to compensate for gain differences and high frequency phase and gain errors in the front end. This is done by running a channel dependent 3-tap filter on the input data stream.

Finally a "min/max" subunit measures the minimum and maximum compensated value for each bunch; this is read out at 100 ms intervals and used to provide an accessible overview of bunch motion.

FIR Feedback Processing

Next a 10-tap FIR is run on the stream from each bunch. In practice, this means that 936 separate FIRs are run, with a delay of 936 bunches between each tap. The FIR unit can be programmed with four different filters, each of which can be separately selected for each bunch. The filter taps and final output gain are statically configured through the EPICS interface.

The ability to control bunches independently has two main uses: firstly, it can be used to apply a different filter to the isolated bunch in a hybrid fill; secondly, it can be useful as part of detailed machine physics experiments.

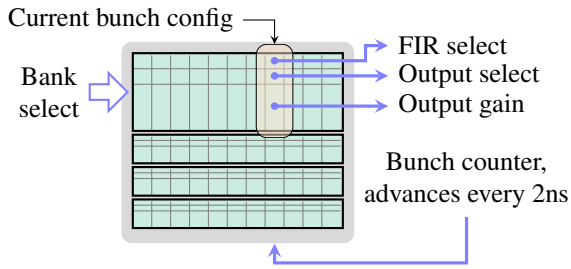


Figure 4: Bunch by bunch FIR and output control. The sequencer selects one of four banks to be active, defining basic machine behaviour. Each bank contains bunch configuration information for each of the stored bunches, used to control the feedback filter and the output configuration.

DAC Data Out

Finally the filtered signal is prepared for output. At this point three candidate output signals are selected and summed together: the FIR filtered signal and two internally generate NCO signals. Each of these three signals can be output or set to zero, and this control is per bunch.

Next the summed output is scaled by a bunch specific scaling factor. As this is signed it can be used to reverse the orientation of feedback on a single bunch.

Finally an output pre-emphasis filter is run to compensate for amplifier effects and the output is delayed so that the complete loop delay is a full machine revolution.

In the same way for ADC data, a "min/max" component measures the movement of each output bunch over a measurement interval of 100 ms, providing a quick view of bunch output motion.

Bunch by Bunch Control

As noted above, the FIR filter, the output sum, and the final output gain are selected per bunch. The "Bunch Select" unit, see Fig. 4, stores four arrays (or "banks") of bunch specific configurations — one configuration is selected by the sequencer unit (described below), and the current bunch configuration advances on each bunch. This configuration can be written through the EPICS interface.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

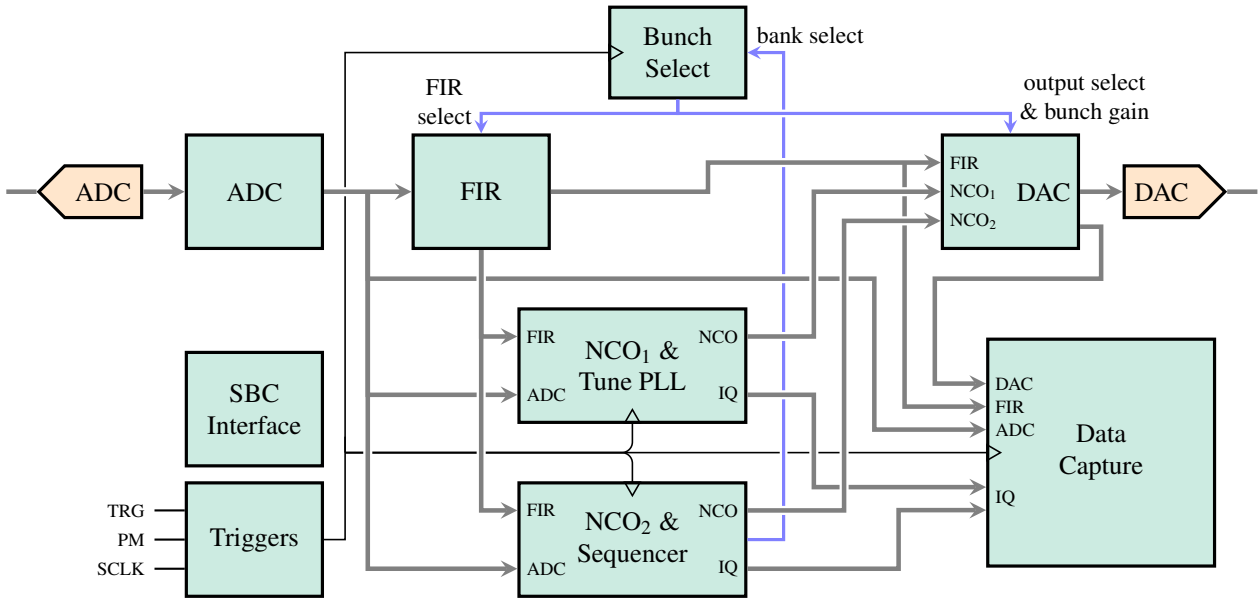


Figure 5: Here we see all of the major blocks of the FPGA system design and their data interconnections. FPGA blocks are shown thus: □ and analogue/digital converters thus: ▷. The main data flow is from the ADC, through the FIR with a separate FIR filter selected for each bunch, and out through the DAC with the option of adding up to two internally generated sine waves. The other paths are for control and data capture. The SBC interface controls and communicates with all other components of the system: the EPICS interface is through this component.

EXTRA DIAGNOSTIC DATA PROCESSING

Figure 5 shows the complete DLS TMBF firmware. The ADC-FIR-DAC chain has been described above. The remaining major components are for extra diagnostics and advanced machine experiments.

Data Capture

As already shown, min/max overview data is available for ADC input and DAC output. The Data Capture unit allows more detailed information: up to 4096 bunches of two out of three of DAC/FIR/ADC data streams can be captured, or up to 64 million bunches (more than 65,000 turns) of any one input can be captured. The shorter data capture is to FPGA block memory, the larger to external DDR2 RAM.

An alternative capture source is detector IQ data from either of the two NCO units. In particular, this is used for tune sweeps and more complex machine experiments.

NCO_1 and Tune PLL

Figure 6 shows the first NCO unit together with the tune tracking application (Tune PLL). When tune tracking is inactive the NCO generates a fixed frequency sine wave with programmable gain which can be used to drive selected bunches.

When tune tracking is active the detector (see Fig. 7 for the detector structure) measures the phase response of the beam at the currently driven frequency and runs a simple PI feedback loop to adjust the frequency to keep the phase response static. The frequency shift can be read as a continuous stream through the EPICS interface.

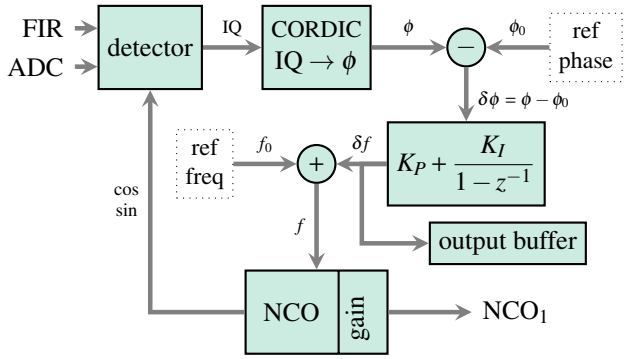


Figure 6: The oscillator NCO_1, can be set to a fixed frequency, or can be used as part of a tune tracking phase locked loop. The tune phase ϕ is measured at the operating frequency f and used to compute a sequence of frequency corrections δf to maintain the phase error $\delta\phi$ at zero. The reference phase and frequency are configured via EPICS.

This is very useful for measuring rapid changes in the machine betatron frequency, but will only work when the centre frequency and phase are already known to sufficient precision. Measuring this is the primary job of the next unit.

NCO_2 and the Sequencer

Figure 7 shows the second NCO unit together with the sequencer unit and a detailed view of the detector. The detector measures the complex response of the selected input to the driving frequency. The duration of each

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

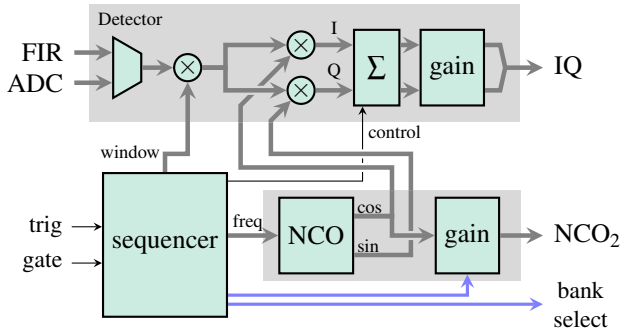


Figure 7: The oscillator NCO_2 is under control of the sequencer, and is only enabled while the sequencer is running a program. The sequencer controls the NCO frequency and gain and performs programmed frequency sweeps. Data from the beam is mixed with the excitation waveform in the detector to measure a complete complex IQ response. Operation of this system produces a waveform of IQ measurements which can be used for measuring betatron tune and other machine parameters.

measurement is controlled by the sequencer. The measured IQ value is scaled to a pair of 16-bit numbers for storage and transferred to the control system.

The TMBF sequencer controls the NCO_2 frequency, the NCO_2 output level, and which bunch bank is currently active. A machine experiment is performed by programming up to seven different control configurations and then triggering the start of the sequencer. The sequencer will then work its way through the programmed states, changing the output configuration as appropriate and capturing the appropriately programmed number of IQ samples, before finishing. Typically the sequencer and the data capture unit are triggered together.

For example, a standard tune sweep is performed using a single sequencer state which configures NCO_2 for output and steps through a preset range of NCO frequencies. Alternatively, a grow-damp experiment requires more states: one to excite the beam for a selected period, one to allow the beam to grow without feedback or excitation for a period, and one to restore feedback.

One final feature of the sequencer is the so-called "super-sequencer": this repeatedly performs a sequencer programme for a series of base frequencies. This is used to perform a grow-damp experiment on each transverse mode in turn, and allows the transverse mode damping times to be measured for all modes in a fraction of a second [8].

EPICS CONTROL SYSTEM

The control system running on the embedded ARM processor was developed concurrently with the FPGA firmware, and provides access to all settings and readings through around 560 EPICS PVs. A number of supporting scripts were written in Python and Matlab to help configure some of the more complex functions of TMBF.

A complete set of user interface screens built using EDM provides access to every PV published by TMBF. Around 255 PVs are used to configure settings, so the supporting scripts are important for normal operation.

The EPICS control system and the TMBF FPGA system are tightly coupled. Although the ARM processor is low powered and lacks hardware floating point support it is capable of some signal processing if care is taken. In particular, tune measurement is done by first performing a frequency response sweep with the FPGA, and then analysing the resulting IQ response and fitting a multi-pole tune response model. It has been possible to develop quite a sophisticated tune measurement process on this system.

CONCLUSIONS

The upgraded TMBF system described here has been in use at Diamond since late 2013 and is allowing us to perform very complex measurements to characterise the beam behaviour. This development has pushed the 10 year old FPGA technology to its limits, the FPGA now has little or no room for further developments.

Our next step at Diamond is to implement longitudinal bunch-by-bunch feedback in preparation for operation with normally conducting cavities. This will be based on a more modern FPGA on industry standard MicroTCA hardware.

REFERENCES

[1] Instrumentation Technologies, *Libera Bunch-by-Bunch*, `http://www.i-tech.si`.

[2] A.F.D. Morgan, G. Rehm, I. Uzun, *First Tests of the Transverse Multibunch Feedback at Diamond*, DIPAC 2007.

[3] A.F.D. Morgan, G. Rehm, I. Uzun, *Performance and Features of the Diamond TMBF System*, EPAC 2008.

[4] G. Rehm, M.G. Abbott, A.F.D. Morgan, J. Rowland, I. Uzun, *Measurement of Lattice Parameters Without Visible Disturbance to User Beam at Diamond Light Source*, BIW 2010.

[5] I. Uzun, M.G. Abbott, M.T. Heron, A.F.D. Morgan, G. Rehm, *Operational Status of the Transverse Multibunch Feedback System at Diamond*, ICALEPCS 2011.

[6] E. Plouviez, P. Arnoux, F. Epaud, J. Jacob, J.M. Koch, N. Michel, G.A. Naylor, J.-L. Revol, V. Serriere, D. Vial, *Broadband Bunch by Bunch Feedback for the ESRF using a Single High Resolution and Fast Sampling FPGA DSP*, EPAC 2006.

[7] M.G. Abbott, G. Rehm, I.S. Uzun, *Capability Upgrade of the Diamond Transverse Multibunch Feedback*, IBIC 2013.

[8] G. Rehm, M.G. Abbott, A.F.D Morgan, *New Features and Measurements using the Upgraded Transverse Multibunch Feedback at Diamond*, IBIC 2014.

[9] A. Olmos, U. Iriso, J. Moldes, F. Pérez, M. Abbott, G. Rehm, I. Uzun, *Integration of the Diamond Transverse Multibunch Feedback System at ALBA*, IBIC 2015.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

PANDA MOTION PROJECT - A COLLABORATION BETWEEN SOLEIL AND DIAMOND TO UPGRADE THEIR 'POSITION AND ACQUISITION' PROCESSING PLATFORM

I.S. Uzun, T. Cobb, A. Cousins, M. Heron, Diamond Light Source, Oxfordshire, UK
YM. Abiven, J. Bisou, P. Monteiro, G. Renaud, SOLEIL, Gif-sur-Yvette, France

Abstract

Synchrotron SOLEIL and Diamond Light Source are two third generation light sources located respectively in France and the UK. In the past 5 years, both facilities separately developed their own platform permitting encoder processing to synchronize motion systems and acquisition during experiments, SPIETBOX by SOLEIL and ZEBRA by Diamond. New operational requirements for simultaneous and multi-technique scanning, and support of multiple encoder standards have been identified by both institutes. In order to address these a collaborative project has been initiated between SOLEIL and Diamond to realize a new 'Position and Acquisition' processing platform, called PandA. The PandA project addresses current systems' limitations in terms of obsolescence and need for more processing power. Its design is going to be a 1U standalone system powered by a Xilinx Zynq SoC to implement a configurable set of logic functionalities. It will provide a flexible and open solution to interface different third party hardware (detectors and motion Controllers). This paper details the organization of this collaboration, sharing technical leadership between both institutes and the status of the project.

COLLABORATION MANAGEMENT

SPIETBOX [1] and ZEBRA [2] were designed in-house by SOLEIL and Diamond respectively, and both have been integral part of beamline operations, and proven useful for synchronising various equipment and for position capture. One application is to synchronize detectors with optics that select the photon energy of the beam to acquire only useful data, another one is facilitating the use of continuous scanning for faster acquisition in 2014. Both institutes share the same vision for a new motion control and data acquisition architecture. So SOLEIL and Diamond have gone into a collaboration to re-design and improve these platforms with following guidelines:

- Both institutes want to standardize hardware, in order to reduce number of systems which need to be developed and maintained by the support staff.
- Develop a ready to use and complete solution, in order to install quickly and easily all the hardware and software components.

The main goal of the collaboration was to share resources and experiences between both synchrotrons for the development. The first step was to understand existing skills and human resource availability at both institutes and agree on sharing the responsibilities of each task at design level as shown in Table 1. We decided to manage the work through regular video conference meetings every two weeks, face-to face meeting every 3 months, as well as sharing minutes and documentation over a private Github repository. Once the organization was set-up, a set of hardware, firmware and software specifications were defined as described in the following sections. It is agreed that the hardware developed during the course of the collaboration would be developed under the CERN Open Hardware (OHW) Licence [3] and released on the public OHW repository following completion of the project. Both parties would also have the option to engage third parties to commercialise the resulting project.

Table 1: Task Sharing Between Project Partners

Task leadership	DIAMOND	SOLEIL
Project management	X	
HW, FW specifications	X	X
Schematic Design		X
PCB Layout		X
Mechanics		X
FPGA – Zynq Processor Design	X	
FPGA – Zynq Logic Design	X	
Linux Kernel Development	X	
Linux Application Development	X	
TANGO interface		X
EPICS interface	X	

HARDWARE ARCHITECTURE

New applications [4], operational feedback from beamline scientists and from the Flyscan project [5] identified the following requirements to address limitations of existing SPIETBOX and ZEBRA solutions:

- Multi Channel TTL and LVDS I/Os for synchronous triggering and clocking with higher timing resolution.
- 4-Channels of encoder interface I/Os, supporting a wider range of protocols including Incremental, Absolute, EnDat and BiSS.
- 2-Channels of SFP Gigabit Transceiver interface for IpBus, Timing System, Diamond Communication Controller [6] or custom high-speed serial connectivity.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

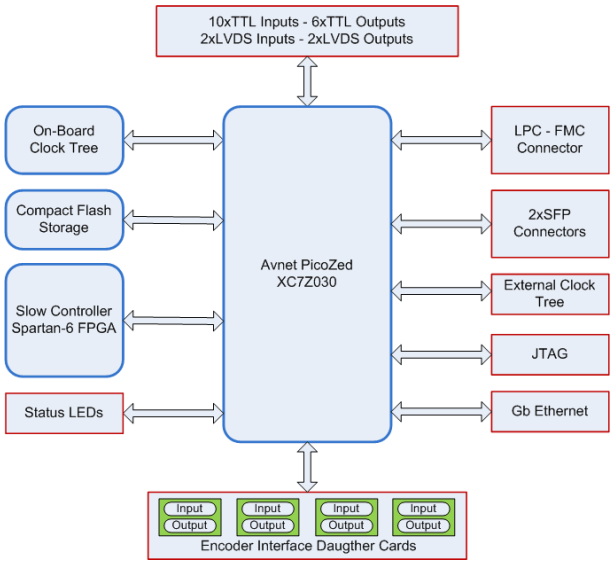


Figure 1: System Architecture.

- A fully compliant Low-Pin Count FMC slot for interfacing to analog and digital off-the-shelf boards or custom I/O modules.
- A Gigabit ethernet connectivity for control systems integration and high-speed data acquisition.
- A capable FPGA-SoC device for significant improvement of existing processing, programmable functionality and tighter system integration.

The proposed hardware architecture in Fig. 1 gives us a platform we can develop specific solutions with simple, reliable and durable tools. The platform is suitable in particular for applications where it is generally necessary to synchronize a range of equipment, which are managed by implementing the process at low level.

At the heart of the system architecture is the PicoZed [7] System-On-Module as shown in Fig 2. The PicoZed module contains Xilinx Zynq-7030 All Programmable (AP) SoC and the common functions required to support most SoC designs, including memory, configuration, ethernet, and clocks. It provides easy access to over 100 user I/O pins through three I/O connectors on the rear of the module.

A modular approach to the hardware design was taken by:

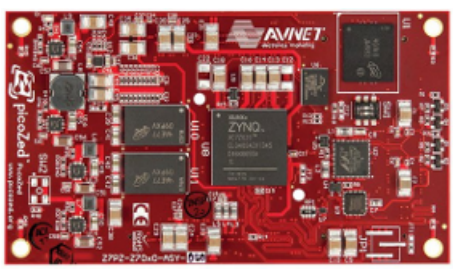


Figure 2: Avnet Picozed SOM.

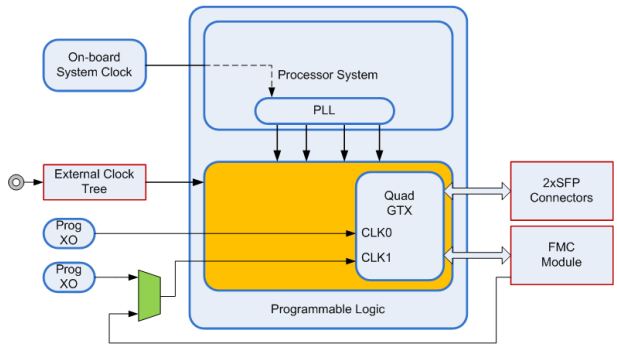


Figure 3: PandA Clocking Scheme.

- Using daughter cards for the encoder interface electronics permitting custom solutions, and to accelerate the development process. The current daughter card has been designed with pin-compatibility with the SOLEIL and Diamond motion controllers.
- Adding an industry standard fully-compliant Low Pin Count (LPC) FPGA Mezzanine Connector (FMC) to the architecture to enable analog I/O capability by using 3rd-party products [8] or custom modules.

In order to achieve the maximum flexibility that PandA can offer in terms of application support, a clocking architecture is built for clocking user programmable logic and Gigabit transceivers as shown in Fig. 3.

One of the drawbacks of modern programmable SoC devices is the limitation in terms of user I/O count. In our case, PicoZed module provides 135 user I/Os which is not enough to cover all the connectivity required by the specifications. To address this limitation, another lower-end FPGA device with multiple user I/Os is used to handle slow control functionality of various on-board devices. The communication between the main XC7Z030 Zynq device and slow control FPGA is achieved using an SPI-like custom interface.

PandA will consists of a 1U metal box, with BNC and LEMO connectors for single ended, differential signals and Gigabit Ethernet on the front panel. 15-way D-type connectors for RS485 encoder signals and power on the back panel.

FIRMWARE CAPABILITIES

PandA firmware functionality was designed to achieve full configuration flexibility and tight control system integration. The firmware implements a list of logical functions, a range of position compare functionality, high-throughput position capturing, multi-gigabit serial interconnections, and Arm processor interface.

Logic Operations and Position Based Blocks

A list of logical functions that had already been implemented by existing solutions was reviewed and following improved blocks were defined for PandA:

- Function generators to implement any 5-input boolean function.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

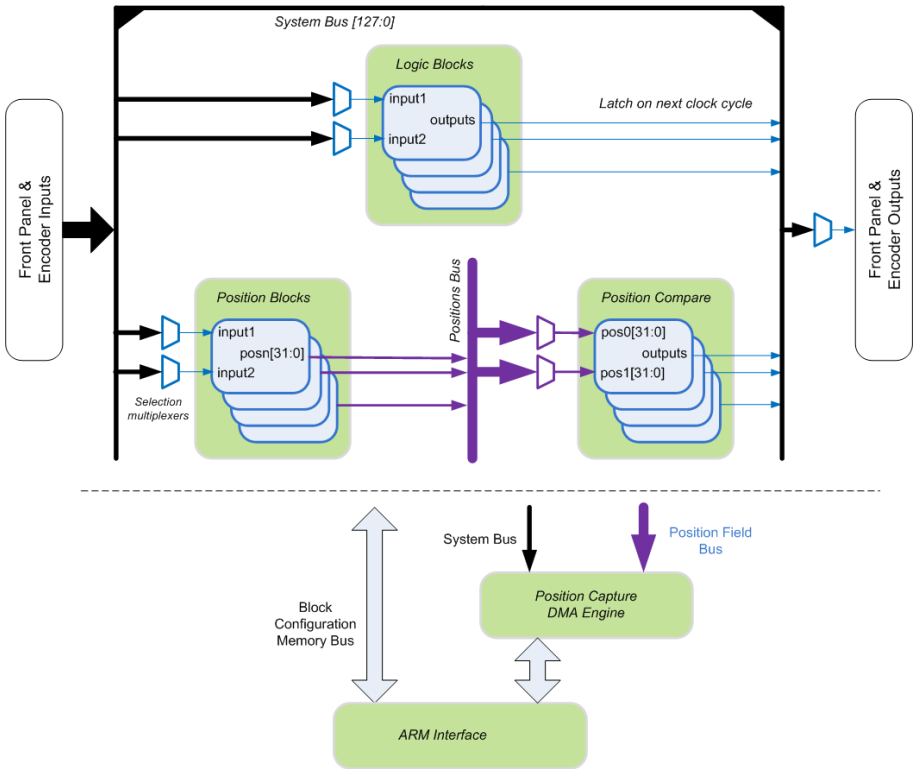


Figure 4: PandA Firmware Architecture.

- Set/Reset Gate blocks with configurable inputs, and an option to force its output independently.
- 32-bit Pulse divider with programmable divisor and two pulse outputs (divided and non-divided).
- Pulse Generator blocks to produce configurable width and delayed pulses triggered by its input and also able to handle pulse trains.
- Sequencer blocks to perform automatic execution of sequenced frames to produce timing signals.
- Programmable clock generators.

Similarly, a wide range of position based blocks are defined:

- Encoder Input/Output blocks supporting multiple encoder protocols.
- Quadrature Input/Output blocks to enable of any discrete signal on the system for quadrature operations.
- Counter/Timer blocks to implement multi-channel 32-bit up/down counters synchronous to system clock.
- Analog input/outputs upto 8-channels through FMC module connectivity.

To achieve connectivity between all design blocks, two internal buses are defined. The System Bus is a 128-bit wide register is composed of concatenating all physical I/Os, and registered discrete outputs from logic and position-based blocks. Similarly, all 32-bit outputs of the position based blocks are concatenated together to create the 1024-bit wide (32x 32-bit) Position Bus.

By connecting the input of any logic block to the system bus via a 128x1-bit multiplexer, it becomes possible to cascade multiple blocks to achieve the desired functionality. Each physical output is also taken from the system bus in the same way (see Fig. 4). Position Bus feeds all the 32-bit position inputs of all position compare blocks using 32x32-bit multiplexer so that position compare processing can be done on encoder, timer or analog input values.

Position Compare and Capture

The position compare blocks in PandA are designed to be able to generate a stream of trigger pulses, which can be position based, time based, analog input based or externally triggered. A new feature added to the position compare functionality is the ability to use LUT-based arbitrary arrays of position or time points instead of regularly spaced series of values. This facilitates:

- Position based capture – where output pulses are generated when the encoder position reaches a certain value, or series of values.
- Time based capture – where output pulses are generated at regular time intervals, storing the encoder positions as well.
- Analog input based capture – where output pulses are generated when analog input position reaches a certain value, or series of values.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- External trigger capture – where the motion controller stores the encoder position on an external trigger signal.

By cascading multiple position capture blocks, it is possible to mix modes together, such as outputting a time based pulse stream within a series of position based gates.

A central position capture block is responsible for capturing user-defined fields across the design including timestamps, encoder values, analog input values, counters and system status on trigger events. The captured data is DMA transferred into the system memory, and read out via gigabit ethernet interface.

Multi Gigabit Connectivity

The 3-channels of Multi-Gigabit Transceivers (MGTs) on the Zynq device are brought onto the front panel through SFP connectors. Along with the flexible clocking architecture in Fig. 3, these MGTs enable implementation of following interfaces on PandA for wider-level connectivity.

- Diamond Communication Controller: Light weight, low latency and high reliability data distribution.
- IPBUS: Simple, reliable, UDP/IP based protocol for controlling hardware devices.
- Micro-Research Event Receiver: to recover timing events and clocks distributed across the event network.

Processor Interface Logic

The processor interface logic handles communication with the host on Zynq-Arm Processor System for control and data acquisition purposes. The Zynq-Arm will run linux and use a dedicated linux device to tightly integrate with the FPGA logic. This will be through

- A dedicated address space on the host system memory enable direct access to each firmware block's configuration and status registers.
- A Write DMA master engine to transfer position capture data onto the system memory in real-time. Data rates up to 128MB/sec (capturing up to 32 position fields at 1MHz pulse rate) are envisaged.
- A Read DMA master engine for synchronous reading of user-defined arrays for position compare operation. The length of user defined arrays being limited to 1MSamples.

CONTROL SOFTWARE ARCHITECTURE

Figure 5 depicts standalone software architecture for PandA project. The kernel driver provides control and status register access to system blocks, and interface with the DMA engines for synchronous read/write data transfers. It is wrapped by a TCP Server that implements custom protocols over two channels for data and control interfacing. It is our intention that PandA will be integrated either with an EPICS IOC, a webserver interface, or with a separate TANGO server.

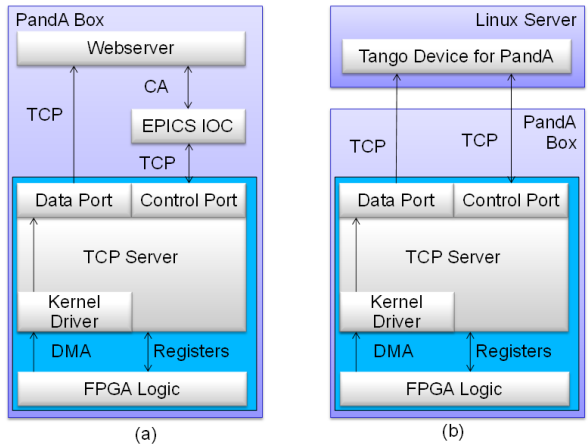


Figure 5: Software Architecture for (a) Diamond, (b) SOLEIL.

SUMMARY

Diamond and SOLEIL share the same vision of motion control architecture and have started collaboration for development of a common platform for Position and Acquisition of encoder sensors (PandA). Following agreement on the hardware, firmware and software specifications, both parties started design work mid-2015. Using commercial products, as much as possible, allows efforts to focus on applications requirements and saving time in development. The initial design and development phase is planned to be completed in the first quarter of 2016, with a provision of having prototype units during mid-2016.

REFERENCES

[1] Y.M. Abiven, et al., *SPI Boards Package, A New Set Of Electronic Boards at Synchrotron SOLEIL* , Proc. of ICALEPCS'2011, Grenoble (France).

[2] T. Cobb, Y. Chernousko and I. Uzun, *ZEBRA: A Flexible Solution for Controlling Scanning Experiments*, Proc. of ICALEPCS'2013, San Francisco (USA).

[3] CERN, *Open Hardware License*, `http://www.ohwr.org/projects/cernohl/wiki`.

[4] R.D. Walton, *Mapping Developments at Diamond*, THHB3O01, these proceedings, ICALEPCS'2015, Melbourne (Australia).

[5] N. Leclercq, at al., *Flyscan: a Fast, Simultaneous and Multitechnique Data Acquisition Platform for the SOLEIL Beamlines*, WEPGF056, these proceedings, ICALEPCS'2015, Melbourne (Australia).

[6] I. Uzun, et al., *Initial Design of the Fast Orbit Feedback System for DIAMOND Light Source*, Proc. of ICALEPCS'2005, Geneva (Switzerland).

[7] Avnet, *Picozed Product Brief*, `http://zedboard.org/product/picozed`.

[8] D-TACQ Solutions, *ACQ430FMC: 8 channel simultaneous sampling digitiser*, `http://www.d-tacq.com/modproducts.shtml`

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

UPGRADED CONTROL SYSTEM FOR LHC BEAM-BASED COLLIMATOR ALIGNMENT

G. Valentino*, G. Baud, M. Gasior, S. Jackson, L. Jensen, J. Olexa, S. Redaelli, J. Wenninger
CERN, Geneva, Switzerland

Abstract

In the Large Hadron Collider (LHC), over 100 movable collimators are connected to a three-tier control system which moves them to the required settings throughout the operational cycle from injection to collision energy. A dedicated control system was developed to align the collimators to the beam during machine commissioning periods and hence determine operational settings for the active run. During Long Shutdown 1, the control system was upgraded to allow beam-based alignments to be performed using embedded beam position monitors in 18 newly installed collimators as well as beam loss monitors. This paper presents the new collimation controls architecture for LHC Run II along with several modifications in the Java-based application layer.

INTRODUCTION

The Large Hadron Collider (LHC) accelerates two counter-rotating beams to an energy of 7 TeV before colliding them in four points where experiment detectors are located [1]. Uncontrolled beam losses can result in quenches of the super-conducting magnets, which can damage the machine. For this reason, a beam collimation system is installed in the LHC, comprising over 100 collimators [2]. It is also designed to protect the LHC in the event of fast failures, which might damage accelerator components.

Each collimator installed for ring cleaning consists of two jaws, made of graphite, tungsten or copper, which have to be positioned around the beam at all times with an accuracy of less than 50 μm. The operational settings of the collimators are defined in order to establish a four-stage hierarchy. They are determined via a beam-based alignment procedure [3] which uses feedback from a Beam Loss Monitoring (BLM) [4] detector positioned downstream from the collimator. From the aligned jaw positions, the measured beam centers and beam sizes can be calculated.

The primary collimators (TCP) are positioned closest to the beam, followed by the secondary collimators (TCSG), tertiary collimators (TCT) and absorbers (TCLA). Most of the collimators are installed in Insertion Region (IR) 3 and 7 for off-momentum and betatron cleaning respectively. The TCTs are installed upstream of the experiment detectors (IR1, 2, 5 and 8), and a TCSG is installed in the dump region (IR6).

The software architecture for the LHC collimation system respects the standards of the LHC Software Architecture (LSA) [5], which consists of a 3-tier structure. For the collimators, the bottom layer is composed of a double PXI system that is used to control and read out stepping motors, sensors and measurement devices. The collimator jaw positions can be positioned with an accuracy of 5 μm, i.e. less than 2% of the 1σ beam size at the primary collimators at 7 TeV. The maximum jaw movement rate is 2 mm/s.

Application servers that host databases and operational files make up the middle layer, on top of which Graphical User Interface (GUI) console applications run. The collimator positions can be set from a remote location in the CERN Control Centre (CCC) and synchronized with the operation of the LHC cycle. The software applications interact with the hardware via the Common Middleware (CMW) [6] and Front-End Software Architecture (FESA) [7] infrastructures.

During Long Shutdown 1 (LS1), all 16 TCTs and the 2 TCSGs in IR6 were replaced by a new design, in which Beam Position Monitor (BPM) pick-ups are embedded in the upstream and downstream corner of each collimator jaw [8, 9]. The LHC installation followed a successful validation of a prototype with beam in the Super Proton Synchrotron (SPS) [10]. This required an upgrade of the beam-based alignment control system to acquire and use both the BLM and the BPM data for the alignment.

ALIGNMENT PROCEDURES

BLM-based Alignment

A four-stage procedure is used to align the collimators with feedback from the BLMs. Both jaws of a reference collimator are moved towards the beam in steps of 5-20 μm (1) until they reach the beam halo on either side, which is established when an appropriate loss pattern is observed in a downstream BLM. Then, the collimator for which the beam center and beam size need to be measured is aligned (2). The center is given as the average of the two aligned jaw positions. The reference collimator is then re-aligned (3), and the beam size at the previous collimator is given as the ratio of its gap in mm when aligned to the average of the cut in units of beam σ. The final step is to open the collimator to the new operational positions (4). This procedure is performed at very small gaps of $< 4\ \sigma$.

One of the main drawbacks of this procedure is that it causes losses of a fraction of the circulating beam and therefore can only be carried out with safe beam intensities. Collimator alignment can therefore not be performed dur-

* gianluca.valentino@cern.ch

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ing standard operation. Due to the time needed to complete an alignment of the full system (∼4 hours), which needs to be repeated for different subsets of collimators at different points in the machine cycle, the system performance relies on machine and collimator setting stability and reproducibility from fill to fill.

BPM-based Alignment

The beam position between two BPM electrodes can be calculated using a well-known linear technique. In a simple 2D approximation of a BPM arrangement, which consists of a circular beam-pipe and two point-like electrodes located 180° apart on a horizontal axis, the approximate position of a charged particle, denoted as X_{bpm}, is calculated from the distance between the opposite BPM electrodes B and the induced potential V_1 and V_2 on opposite BPM electrodes:

$$X_{bpm} = \frac{B}{4}\frac{V_1 - V_2}{V_1 + V_2} \quad (1)$$

The objective of the alignment is to minimize X_{bpm}. A successive approximation algorithm was developed to automatically align the collimator jaws around the beam axis from any starting jaw gap and beam offset [11]. The left and right jaws are moved towards and away from the beam respectively, or vice-versa, depending on the sign of the beam offset. Several iterations are needed due to non-linearities inherent in the BPM geometry. Contrary to the BLM-based method, where the jaws touch a reference halo, this technique does not provide a measurement of the beam size at the collimator. However, the measured beam sizes are only used to calculate the operational settings at injection, as the error introduced by the β-beat tends to be more than the alignment error due to the jaw step size, while at top energy the nominal beam sizes are used as the beam sizes shrink. BPM-based alignment is more useful at top energy in terms of machine commissioning efficiency, as the majority of the machine configuration changes throughout the run take place there.

ALIGNMENT SOFTWARE ARCHITECTURE

LHC Run 1 Architecture

During Run 1, all collimators were aligned using the BLM-based technique as the embedded BPM collimators were not installed. BLM data was transmitted at 12.5 Hz from the crates in the tunnel via UDP to a Linux server, which swallowed or forwarded the packets depending on whether a Java client GUI subscribed to the data. The header of each UDP packet contained the IR and the position within the IR (left, right, center) of the BLM crate. The payload consisted of the data arranged in a 16 × 16 2D array. The data are in integer format, and were then converted to units of Gy/s. The collimator jaw positions were set and read out via a FESA class called *LHCCollimator* [12]. A feedback loop was implemented in the Java GUI, which stopped the jaw movement when the losses exceeded a pre-defined threshold in Gy/s [13].

A prototype collimator with embedded BPMs was installed in the SPS in Run 1 for beam tests. This architecture was re-used for the BPM data acquisition, which is provided by electronics based on compensated diode detectors (called DOROS) [14]. By taking measurements over thousands of turns, sub-micrometer resolution is achieved. The BPM-based alignment algorithm was implemented in the Java GUI to make it easier to modify during the beam tests.

LHC Run 2 Architecture

In LS1, the Run 1 control system was upgraded to allow both BLM-based and BPM-based collimator alignment to be performed in the same software module. The new software architecture is shown in Fig. 1. The functionality previously present in two separate Java applications was moved to a new FESA class (called *LHCCollAlign*) running on a Front-End Computer (FEC). The Run 1 UDP packet format was reused to allow the BLM crates to send the 12.5 Hz data to the FEC. *LHCCollAlign* acts as a BLM concentrator and combines the 12.5 Hz data from all 27 BLM crates for logging purposes.

The DOROS electronics was upgraded from the SPS beam tests with automatic gain control to ensure that the signals remain within a fixed range independently of the BPM aperture and beam intensity in the LHC. In addition, asymmetries between two opposite BPM pick-up channels are corrected online by means of a switching mechanism, in which the signals A and B from two opposite electrodes are connected to respective channels A and B or to channels B and A. An averaged, calibrated beam position measurement is then provided at 1 Hz by a second FESA class (*BPMCOL*), which receives the data from the DOROS boxes via UDP and send back control messages to the DOROS boxes (therefore acting as a DOROS controller). The beam positions (X_{bpm}) at each collimator are calculated based on the electrode signals and the BPM aperture. The aperture is calculated by *LHCCollAlign* from the upstream and downstream jaw positions of the 18 BPM-equipped collimators, and a constant offset which is the retraction of the BPM pick-up button with respect to the jaw surface. This is then sent to BPMCOL at 1 Hz, which is the readout rate from *LHCCollimator*. All collimators can be concurrently aligned using either of the two techniques.

The BLM-based feedback loop is always active, meaning that should high losses occur during the BPM-based alignment, when the collimator is supposed to be far from the beam, the jaw movement is stopped. The software architecture is designed to be easily extensible should further collimators be equipped with BPMs in future LHC runs. Once sufficient experience is gained with the system, the existing collimation hierarchy margins in the TCSPs and TCTPs (placed to account for beam orbit drifts) can be re-

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

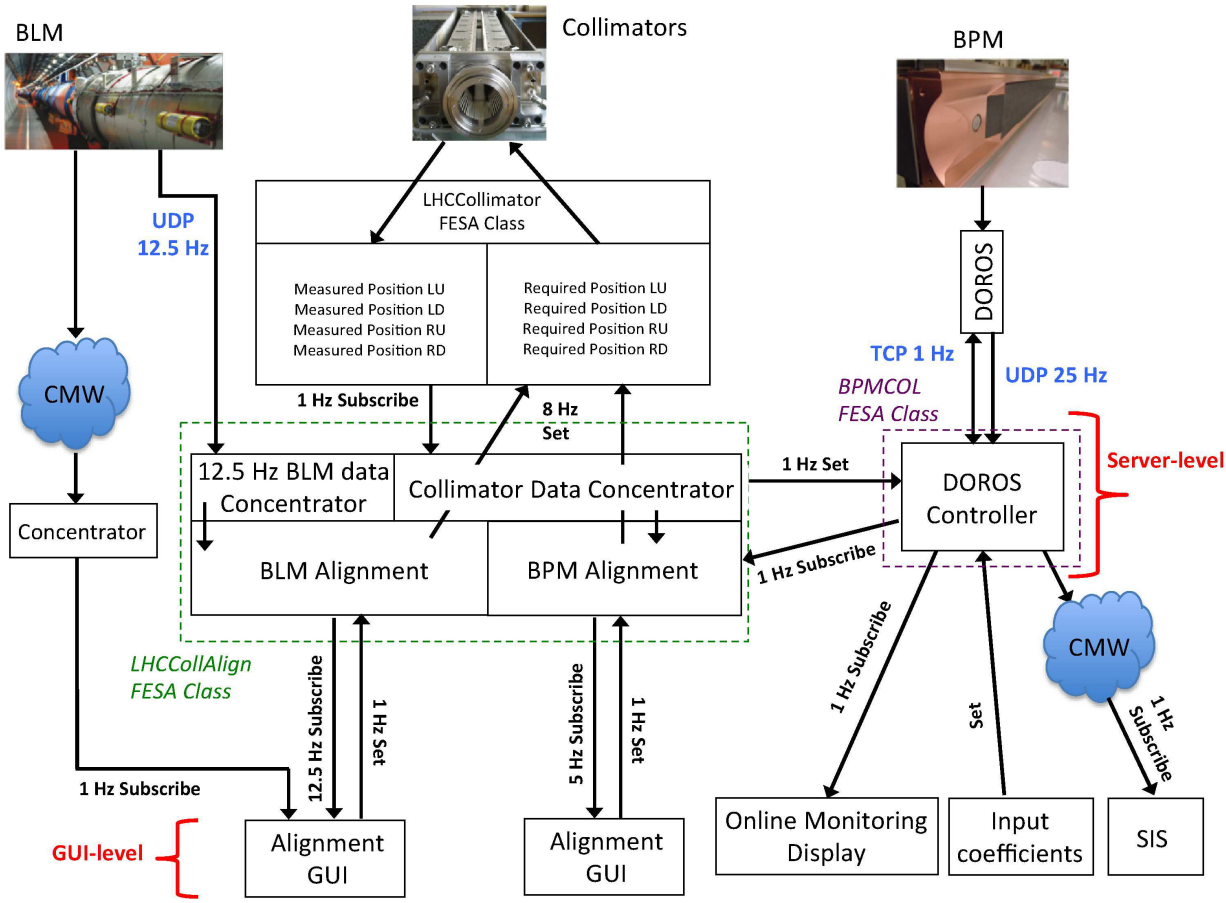


Figure 1: The software architecture for the BPM and BLM data acquisition and collimator alignment.

duced [15]. The beam position measurements would be sent to the LHC Software Interlock System [16] to dump the beam in case of orbit drifts above a certain threshold.

USER INTERFACES

Two dedicated Java GUI applications are used for the BLM-based and BPM-based alignment. The BLM alignment GUI also subscribes to both the standard 1 Hz and 12.5 Hz data for display purposes, and is identical to that used for Run 1 [17]. The user can input 4 parameters, namely the left and right jaw step size, the time interval between each step, and a stopping BLM threshold. On the other hand, the BPM alignment GUI requires a value for X_{bpm} below which the successive approximation algorithm stops, and a time interval between each step. A minimum gap value is also required to ensure that the collimator jaw does not inadvertently move too close to the beam.

An online display was developed to provide monitoring of the beam orbit in all embedded BPM collimators. A screenshot of the GUI for one collimator is shown in Fig. 2. The upstream and downstream beam positions relative to the collimator center are represented by circles. The half-width of the box represents 1 σ, and the intermediate lines indicate an editable threshold in mm, which is then converted to units of σ. The circles turn red if they exceed this

Figure 2: Subview of the monitoring display showing the up and downstream beam positions at one collimator.

limit. The non-linearities inherent in the BPMs can be corrected for by performing a 2D polynomial fit to a series of measured beam positions at different collimator gaps and offsets [11]. The fit coefficients can be sent to *BPMCOL* via a dedicated GUI.

ALIGNMENT RESULTS

The software architecture was tested during a SPS beam test and in the LHC during the beam commissioning period at the start of Run 2. An example of a BPM-based collimator alignment is shown in Fig. 3. A total of 12 iterations were needed to complete the alignment. Initially, both jaw corners are moved in parallel until the upstream electrode signals are equalized. Then, a tilt is gradually introduced in the jaws until the downstream electrode signals are also equalized. All collimators were aligned simultaneously in 15-25 seconds, which is a remarkable speed-up over the ~1.5 hours required to align the same collimators with the BLM-based technique.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

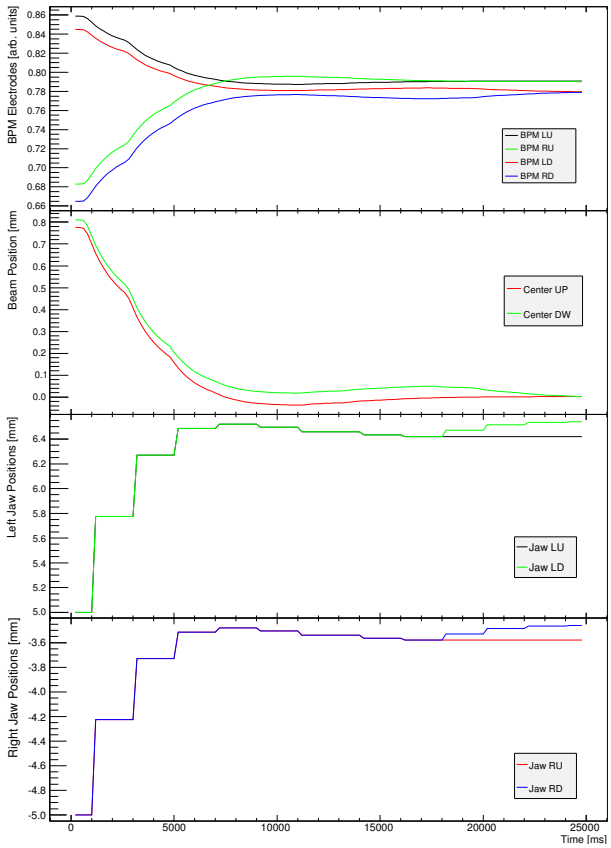


Figure 3: BPM-based collimator alignment.

CONCLUSION

Beam-based alignment of the LHC collimators is used to determine the jaw settings needed for operation. During Run 1, all collimators were aligned with a beam loss feedback algorithm, which ensured that the jaws stopped moving when the beam halo was reached. This was performed using a Java GUI application. The replacement of 20% of the system with embedded BPM collimators required a controls software upgrade. A new middleware layer was developed in FESA to ensure that the BLM and BPM data acquisition could be performed reliably for the given real-time constraints, and the separate alignment techniques could be performed in the same software module. The next steps will involve a thorough fill-to-fill analysis of the BPM data to determine whether the present collimation hierarchy margins to account for orbit drifts can be removed, and orbit interlocks could instead be put in place. This would allow to reduce the β^* and therefore extend the luminosity reach of the LHC.

REFERENCES

[1] Report No. CERN-2004-003-V1, edited by O. S. Brüning, P. Collier, P. Lebrun, S. Myers, R. Ostojic, J. Poole, P. Proudlock, 2004.

[2] R. W. Assmann *et al.*, "Requirements for the LHC collimation system". In Proceedings of EPAC'02, Paris, France, pp. 197-199.

[3] G. Valentino *et al.*, "Semiautomatic beam-based LHC collimator alignment", Phys. Rev. ST Accel. Beams, **15** 015002 (2012).

[4] E. B. Holzer *et al.*, "Beam loss monitoring system for the LHC". In Proceedings of the IEEE Nuclear Science Symposium Conference Record, San Juan, Puerto Rico, 2005, pp. 1052-1056.

[5] G. Kruk, S. Deghaye, M. Lamont, M. Misiowiec, W. Sliwinski, "LHC software architecture (LSA) - evolution towards LHC beam commissioning". In Proceedings of ICALEPCS'07, Knoxville, Tennessee, USA, pp. 307-309.

[6] K. Kostro, V. Baggiolini, F. Calderini, F. Chevrier, S. Jensen, R. Swoboda, N. Trofimov, "Controls middleware - the new generation". In Proceedings of EPAC'02, Paris, France, 2002, pp. 2028-2030.

[7] M. Arruat *et al.*, "Front-end software architecture". In Proceedings of ICALEPCS'07, Knoxville, Tennessee, USA, 2007.

[8] A. Dallocchio *et al.*, "LHC collimators with embedded beam position monitors: a new advanced mechanical design". In Proceedings of IPAC'11, San Sebastian, Spain, 2011.

[9] B. Salvachua *et al.*, "Collimation system post-LS1: status and commissioning". In Proceedings of the 5th LHC Beam Workshop, Evian, France, 2014.

[10] D. Wollmann *et al.*, "Beam feasibility study of a collimator with in-jaw beam position monitors", Nucl. Instr. Meth. Phys. Res. A, Vol. 768, pp. 62-68 (2014).

[11] G. Valentino, A. Nosych, R. Bruce, M. Gasior, D. Mirarchi, S. Redaell, B. Salvachua, D. Wollmann, "Successive approximation algorithm for beam-position-monitor-based LHC collimator alignment", Phys. Rev. ST Accel. Beams, **17** 021005 (2014).

[12] A. Masi, R. Losito, "LHC collimators low level control system". IEEE Trans. Nucl. Sci., vol. 55, no. 1, pp. 333-340, 2008.

[13] G. Valentino, R. W. Assmann, R. Bruce, S. Redaelli, N. Sammut, "Automatic threshold selection for BLM signals during LHC collimator beam-based alignment". In Proceedings of UKSim/AMSS EMS'12, Valletta, Malta, pp. 210-213.

[14] M. Gasior *et al.*, "BPM electronics based on compensated diode detectors - results from development systems". In Proceedings of the 2012 Beam Instrumentation Workshop, Newport News, VA, USA, 2012.

[15] R. Bruce, R. W. Assmann, L. Lari, S. Redaelli, "Collimator hierarchy limits: assumptions and impact on machine protection and performance". In Proceedings of the Machine Protection Workshop, Annecy, France, 2013.

[16] J. Wozniak, V. Baggiolini, D. Garcia Quintas, J. Wenninger, "Software interlocks system". In Proceedings of ICALEPCS'07, Knoxville, Tennessee, USA.

[17] G. Valentino, R. W. Assmann, S. Redaelli, N. Sammut, "LHC collimator alignment operational tool". In Proceedings of ICALEPCS'13, San Francisco, CA, USA, 2013.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

HIGH LEVEL CONTROLS FOR THE EUROPEAN XFEL

Lars Fröhlich, Bolko Beutner, Winfried Decking, Olaf Hensler, Raimund Kammering,
Torsten Limberg, Sascha Meykopff, Josef Wilgen, DESY, Hamburg, Germany

Abstract

The European X-Ray Free-Electron Laser (XFEL) will generate extremely short and intense X-ray flashes from the electron beam of a 2.1 km long superconducting linear accelerator. Due to the complexity of the facility and the sheer number of subsystems and components, special emphasis needs to be placed on the automatization of procedures, on the abstraction of machine parameters, and on the development of user-friendly high-level software for the operation of the accelerator. The paper gives an overview of the ongoing work and highlights several new tools and concepts.

INTRODUCTION

The European X-ray Free-Electron Laser (XFEL) is a research facility currently under construction in close collaboration between the European XFEL Facility GmbH[1] and DESY[2] in Hamburg, Germany [1–3]. It consists of a superconducting linear accelerator delivering an electron beam with particle energies up to 17.5 GeV and a total beam power up to ~600 kW, several long undulator sections enabling the generation of extremely brilliant X-ray pulses at wavelengths down to 0.05 nm, beamlines and setups for photon science experiments, and the associated infrastructure. An overview of the control system architecture is given in [4].

Compared to other accelerator facilities operated by DESY, the XFEL presents itself as a system of unprecedented complexity. It will generate ~600 µs long *macropulses* of up to ~3000 bunches at a repetition rate of 10 Hz. Individual bunches out of these trains will then be distributed to several undulator beamlines and beam dumps using a system of fast intra-train kickers. Even the sheer number of subsystems and components—more than 1400 electromagnets for beam guidance and focusing, more than 450 beam position monitors, to pick two popular examples—makes it clear that a high degree of automatization will be needed to operate the machine efficiently. Control software should offer high level abstractions for all important machine parameters rather than forcing operators to work at the level of individual technical subsystems.

We have formed an interdisciplinary team of physicists, engineers, and computer scientists from multiple DESY groups with the goal of establishing the best possible operability for the machine. The scope of this *high level controls* (HLC) effort is a deliberately broad one. It includes the preparation of user friendly tools for various physical or technical tasks in the control room as well as the development of feedbacks and of middle layer servers. A lot of work is invested to improve the entire software stack of libraries, toolkits, and network services. This paper gives an overview of the various projects and activities, starting at the infrastructure level and closing with a discussion of stand-alone tools. For obvious reasons, we limit ourselves to the description of a few selected examples for each category.

[1] European X-Ray Free-Electron Laser Facility GmbH, Albert-Einstein-Ring 19, 22761 Hamburg, Germany
[2] Deutsches Elektronen-Synchrotron DESY, Notkestraße 85, 22607 Hamburg, Germany

PROTOCOLS AND INFRASTRUCTURE

Three main control system protocols are used at the XFEL: DOOCS [5, 6], Karabo [7], and TINE [8, 9]. Much work has been invested to improve the interoperability of these protocols [10], so that the problem of network communication across protocol boundaries is mostly solved. For HLC applications, efforts have concentrated on the development and improvement of interfaces for Matlab [11] and Python.

A *configuration database* has been set up as a central network service using DESY's Oracle 12c database system. It stores a complete list of beamline components and associated information such as calibration data. In this way, inconsistencies in the configuration of distributed servers can be minimized [12].

Finally, a *virtual accelerator* has been created by duplicating most of the middle layer and data acquisition software of the XFEL and exchanging the real front-end servers with dummies generating simulated data. This virtual XFEL has proven invaluable for the testing of high level software [13].

LIBRARIES AND TOOLBOXES

A multitude of libraries has been created to support the development of middle layer servers in C++. These projects include generic class libraries for access to various accelerator components, collections of algorithms for numerical tasks and fitting routines, classes for image analysis and conversion, an easy-to-use interface for retrieving information from the central configuration database, and various utility functions.

Matlab has traditionally been the main high-level interpreted language for physics applications at DESY. A broad assortment of utility functions is therefore available and actively maintained; a report on the DataGUI library for the design of advanced graphical user interfaces is found in [14]. We are currently focusing on the implementation of Matlab classes for access to specific accelerator components.

Recently, Python has been introduced as a second high-level language. Its main application at the moment is the rapid development of graphical user interfaces using the QT framework. Toolkit and library support is still fragmentary, but expanding continuously.

In general, we are encouraging the use of modern language standards where possible. For example, big parts of our C++

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

library code use C++11 and all of our Python developments are based on version 3.4 of the language.

MIDDLE LAYER SERVERS

The XFEL is equipped with lots of front-end servers that are distributed all over the accelerator, undulator, experimental, and infrastructure buildings. These front-ends establish the interface between hardware components and the controls network, fulfilling functions such as data acquisition from ADCs, communication with field buses, controlling actuators and the like.

Above this level, another system of servers pre-processes data and offers advanced functionality. In contrast to the distributed front-ends, these *middle layer servers* are running on only a few powerful computers in central locations. Among other considerations, this approach allows data to be streamed and processed via a powerful shared memory based data acquisition (DAQ) system in a fully synchronous manner [15]. A major part of the HLC effort is focused on the development of such middle layer servers, with the following goals:

More physics: We are continuously integrating more physical models directly at the server level. For example, magnets can not only be controlled via their power supply current, but also through the setting of deflection angles or fields under automatic tracking of hysteresis effects. Radiofrequency (RF) stations can not only be controlled via amplitude and phase, but also by specifying a desired energy chirp on the electron bunch. Even a complete optics model with matching capabilities is provided by an optics middle layer server [16].

More automatization: Time-consuming and complex procedures should be automatized wherever possible. For example, state machine based middle layer servers handle the startup, restart, and shutdown of RF stations along with the conditioning of couplers. Another server can perform a phase scan of the RF gun and determine the optimal phase for operation.

More online measurements: The implementation of measurement procedures at the server level makes it possible to provide a bunch-synchronous measurement of beam parameters like energy, slice emittance, or bunch length.

More uniformity: Some subsystems consist of device families with quite different interfaces. Middle layer servers present a uniform interface for all devices with a specific functionaliy—for example, the beam profile server can measure bunch profiles using scintillating screens or wirescanners, and the charge server will integrate readings from toroids and from beam position monitors (BPMs).

Less load: Because applications retrieve their data from the middle layer, there is less network and CPU load on the front-end side where infrastructure and computing resources are costly.

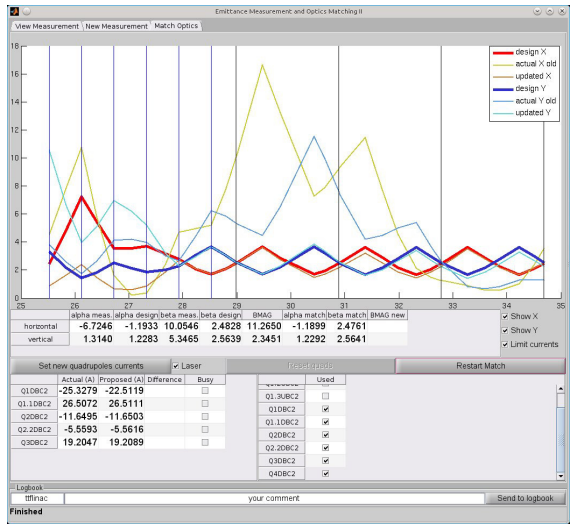


Figure 1: Screenshot of the *Emittance & Matching* tool while matching the beam optics to a periodic lattice.

More jddd: The Java DOOCS Data Display *jddd* [17, 18] is the main user interface builder used for constructing control system panels at DESY. It allows the deployment of standardized, powerful graphical interfaces with a minimum of effort. A lot of functionality that would traditionally have to be implemented as a full graphical application can therefore be integrated at the server level and use jddd as a user interface.

Feedbacks

Feedbacks are of enormous importance for the stability of the machine. Apart from a few very fast regulation loops acting within a single macropulse (at time scales of tens of microseconds), almost all beam-based feedbacks are implemented as middle layer software: Trajectory, bunch compression, charge, energy, and more parameters are stabilized by DAQ-based servers. Apart from the stabilization aspect, feedbacks are also important because they enable direct control of the quantity of interest—beam positions are adjusted instead of correction coil currents, beam energies instead of RF amplitudes, and so on.

APPLICATIONS

As mentioned before, a powerful platform for the rapid deployment of graphical control system interfaces, *jddd*, is available for the operation of the XFEL. jddd panels cover the vast majority of use cases for applications in the control room, both at the operator and at the expert level. A few cases nonetheless require the development of specialized stand-alone software—when the functionality is too complex or too seldomly used to be implemented in a server, when very specific graphical user interfaces are needed, or when the implementation in a high-level language offers other advantages such as the availability of skilled developers.

While almost all middle layer servers are written in C++ and deployed on Linux systems, applications should gener-

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

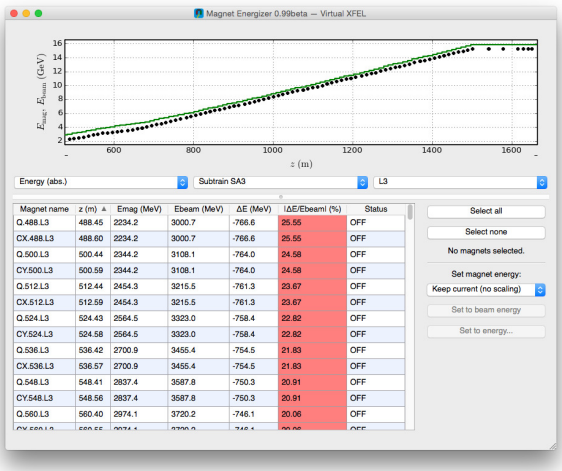


Figure 2: Screenshot of the *Magnet Energizer* tool for the scaling (setting of nominal particle energies) of magnets.

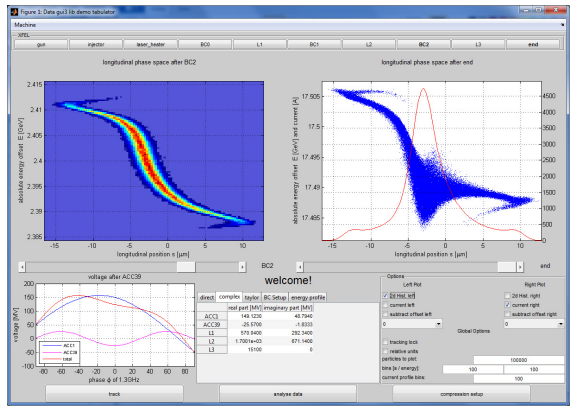


Figure 3: Screenshot of the *RF Tweak* tool for the calculation of longitudinal bunch profiles from RF settings.

ally run on all of our three supported desktop platforms—MacOS, Windows, and Linux. Therefore, platform-independent development in high-level interpreted languages such as Matlab or Python is preferred. Java Web Start plays an important role in the platform-independent launching of such tools, and many low-level control system utilities are implemented directly in Java. Like the jddd panels, most high level applications rely heavily on the services of the middle layer server system.

The range of applications available and under development is quite broad and very similar to those used at other high gain free-electron laser facilities: To name only a few examples, an *Emittance & Matching* tool supports the measurement of the beam emittance with the 3-/4-screen method, calculates the optical functions and matches them to the design optics (Fig. 1); the *Magnet Energizer* compares the measured beam energy with the settings of magnets and adjusts them accordingly (Fig. 2); *RF Tweak* functions as a graphical frontend to a fast tracking code and can calculate longitudinal bunch profiles from RF settings and optimize bunch compression settings, taking into account radiative, wakefield, and space charge effects (Fig. 3).

CONCLUSION AND OUTLOOK

The European XFEL is a machine of considerable complexity, not only due to its sheer size and number of single components, but also because of the intricacies of its pulsed mode of operation, of its beam distribution system, and of many subsystems. To ensure the best possible operability of the entire accelerator, we are advancing high level controls developments from multiple sides:

- through the preparation of libraries, toolkits and configuration databases (at the infrastructure level)
- by providing more physical values and more automatized procedures in the control system (at the server level)
- by providing powerful and easy-to-use graphical user interfaces for tasks in the control room (at the application level).

A big part of our software could be tested in the *Virtual XFEL* environment, and as many developments as possible were ported back to the FLASH [19–21] user facility, which is in many ways the predecessor of the XFEL. Experience has also been gained from the first operation of the XFEL gun with beam in February 2015. All of this makes us look forward to the injector commissioning at the end of this year with confidence.

REFERENCES

[1] A. Aghababyan et al., "The European X-Ray Free-Electron Laser technical design report", DESY 2006-097, DESY, Hamburg, Germany (2007).

[2] W. Decking et al., "European XFEL post-TDR description", XFEL.EU TN-2013-004-01, European XFEL GmbH, Hamburg, Germany (2013).

[3] W. Decking et al., "Status of the European XFEL", WEA04, FEL'15, Daejeon, Republic of Korea (2015).

[4] A. Aghababyan et al., "The large scale European XFEL control system: Overview and status of the commissioning", MOA3O02, *These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

[5] A. Aghababyan et al., "The accelerator control system at DESY", in: ICFA Beam Dynamics Newsletter 47 (2008), pp. 139–167.

[6] The DOOCS website, http://doocs.desy.de/.

[7] B. C. Heisen et al.,"Karabo: An integrated software framework combining control, data management, and scientific computing tasks", ICALEPCS'13, San Francisco, USA (2013), pp. 1465–1468.

[8] P. K. Bartkiewicz and P. Duval, "TINE as an accelerator control system at DESY", Meas. Sci. Technol. 18 (2007), pp. 2379–2386.

[9] The TINE website, http://tine.desy.de/.

[10] P. Duval et al., "Control system interoperability, an extreme case: Merging DOOCS and TINE", PCaPAC'12, Kolkata, India (2012), pp. 115–117.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

[11] J. Wilgen and S. Meykopff, "A unified Matlab API for TINE and DOOCS control systems at DESY", PCaPAC'14, Karlsruhe, Germany (2014), pp. 55–56.

[12] L. Fröhlich et al., "Magnet server and control system database infrastructure for the European XFEL", WEPGF006, *These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

[13] R. Kammering et al., "The virtual European XFEL accelerator", TUD3O04, *These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

[14] S. Meykopff, "Advanced Matlab GUI development with the DataGUI library", WEPGF142, *These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

[15] V. Rybnikov et al., "FLASH DAQ data management and access tools", PCaPAC'10, Saskatoon, Canada (2010), pp. 195–197.

[16] S. Meykopff, "An optics-suite and -server for the European XFEL", PCaPAC'14, Karlsruhe, Germany (2014), pp. 52–54.

[17] E. Sombrowski et al., "jddd: A tool for operators and experts to design control system panels", ICALEPCS'13, San Francisco, USA (2013), pp. 544–546.

[18] E. Sombrowski et al., "A HTML5 web interface for JAVA DOOCS Data Display", WEPGF150, *These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

[19] J. Roßbach et al., "A VUV free electron laser at the TESLA test facility at DESY", Nucl. Instr. and Meth. A 375 (1996), pp. 269–273.

[20] V. Ayvazyan et al, "First operation of a free-electron laser generating GW power radiation at 32 nm wavelength", Eur. Phys. J. D 37 (2006), pp. 297–303.

[21] K. Honkavaara et al., "FLASH: First soft X-ray FEL operating two undulator beamlines simultaneously", FEL'14, Basel, Switzerland (2014), pp. 635–639.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE NEW CONTROL SOFTWARE FOR THE CERN NA62 BEAM VACUUM

S. Blanchard, F. Antoniotti, R. Ferreira, P. Gomes, A. Gutierrez, B. Jenninger, F. Mateo, H. Pereira
CERN, Geneva, Switzerland
L. Kopylov, S. Merker, IHEP, Protvino, Russia

Abstract

NA62 is a fixed target experiment to measure very rare decays of Kaons at CERN Super Proton Synchrotron accelerator. The NA62 experiment line comprises several large detectors installed inside a vacuum vessel with a length of 250 m and an internal diameter of up to 2.8 m.

The vacuum installation consists of 170 remote controlled pumps, valves and gauges. The operational specifications of NA62 require a complex vacuum control system: tight interaction between vacuum controllers and detector controllers, including pumping or venting vetoes, and detector start-stop interlocks; most of the valves are interlocked, including the large vacuum sector gate valves; the vacuum devices are driven by 20 logic processes.

The vacuum control system is based on commercial Programmable Logical Controllers (Siemens PLC: S7-300 series) and a Supervisory Control And Data Acquisition application (Siemens SCADA: WINCC OA). The control software is built upon the standard framework used in CERN accelerators vacuum, with some specific developments. We describe the control architecture, and report on the particular requirements and the solutions implemented.

INTRODUCTION

At CERN, the European Organization for Nuclear Research, the NA62 experiment analyses the outcomes of collisions of proton beams from the SPS (Super Proton Synchrotron) against a fixed target. The rare decays of Kaons are detected along a vacuum vessel of the 250 m. This large vacuum installation is broken down into 7 volumes, called vacuum sectors, separated either by sector valves or by windows. The use of sectors reduces mechanical intervention impact and sector valves, interlocked on pressure rise, avoid the propagation of leaks. The vacuum pressure specification for sectors 1 and 2 is lower than 10^{-2} mbar; it is achieved using 5 primary pumping groups. The vacuum pressure specification for sectors 3 to 7 is lower than 10^{-5} mbar; it is achieved using 8 turbo-molecular pumping groups and 7 large cryogenic pumping groups.

The control software for NA62 Beam Vacuum is based on the same framework as the vacuum control of the LHC (Large Hardon Collider) and its injectors [1].

The huge NA62 vacuum vessels require very large cryogenic pumping stations that offer very high pumping speed. In addition, usual turbo-molecular pumping is installed. The large detectors of NA62 are installed inside the beam vacuum vessels and require a complex hardware and software alarm system between vacuum and detector controls. Performant diagnostic tools are required to improve interventions and reduce costly beam interruptions.

New control device types and new control functionalities have been especially developed in 2014 for the NA62 Experiment. The software architecture is a two-layer system. The lower one, or automation layer, comprises a set of functions and data blocks, running in a Siemens Simatic S7™ PLC. The higher one, or supervisory layer (SCADA), was developed using the Siemens WINCC OA™ application; it consists of a set of routines and processes for monitoring, diagnostics and archiving, together with the graphical user interface.

DATABASE

The database structure, generically called Vac-DB, was developed in collaboration with IHEP for the vacuum control of the LHC and its injectors. This structure is also used for NA62.

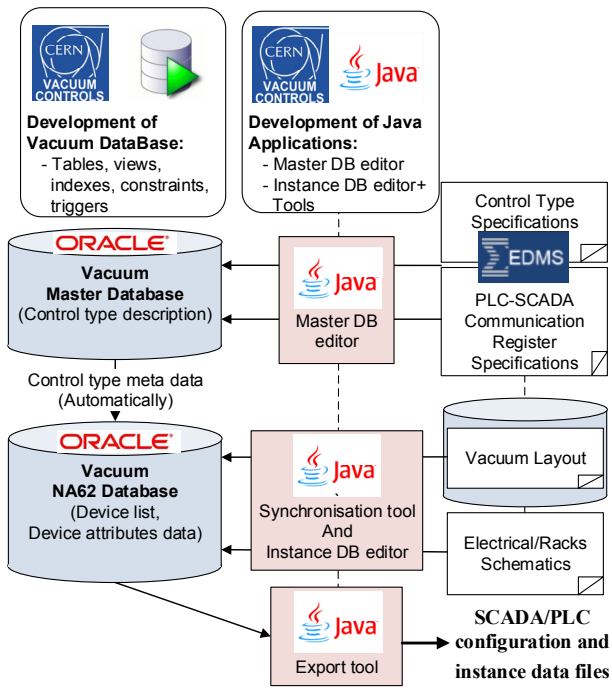


Figure 1: Vacuum database and applications.

Vacuum devices are classified into control types. The control type is identified by the family, type and sub-type of the device; it represents field devices (valves, mechanical pumps, active gauges, passive gauges, etc.) or virtual devices (pumping group process, gas injection process, software interlock, etc.).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A first Oracle database called "Vacuum Master Database" contains the definitions of control types. The definitions include the Data-Point structures for the SCADA application, the description of SCADA-PLC communication registers, and the list of attributes for every control type.

A second Oracle database, the "Machine Instance Database", contains all device instances and all attribute values. The "Vacuum Database Editor" is a Java application used to insert user's inputs and to synchronise attribute values with the CERN central Layout Database. Then the Java application automatically produces SCADA/PLC configuration and instance data files, from database views of the NA62 Vacuum instance database.

PLC SOFTWARE

Simatic Step7 PLC functions have been developed by the vacuum control team. The Baseline of the software is composed of organisation blocks (OBs), diagnostic functions and control type functions. Both the baseline and the PLC hardware configuration of NA62 are archived in a versioning repository (SVN) and then deployed to PLC projects; the instance data is generated by the Java export tool from Vac-DB.

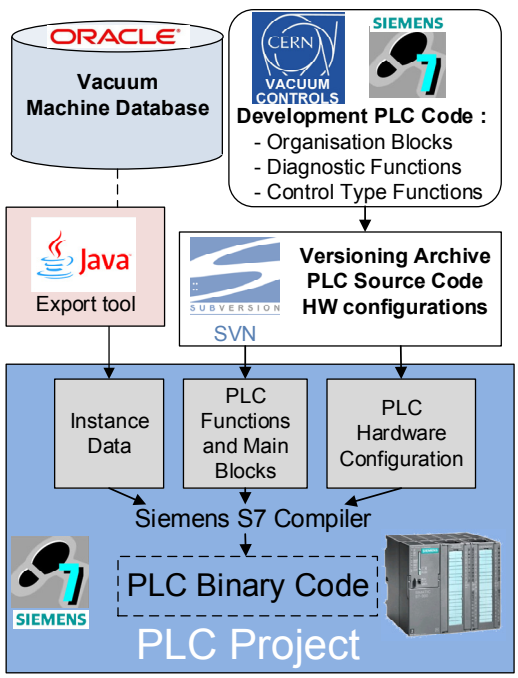


Figure 2: PLC software production.

The PLC binary code is generated by compiling the hardware configuration, the baseline and the instance data. The process of development, archiving and production of PLC code is illustrated by Fig. 2.

Two different software standards run in each PLC CPU. Before 2011, control types were developed using assembly language for PLC functions [2]. Since 2011, all new control types are developed in structured text language, within a novel standard.

Legacy control types that required new functionalities (primary and turbo-molecular pumps, valves and active gauges) were migrated to the new standard. Nevertheless, a significant number of legacy control types remained unchanged.

Legacy Assembly PLC Functions

A set of PLC functions and data blocks was created in 2001 for the vacuum controls of the SPS, then deployed in the LHC and later in PS-Complex and experimental areas. The functions were developed in Statement List Siemens (STL), similar to assembly language. The main organization block (OB) polls throughout a device list and handles one device per PLC cycle. The control type for a device is taken from device Data Block (DB) and the associated device type function (FC) is called with the device DB number as a parameter. Each control type has a dedicated FC; a complete set of FCs covers devices such as ion pumps, sector valves, passive gauges, etc.

Each device has its own device DB, containing all the variables needed by the function; the DB structure depends on device control type. Functions access and update DB variables, global variables and Write and Read Registers.

New Structured Text PLC Function Blocks

A set of new PLC Function Blocks (FB) were develop to incorporate new functionalities for existing control types, and to integrate new control. It has become the standard in vacuum controls PLC software, for all new developments since 2011. Not all of the previously created control types have been migrated to this standard yet; hence, both VACCIN and PSL system coexist in same PLC CPU.

The new standard has the following specifications:

- SIEMENS-SCL Language: textual high-level language; used for complex algorithms.
- FBs: functions with instance DBs.
- Process driven by a phase sequencer: sequence of actions with conditions and time dependencies.

There is one PLC FB per control type, who manages the behaviour of the device taking into account its sub-type. The typical structure of an FB is: get orders and parameters (from communication registers), read physical inputs, manage software interlocks, calculate and update maintenance variables, manage errors and warnings, manage behaviour according to the logic (which can be sub-type dependant), write outputs, update statuses (communication registers). Each device has a DB instantiated by the respective control type FB. This instance DB is composed of: header variables (name, control type, sub-type, machine…), common variables (auto request, manual request, interlock request, on status, off status, error status, object status, mode status…), control type specific variables (input/output addresses, valid range, unit conversion variables, maintenance variables…).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PLC-SCADA Communications

Both PLC programming standards have the same PLC-SCADA communications schema. Devices statuses, orders and parameters are stored in a set of communication DBs. The SCADA application writes orders and parameters into the Write Register DB, from which the PLC function gets the data during initialisation phase. At the end of the PLC function, statuses are updated in the Read Register DB. Vacuum systems are slow processes, allowing 1s as polling period of the Registers by the SCADA. Orders and parameters are transmitted to the PLC Write Register immediately after operator request. The PLC time is synchronized with Network Time Protocol servers, and some device data are time stamped with the accuracy of the PLC cycle time.

SCADA SOFTWARE

The supervisory layer of the NA62 vacuum controls is based on Siemens WINCC OA™ application. The SCADA framework [3] developed in collaboration with IHEP, for the vacuum control of the LHC and its injectors, has been reused for NA62. The schema of development, archiving and production of SCADA code is illustrated by Fig. 3.

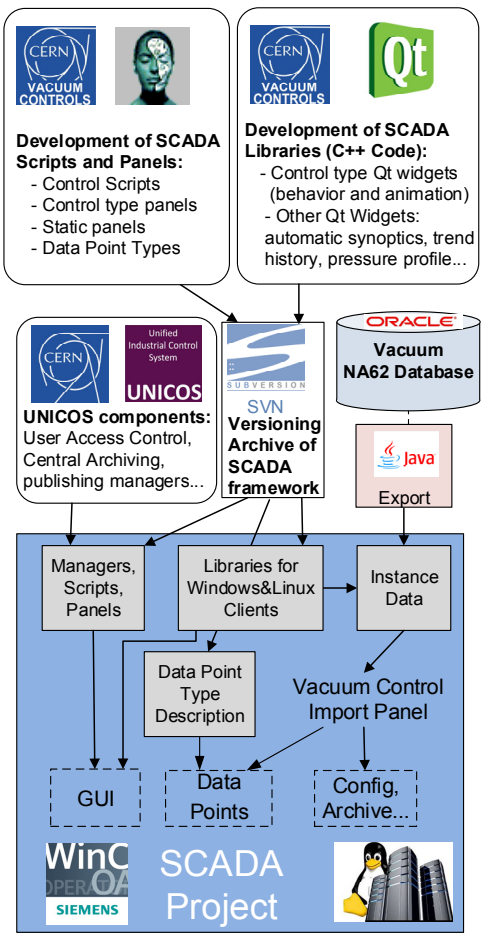


Figure 3: SCADA software production.

Graphical User Interface (GUI)

Devices are represented in the GUI using external widget objects. Symbols, animations and behaviours are defined by QT/C++ code and are control type dependant. Automatic synoptic, trend history and pressure profile are generated by QT/C++ libraries.

In the synoptic panel, main orders are directly accessible with a right click on the device symbol. A double click on this symbol opens the details panel of the corresponding control type. This details panel is graphically developed with the WinCC OA Graphical Editor.

Other functionalities, common to different panels, are developed in QT/C++ libraries.

NA62 NEW CONTROL TYPES AND FUNCTIONALITIES

Several control types and functionalities have been especially developed for NA62 and are currently only deployed in this experiment.

Pumping Group Logic

The high vacuum in sectors 3 to 7 is produced with turbo-molecular pumping groups. The control hardware for pumping groups has been improved [4] with an automatic venting in case of power cut, thus avoiding oil contamination from the primary pump to the turbo-molecular rotor. In addition, the logic disables the venting in case the valves to the vessels are not closed, to prevent accidental venting of the sector vessels.

The vacuum control system is linked to the NA62 detectors control system through several alarms.

Alarms to DCS

Three kinds of alarms are provided to the Detector Control System (DCS).

First, pressure level alarms: free potential relay contacts from the vacuum gauge controllers are hardwired to the DCS. The contact is triggered by the controller following a hysteresis comparison of the pressure measurement against pre-programmed thresholds. The pressure measurement can be low-pass filtered at 16, 160 or1 600 ms; the thresholds can be remotely setup from the SCADA, through the PLC and Profibus network.

Second, vacuum process alarms: PLC output relay modules are hardwired to the DCS. They deliver vetoes to the detectors, according the status of the vacuum sectors. These alarms are calculated by complex algorithms according to the vacuum process states. They are parameterized in the vacuum database, from which is defined the PLC instance data block, the type of logic and the reference to the source devices.

Finally, high level notification alarms: the SCADA application publishes the statuses of gauges and valves to

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CMW[1] for the NA62 experiment operators. These alarms are not critical and used for information only.

Alarms from DCS

In the opposite direction the DCS provides 2 types of hardwired alarms to the vacuum PLC.

First, pump enable alarms: if the detector is not ready for pumping, these alarms prevent any associated pumping valves to be opened. During operation, in the event of a detector window rupture, H2 would be released in the vacuum vessel; the pumping and so the compression of a large quantity of H2 being highly explosive. In case of window rupture detected, the process will automatically close pumping valves.

Second, vent veto alarms: the venting of some sectors may be required during a mechanical intervention. The venting is performed in-situ using a manual valve. Local notifications, using flashing LED, remind to the vacuum operator if venting is allowed or not.

PLC and SCADA developments for these new alarm types have been added to the vacuum control framework. These alarms allow a strong interaction between Detector and Vacuum Control Systems; they provide effective diagnostic information to both vacuum and detector operators.

Pulsed-command Valves and Pumps

In most of the accelerator vacuum systems, the fail-safe position of sector valves is closed, to avoid propagation of any unexpected leak. In the NA62 experiment, due to beam physics transparency requirement, very large valves have been installed with a specific mechanism. As any valve movement produces a leak, a new control type has been developed to cope with this behaviour. The same kind of control type for pumps has also been developed, to drive primary pumps with similar specifications. For both valves and pumps the last order requested is thus maintained in case of power failure.

Gauges with Profibus Interface

Gauges with embedded controller and Profibus interface have been installed in NA62 for the first time at CERN's accelerators vacuum. These gauges may reduce significantly the cost of controls. A new control type has been developed to integrate them in the vacuum controls framework. The new control type offers some extra functionalities as compared with standard one, like the possibility to act individually on each sensor of the full-range pair.

HSR™ Cryogenic Pumping Group

Seven very large cryogenic pumps have been installed in the Decay tubes vacuum sector (135 m long and 2.8 m diameter vessels with decay tubes detectors inside). These are the largest cryogenic pumps installed at CERN, if not counting the pumping effect of the superconducting magnets in the LHC. They are provided by HSR™ together with their dedicated controller.

For turbo-molecular pumping groups, the PLC manages the logic, which can be found in several control types (one per field device type, and another for the process).

For the HSR™ cryogenic pumping groups, it is slightly different: the group is controlled as a whole by the vacuum PLC, while the process is managed by the HSR™ controller. A first control type is dedicated to the cryogenics cold head, its sensors and the back valves. A second control type is dedicated to the large valve with same behaviour as standard vacuum valves. Both send global orders and get field device statuses from the HSR™ controller via a RS-232 interface. Some vacuum framework functionalities required upgrading; e.g. the trend history originally developed for one analogue value per device has been extended to integrate all the analogue values and bit statuses of the cryogenic pump control type: 2 pressure analogue values, 2 temperature analogue values, 4 bit statuses for the fore valve (V1) and purge valve (V5).

CONCLUSION

The vacuum control framework has shown a high level of flexibility, while being used to build the core of the NA62 vacuum control system. New device types and particular user specifications required custom developments. Thanks to the well-organized structure of the vacuum control framework, the new control types and functionalities have been easily developed and deployed. The result is a very effective control system well adapted to NA62 experiment layout and operational specifications.

REFERENCES

[1] P. Gomes et al, "The Control System of CERN Accelerators Vacuum [Current Projects & LS1 Activities]", ICALPECS13, San Francisco, 2013, MOPPC027.

[2] R. Gavaggio et al., "Development of The Vacuum Control System for The LHC", ICALPECS05, Geneva, 2015; PO1.035-6.

[3] F. Antoniotti et al., "Developments on the SCADA of CERN Accelerators Vacuum", ICALPECS13, San Francisco, 2013, MOPPC030.

[4] S. Blanchard et al., "Vacuum Pumping Group Controls based on PLC", PCaPAC14, Karlsruhe, 2014; WPO030.

[1] Control MiddleWare is a software communication infrastructure for the CERN accelerators and experiments

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

THE UPGRADE OF CONTROL HARDWARE OF THE CERN NA62 BEAM VACUUM

F. Mateo, F. Antoniotti, S. Blanchard, R. Ferreira,
P. Gomes, A. Gutierrez, B. Jenninger, H. Pereira
CERN, Geneva, Switzerland

Abstract

NA62 is the follow-up of the NA48 experiment, in the SPS North Area of CERN, and reuses a large fraction of its detectors and beam line equipment. Still, there are many new vacuum devices in the beam line (including pumps, valves & gauges), which required a thorough modification of the control system and a large number of new controllers, many of which were custom-made.

The NA62 vacuum control system is based on the use of PLCs (Programmable Logic Controllers) and SCADA (Supervisory Control and Data Acquisition). The controllers and signal conditioning electronics are accessed from the PLC via a field bus (Profibus); optical fibre is used between surface racks and the underground gallery.

The control hardware was completely commissioned during 2014. The nominal pressure levels were attained in all sectors of the experiment. The remote control of all devices and the interlocks were successfully tested.

This paper summarizes the architecture of the vacuum control system of NA62, the types of instruments to control, the communication networks, the hardware alarms and the supervisory interface.

INTRODUCTION

NA62 will study the rare decay $K^+ \to \pi^+ \nu\bar{\nu}$ at the CERN SPS. NA62 is built to improve the current experimental precision by measuring kaon decays-in-flight with calorimetry to veto extra particles, very light mass trackers to reconstruct the momenta of the K^+ and the π^+ and full particle identification capability [1].

To reach the required sensitivity and acceptance, the vacuum in the decay area needed to be improved by at least two orders of magnitude compared to NA48. Many detectors that would normally be outside the vacuum chamber are now part of the vacuum system leading to a significant increase of the gas load to be pumped. A particularity of NA62 is the distribution of detectors along 250 m (Figure 1).

The NA62 experiment went through a complete refurbishment with the introduction of new detectors filed with gas. Their operation requires a vacuum system, based on turbo molecular and cryogenic pumps. The beam line is divided into 7 sectors, each having specific vacuum requirements. The vacuum equipment in each sector will be described in the next section.

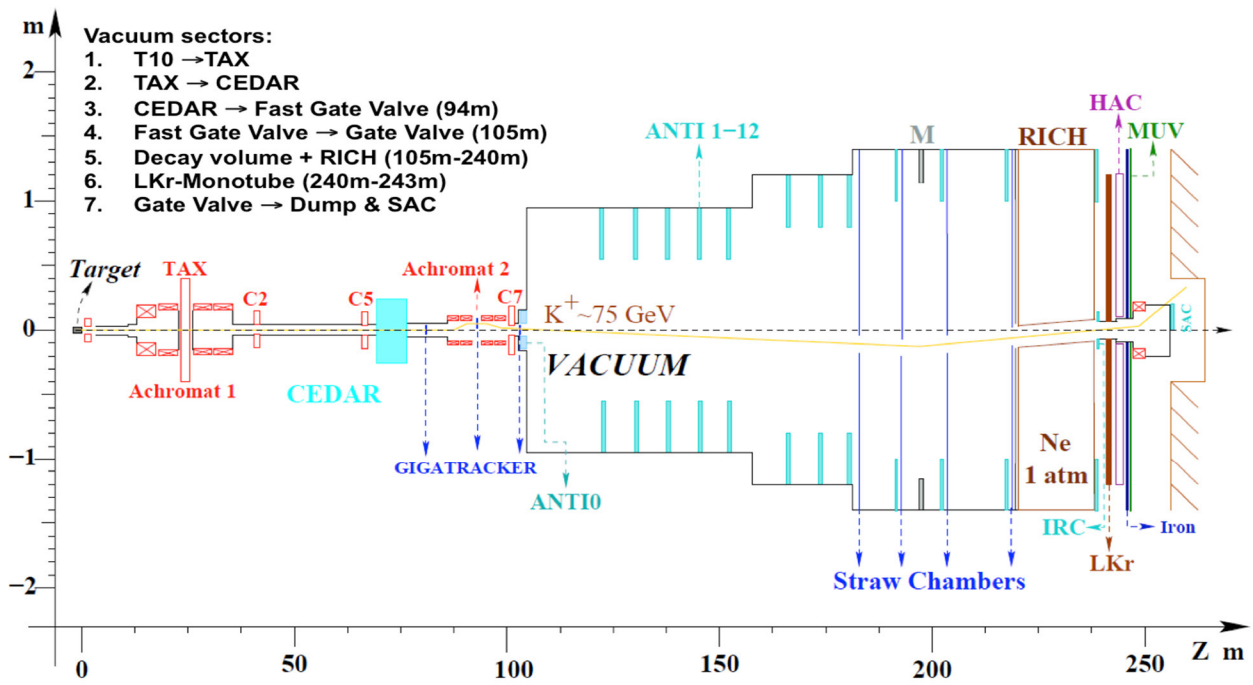


Figure 1: Layout of the NA62 experiment

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

VACUUM EQUIPMENT

The upgrade of NA62 beam vacuum comprises more than 200 new instruments (including different types of pumps, valves and gauges) along the beam line. In addition, a large amount of cabling had to be designed, installed and commissioned.

This was a challenging work, as the NA62 beam line measures roughly 270 m from the SPS T10 target to the NA62 beam dump, with also some equipment in a surface building.

In sectors 1 and 2 only passive equipment is used due to the high level of radiation. In sectors 3 and 4, radiation-tolerant active vacuum gauges may be used but not instruments with embedded Profibus interface. Sectors 5 to 7 are radiation-free and allow the installation of controllers and PLCs close to the instruments. The distribution of vacuum devices is shown in Figure 2.

Vacuum Pumps

Vacuum pumps are the basic elements needed to achieve high vacuum in accelerators and experiments. Most of the time, several types are combined and assembled together in pumping groups to pump down from atmospheric pressure down to high vacuum. Three types of pumps are used in NA62 (Table 1):

- **Primary or roughing pumps** (VPR) are used to evacuate from 1 bar down to 10-3 mbar; they are also used as a backing pump for Turbo Molecular pumps.
- **Turbo molecular pumps** (VPT), are effective in the range 10^{-3} to 10^{-10} mbar, therefore requiring rough vacuum achieved by a primary pump. A VPR and a VPT are often assembled and controlled together as a turbo molecular pumping group (VPG).
- **Cryogenic pumps** (VPC) are used to achieve very high pumping speeds at low pressures. They only need rough pumping during regeneration, but not during operation. In NA62, a pumping speed of more than 100 000 l/s is required to reach the low 10^{-6} mbar pressure range, with the particularly high gas load (Ar, CO_2) from permeation from the straw detectors. This is accomplished by 2-stage Gifford-McMahon cryo-coolers, using gaseous helium closed circuits and cryosorbers on the second stage. A cryogenic pumping group (VPGC) includes heaters, gauges and valves.

Table 1: Summary of vacuum pumps in NA62

Type	Units
VPR	7 (5 rotary vane primary @220V, 2 combined dry screw/roots @400V)
VPT	8 (7 DN 100 for normal operation, 1 DN 250 mainly for leak detection)
VPC	7

Vacuum Valves

Apart from the standard electro-pneumatic or electro-magnetic (angle or gate) valves commonly used in accelerators, two special types can be found in NA62.

A fast shutter valve (VVQ) closes within 20 ms, in case of rupture of the thin window that separates the CEDAR detector from vacuum, thus protecting fragile equipment downstream.

The aluminium-sheet valve upstream of the liquid Kr calorimeter must not move unnecessarily, to avoid air-inrush. Its bi-stable actuator does not lose position under a loss of power or compressed air. Also, together with the associated gauges, the controller is powered by an UPS.

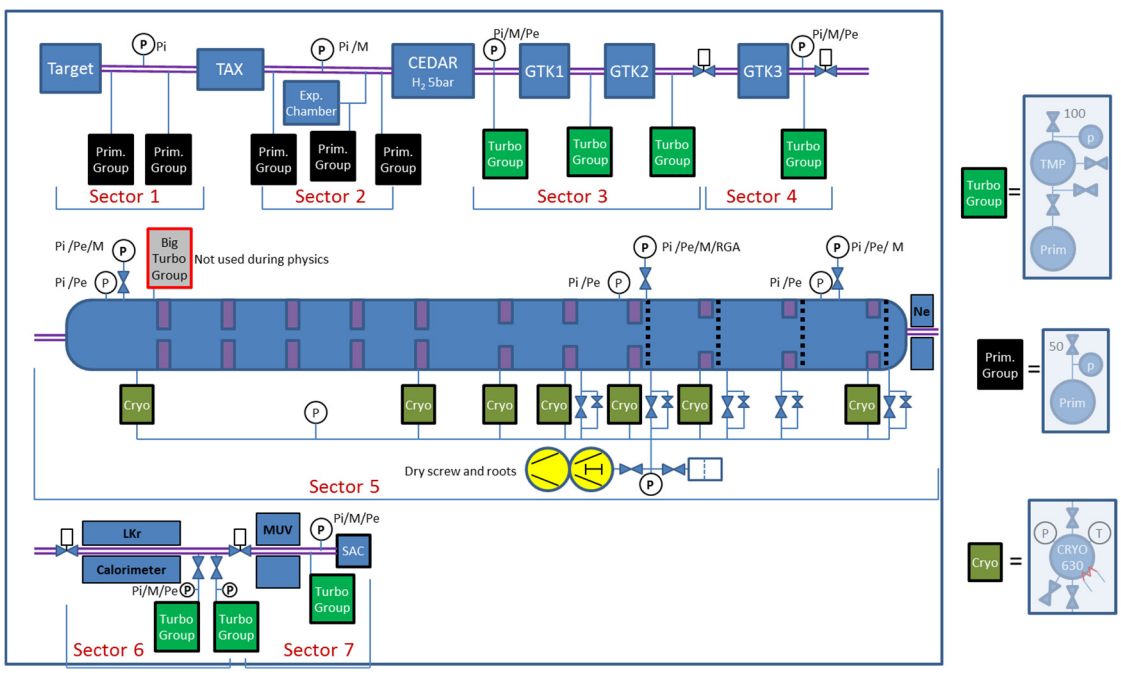


Figure 2: Vacuum equipment layout in NA62 experiment

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Vacuum Gauges

Several types of vacuum gauges are used in NA62 depending on the range of vacuum to be measured. Both passive and active electronics gauges are used depending on the radiation levels. In the NA62 experiment, the following types of gauges can be found:

- Passive Pirani gauges (VGR)
- Passive Penning gauges (VGP)
- Active Membrane gauges (VGM)
- Active Full Range gauge-pairs (VGF)
- Active Profibus interfaced gauges(VGF, VGM)

CONTROL SYSTEM

NA62 follows the same controls architecture of the LHC [2]. This means that the vacuum controllers are managed using a PLC-based architecture, using the Siemens™ S7-300 series. The human-machine interface is a SCADA (Supervisory Control And Data Acquisition), built with Siemens WinCC OA®, formerly PVSS®.

The PLC architecture of NA62 is based on a master-slave topology. There is one master PLC, 8 slave PLC for the pumping group control and 5 remote I/O stations that have specific functionality and connectivity, depending on the instruments to interface.

Specific control crates are used for each type of instrument. Some of them have been designed, manufactured and tested at CERN while others are commercial. The controllers have been distributed in six racks in a radiation-safe area of the underground gallery, and two racks in the surface building, where the Master PLC is located. Tables 2 and 3 summarize the controller types used in NA62. The VAT VP-3 valve controllers needed an additional UPS to ensure protection of the LKr in case of power cut.

Table 2: Custom-made controller crates for NA62

Name	Description	Units
VRLMS	Master PLC	1
VRLRG	PLC Remote I/O station	5
VRPGMF	VPT controller	8
VRJTB	VPT patch panel	5
VRPPH	VPR controller	2

Table 3: Commercial controllers at NA62

Name	Description	Units
VRPP	Primary pump local crate	5
VRVRC	VAT VF-2 fast valve controller	1
VRGPT300	Pfeiffer-Balzers TPG 300 gauge controller	5
VAT_VP	VAT VP-3 dual-acting valve controller	2
VRPYA	HSR PCA700C VPC controller	7

The installation and commissioning of all control crates took place during spring and summer of 2014.

COMMUNICATION

For the connexion of all instruments to the respective controllers, a cabling campaign was carried out during spring 2014.

As some equipment (including the Master PLC, one remote I/O station and other controllers) are in a surface building, about 100 m away from the beam line, an optical link was devised to provide fast and reliable communication.

The communication between controllers and PLC is made via Profibus-DP, where there is no radiation hardness (Sectors 5 to 7). Otherwise the communication relies on dedicated individual cables. Three separate Profibus-DP networks have been devised for each specific purpose:

- **Profibus Alarms network (I)**: Links the equipment that are source of some type of Alarm. The alarms will actually be generated by the Master PLC that will send them to the Detector Control System (DCS) [3].
- **Profibus VPT network (T)**: Establishes the communication between the VPT controllers and the Master PLC.
- **Profibus All-purpose network (A)**: Provides the communication interface for Profibus instruments that do not belong to the aforementioned groups (i. e. Profibus gauges).

The interface between the optical and copper Profibus networks is made using Siemens™ Optical Link Modules (OLM).

The management of the VPC controllers was carried out by a remote I/O station using a specific communication protocol based on a Profibus to RS-232 gateway, to interface with the master cryo-controller, which periodically polls the rest of cryo-controllers using a token ring topology [4].

As in other experiments and accelerators at CERN, the Master PLC is connected to the SCADA server via a dedicated Ethernet Technical Network.

The communication networks between vacuum instruments and controllers are summarized in Figure 3.

SUPERVISORY APPLICATION

The supervisory application of NA62 acts as a user interface that can be accessed from the NA62 control room to monitor the status, history and, depending on the user privileges, act on the control of the vacuum devices. It was developed in WinCC® OA and its main screen is illustrated in Figure 4. All the vacuum devices have been modelled and all the relevant attributes are retrieved from a SCADA database. The pumping groups have also been modelled in detail, showing the status of all the devices in the group, as well as the hardware alarms status and history.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

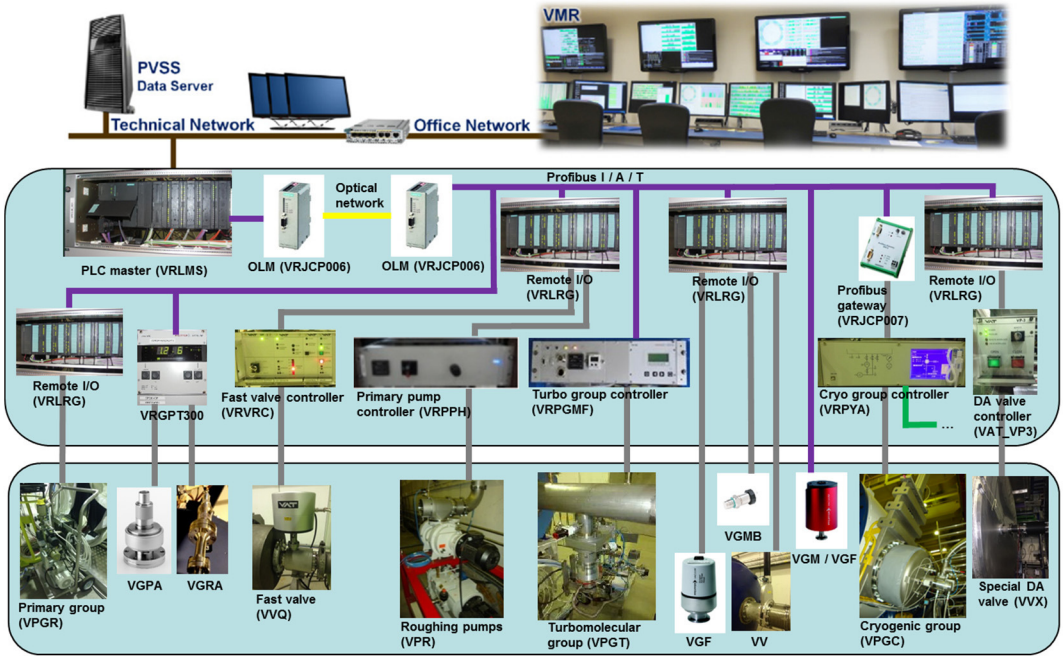


Figure 3: Schematic view of the vacuum controls architecture and communication of NA62.

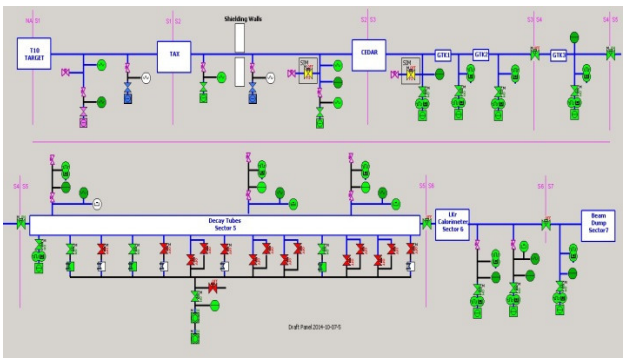


Figure 4: Supervisory application main screen.

HARDWARE ALARMS

For a safe operation of the experiment, the relevant alarms had to be implemented to prevent damage of equipment in case of hypothetical operation failures. The hardware alarms of NA62 are described in Table 4, indicating if it is an output or an input to the vacuum system. The alarms are hard-wired with a dedicated cable from the master PLC crate to the Detector Control System. The alarms were successfully tested during autumn 2014.

CONCLUSION

The NA62 control hardware upgrade was a complex task that required the active participation and effort of several groups at CERN, including engineers, technicians, external workshops and industrial partners.

The cabling and control equipment deployment took place in spring 2014. The full system was commissioned during summer 2014. The measured pressure levels meet the nominal specifications [5] in every sector. The alarms are fully operational.

Table 4: Hardware alarms in NA62 vacuum

Output	Enable Inside CEDAR pumping (Sectors 2 and 3 are under vacuum)
Output	Enable GTK cooling (Sectors 3 and 4 are under vacuum)
Output	Enable High Voltage on LAV (Sector 5 atmosphere or under high vacuum)
Output	Enable Inside RICH pumping (Sector 5 is under vacuum)
Output	Enable High Voltage on SAC (Sector 7 atmosphere or under high vacuum)
Input	Pumping Enable for Sector 2 and 3 (CEDAR Window No Rupture)

REFERENCES

[1] F. Hahn *et al.*, "NA62 Technical Design Document, NA62-10-07", (2010); http://cds.cern.ch/record/1404985

[2] P. Gomes *et al.*, "The Control System of CERN Accelerators Vacuum (LS1 Activities and New Developments)", Proceedings of ICALEPCS2013, San Francisco, CA, USA, (2013).

[3] G. Maire *et al.*, "The Detector Safety System of NA62 experiment", Proceedings of ICALEPCS2013, San Francisco, CA, USA, (2013).

[4] R. Ferreira, "PLC devices for the NA62 Cryogenic Pump Controllers VV_H_HCC | VPC_HCCC | VPC_HCCG", CERN Technical Note, (2014); https://edms.cern.ch/document/1491552/1

[5] S. Blanchard *et al.*, "Vacuum control system for the NA62 – Work Package Description", CERN Technical Note, (2013); https://edms.cern.ch/document/1280667/1

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

CONSOLIDATIONS ON THE VACUUM CONTROLS OF THE CERN ACCELERATORS, DURING THE FIRST LONG SHUTDOWN OF THE LHC

P. Gomes, F. Antoniotti, F. Aragon, F. Bellorini, S. Blanchard, J-P. Boivin, N. Chatzigeorgiou, F. Daligault, R. Ferreira, J. Fraga, J. Gama, A. Gutierrez, P. Krakowski, H. Pereira, G. Pigny, P. Prieto, B. Rio, A. Rocha, H. Vestergard
CERN, Geneva, Switzerland
L. Kopylov, S. Merker, M. Mikheev ; IHEP, Protvino, Russia

Abstract

For two years (Spring 2013 – Spring 2015), the LHC went through its first long shutdown (LS1) [1]. It was mainly motivated by the consolidation of magnet interconnects, to allow operation with 6.5 TeV proton beams. Moreover, around the accelerator complex, many other systems were repaired, consolidated or upgraded, and several new installations came to life.

The standardization of vacuum controls has progressed in the injectors, with the renovation of most of their obsolete equipment.

In the LHC, many new instruments were added, the signal transmission integrity was improved, and the exposure to radiation was reduced in critical places. Several developments were needed for new equipment types or new operational requirements.

INJECTORS & LHC CONSOLIDATIONS

Complex PS

The LS1 was an opportunity to extend the standardization of vacuum controls to the Proton Synchrotron ring (PS), the Antiproton Decelerator (AD), and the nTOF experiment. The legacy controls were updated to the same PLC/SCADA-based architecture as in the other machines. Only the pumping groups of the PS complex remain to be upgraded, during the following years. [2]

The existing controllers were either upgraded, or replaced by newer generations, connecting to the PLC directly through Profibus or via remote-IO stations. This involved 500 cables and 700 instruments.

Above all, a large effort was put in the production of complete and up-to-date documentation on the control system layout and on its components technical details.

Linac2, at the head of the proton acceleration chain, received a new set of six pumping groups. These were especially developed to compensate for the dying old ion pumps, helping it to survive until replaced by Linac4.

SPS

In the SPS, controls renovation was applied to its injection line, the interfaces for the Beam Interlock System, and COLDEX.

Several new gauges and valves were added, for a few new sectors. Furthermore, all racks and cables were prepared for the new re-sectorization of all the ARCs.

Campaigns for replacement of aged cables were performed in several places. In total, some 300 cables were replaced, and 200 other were newly installed, mainly for some 80 new instruments.

LHC

About 600 cables and 400 gauges, valves, and pumps, were newly installed in the LHC, together with the corresponding interlocks. Some of them required specific controls developments, treated in detail further down.

The signal integrity has been revised and improved, for the gauges in the ARCs and for the bypass-valves. To mitigate radiation effects in Point7, 27 racks full of equipment were dismounted and relocated into a less exposed zone, while 400 cables had to be extended. These subjects are treated in another paper [3].

During the second half of 2014, six teams of two people were full-time involved in the commissioning of all controls and instrumentation of the LHC. All instrument chains were fully tested; and calibrated when necessary. Documentation was updated about racks layout and interlocks cabling.

During the LHC global check-out, systematic tests were performed on all interlocks, to every sector valve, and to the beam interlock system, automatically from SCADA.

All in all

In total, more than 2 900 cables and 1 500 instruments were commissioned in the accelerators consolidations and new projects. The controls team had to be reinforced by 50%, mainly with engineers from collaborating Institutes, trainees, and technicians from industrial support. The main challenges were the coordination of field activities with access restrictions and other teams' tasks, and having each machine commissioned and ready in time for beam.

DEVELOPMENTS FOR PUMPS

NEG Pumps

A Non-Evaporable Getter (NEG) is a highly reactive and porous material; it needs to be activated, by heating under vacuum, with a suitable temperature and duration. The recommended activation temperature is around 475 °C, corresponding to 6A under 16V, for 1 h.

To increase pumping speed in critical equipment in the LSSs of the LHC, NEG cartridges (Capacitorr D400 2 from

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SAES™) were installed, with their activation cycle remotely controlled.

The DELTA™ SM6000 power supply (300V x 20A) is interfaced to the PLC via Profibus. It would feed up to 18 cartridges in series, if it were not for the voltage drop on the long cables. In practice, less than 10 are installed.

Several series of cartridges can be activated from the same supply, but at different times; a multiplexer crate steers the power to one of the series at time, via relays operated from PLC digital outputs.

Through Profibus, the PLC sets the required current on the power supply, which automatically adjusts the necessary voltage. The PLC will only set the power ON if the pressure is below 1e-3 mbar. Several operation modes, parameters, errors and warnings are implemented.

Sublimation Pumps

A Titanium sublimation pump consists of two Ti filaments in a cylindrical stainless steel vacuum chamber. A power supply, custom-designed in 1994, heats one of the filaments to the temperature of sublimation of Ti, in order to coat the inner surface of the chamber, which can then start to absorb the residual gases.

As the pumping efficiency reduces with time, the sublimation process must be repeated periodically. The power supply has two modes of operation: degassing, where both filaments are powered with a given current, for cleaning; and the actual sublimation of one filament, for a given time. It is hardware interlocked on Penning gauge, preventing any powering above 1e-7 mbar.

Remote access through PLC has now been implemented for PS, AD, and will later be extended to the LHC. The PLC sends binary commands and analog set-points, and receives binary status and analog feedback, through ET200™ remote-IO stations; each station can handle 8 devices. From the SCADA, the operator can define the set-points and the duration of the sublimation periods; recipes are available for repetition cycles, for one or several pumps.

Gas Injection in LHC

A prototype of the Beam Gas Vertex (BGV) detector has been installed in the LHC-LSS4L during LS1. This is a non-invasive beam size measurement instrument. It requires a gas injection system for neon or argon.

The control system for this gas injection is similar to the one used for the two Beam Gas Ionization (BGI) detectors, previously installed on both sides of Point4. Neon gas is injected into the BGI vacuum chamber and then ionized to measure beam emittance on LHC.

For both BGI and BGV applications, the gas injection system is composed of a standard pumping group [4], an analogue valve connected to the gas bottle, and an on-off injection valve connected to the beam pipe. The injection process is handled by a sequencer and operation modes. This PLC-based control achieves a safe and automatic gas injection without the need of a specialist in vacuum and injection techniques.

Gas Injection in Linac4

In 2014, LINAC4 had a new system to inject H2 gas in the Low Energy Beam Transport vessel, for the measurement of the beam current and size. It uses a commercial fine-dosing thermo-valve and its control unit, a primary pump and two on-off valves. They are all connected to a S7-300 PLC, which runs a sequencer and operation modes. The process starts with an isolated mode, to avoid accidental large gas flow that would damage vacuum and beam instrumentation. Then the process can run either in tuning mode (pressure set-point defined in vacuum SCADA) or nominal mode (pressure set point defined by the beam transfer controls).

Cryogenic Pumps

In NA62, seven large and fast Cryogenic pumps are used to reach 1e-6 mbar, by pumping at 100 000 l/s. The process is managed by a proprietary controller (PCA700C from HSR™). Unfortunately, no Profibus interface is available; one controller connects to the PLC via RS-232 and a Profibus gateway; the other controllers are chained with that one, on a token-ring network.

At first, this communication did not work due to a bug in the firmware; it was eventually fixed by the manufacturer, although with some delay for the project.

DEVELOPMENTS FOR GAUGES

Full-Range Gauges

In radiation-free zones of NA62 and HIE-ISOLDE, Pfeiffer™ MPT200 full-range gauges have the signal conditioning and Profibus interface integrated; this reduces cabling costs and offers increased functionality, such as status information and commands directly available on the PLC, without passing through dedicated IOs. The MPT200 comprises a Pirani and a Penning gauge to cover 1e3 to 5e-9 mbar, and switches automatically between the gauges.

After some issues with the evolution of the connector types for Profibus and for power, the gauges performed well. There are two operational modes which, while changeable during the configuration of the device on the Profibus network, cannot be modified during operation.

Thermocouples

800 E-type thermocouple channels have been installed around the machine, for warm magnets and collimators. The measurement covers -40 to +750 °C, with absolute accuracy of +/-2 °C, and is shown with 0.1 °C precision.

Specific 16-input analog modules in Siemens ET200 remote IO-stations, convert the thermocouple voltage into temperature, before sending it to the PLC, who makes them available on the SCADA.

A new instrument-type was defined in Vacuum-DB and developed in the PLC, together with a SCADA panel grouping the 16 channels of each analog input module. There are no commands or interlocks associated to them.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Passive Gauge Controller (TPG)

The TPG300™ gauge controllers are remotely managed by a PLC function created in 2003, never updated since 2007. With the rising number of devices in the system, up to 40 per PLC, some limitations emerged; these are mainly due to the slow polling rate of SCADA to PLC, together with the increasingly slow scheduling of TPG access from PLC.

To visualize on the SCADA all parameters and status of a single TPG, it may take up to 4 minutes, as each value is individually requested to the PLC, and then to the TPG; idem for parameter changes.

In order to keep track of parameter changes by the operators, a daily script gathers the information of all TPGs in each accelerator. This is now taking more than 10 h, and may fail in at some occasions.

A new PLC function was developed and is under test in one of the LHC points. Each PLC cyclically requests all the parameters and statuses of all TPGs connected to its Profibus network; all data is read and logged by the SCADA, and is fully refreshed in the background every 1 to 5 minutes. The historical follow-up of TPG parameters and statuses will thus be directly and immediately available in the SCADA's archives, with no need for heavy daily extraction by an external script.

Cryo-Maintain Interlock to Sector Valves

A temperature increase in one superconducting magnet of the LHC may lead to a pressure rise, propagating into neighboring vacuum sectors. The sector valves around magnets in critical zones have been software-interlocked on the cryogenic conditions. In this way, the loss of "Cryo-Maintain" status will isolate the corresponding sector, avoiding the propagation of the pressure rise.

The cryogenic status for the concerned zones is read directly by the vacuum PLCs in dedicated data-blocks of the cryo-PLCs. An incremental watch-dog is used to assure data validity. Of course, this interlock is not treated by the valves controller if the beam is present.

PROJECTS

COLDEX

In the SPS, the Cold Bore Experiment studies the beam pipe coating with amorphous carbon, under cryogenic conditions. Installed in a by-pass line, on top of two sliding platforms, it is moved into the beam line during experimental periods, and out of the beam for normal SPS operation [5]. The two old step-motors were replaced by new ones, and the old controls gave way to a Siemens S7-300 PLC, integrated in the SPS vacuum control system.

Operation can be manual or automatic, with check and compensation of differences in the advancement of each motor. Each platform provides feedback on the analog position and on the two end-switches.

To move into the beam line, the vacuum sector valves around the experiment must be closed, and the neighboring bumper magnets must be off. For improved security, the operation cannot be done remotely, and requires a physical key on the controller to enable it and to assure the magnets are not powered. A portable touch- panel provides interface to the system from the tunnel, near the experiment. The renovation of the motor control system was extended to the experiment´s vacuum instruments. All can be remotely monitored on the SPS SCADA, and statuses are available for other systems.

Given the importance of amorphous carbon coating for the project HL-LHC, the COLDEX experiment and its controls will continue to evolve in the near future.

SUBU

This Chemical Polishing machine for the surface treatment of superconducting RF cavities was developed in 1989. It was based on Siemens S5 PLCs and a custom HMI, running on Win-3.11. With time, the S5 product line had been discontinued, the HMI hardware was failing and not repairable, the HMI software was running on an obsolete OS, and the PLC code was un-documented. [6]

The challenge of this project was the nearly non-existing documentation, which led to a complete reverse-engineering of the machine: consulting with the operator to understand the process and the plant hardware; visual inspection of the sensors, actuators, and wiring; and re-writing the process description and all electrical diagrams.

The status of the field devices (valves, pumps, sensors, containers) was evaluated and, as it was judged satisfactory after minor reparations, left in operation.

The old control system was completely decommissioned and the new one was developed from scratch, targeting Siemens S7-300 PLCs and WinCC-OA SCADA, under the UNICOS framework, like done before for the ISOLDE complex. As chemical Polishing is a continuous process, the functionalities provided by the UNICOS-CPC are perfectly suited.

Furthermore, because the development in UNICOS is a well-standardized practice, maintenance work and eventual changes can be performed more easily by third parties; also, the development methodology forces the developer document the work thoroughly, resulting in quality documentation that always proves useful.

NA62, AD, LINAC4, HIE-ISOLDE

In NA62, the beam vacuum and the associated controls were completely renovated, and then commissioned during 2014. The new controls architecture follows the framework already used in the LHC and Injectors. It also incorporates a few novelties, which required specific developments in the PLC and SCADA. The hardware is described in [7] and the software in [8].

In AD, many consolidations were completed during LS1: gauges, valves, and ion pumps. The experiments racks were already moved to free space for ELENA installation.

The controls of Linac4 and HIE-Isolde have both been evolving, as new sectors and their instrumentation are added and commissioned. In parallel, the respective test-stands require maintenance and development.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

FROM LS1 TO LS2

The next long shutdown (LS2) foreseen for 2019-20 will be dedicated to: Injectors upgrade; LHC maintenance and consolidation; HL-LHC activities anticipated from LS3. Some tasks will use the next three Year-End Stops.

In parallel, several new installations are under way too, like Awake, Elena, etc.; HIE-Isolde will have more cryo-modules; Isolde controls will be consolidated.

In AD, the final consolidations will be: the integration of cryo-compressors for the RF cavities, together with the temperature of their cold heads; the replacement of the old gas injection system by a new one, similar to LHC-BGI; and the implementation of functions like remote-reset and opening of fast valves.

Important developments, to be considered or started, are: wireless communication for mobile equipment (fieldbus vs. 4G+); industrial ion pump controllers vs. upgrade of old ones; new rad-tol electronics for gauges in the arcs; improved design of sector valve controllers.

Also during LS2: support to vacuum mechanical activities and bake-out; calibration, alarm setting, interlock checks; test and commissioning of all instruments. The workload will require substantial manpower reinforcement, expected to be higher than for LS1.

Injectors

Around the complex PS, during the 3 years before LS2, the pumping groups, ion pumps, and their controllers, will be replaced or upgraded. Linac4 will have a full Sector Test in 2016. During LS2, the connection of Linac4 to the PSB, replacing Linac2, will induce layout modifications.

The SPS ARC re-sectorization should be accomplished before LS2. LSS6 will receive a test station for crab cavities, as part of the studies for High Luminosity LHC. There will be modifications in the layout of several zones, with associated cabling campaigns. The grounding of ion pumps will be consolidated. The racks layout will be reorganized, as it was done in BA4 during LS1.

LHC

In the LSS, the turbo pumping groups and their controllers will be progressively renovated until LS2; the controllers in the ARCs will be changed during LS2. Additional turbo pumps will be installed on the beam kickers.

The controllers for mobile pumping groups must be upgraded, due to obsolescence of the PLCs. The control system of VELO will be renovated, possibly before LS2.

In general, important layout modifications will occur in almost all the LSS, leading to large cabling and connection campaigns, and significant modifications in the racks.

Controls Framework

Following the effort of standardizing the naming of all vacuum controls equipment, the instruments will be next. The implementation of both these naming conventions will involve considerable work on the electrical and mechanical diagrams, databases, and SCADA.

During LS1, a large volume of information was collected, about the layout of vacuum controls and the assets installed in the machines. The organization and formatting of this data is in progress, to later be uploaded into the CERN Layout and Asset-Management databases.

The controls issue-tracker (VTL), together with the handling of vacuum intervention work-orders, is being integrated in Infor-EAM, the CERN asset management system. The tracking of software issues and developments will use Atlassian JIRA. SVN™, used since 2012 for the versioning of SCADA applications, has been extended to the PLC.

SCADA archiving in Oracle (RDB) has been introduced; until the depth of one year is reached, the old archiving system will run in parallel. Some SCADA new features will next be implemented: dynamic calculation and trending of valve reaction times, improved main synoptic warnings for sector valves, and faster acquisition and display of TPG status and parameters.

A new major version (3.14) of WinCC-OA SCADA will be deployed in 2016; it will include a native user interface for mobile devices, dynamic resize of panels, and improved historical data collection.

The UNICOS framework is present in many domains at CERN, who profit from centralized development and support. It is already used for vacuum controls in the Isolde complex and in surface treatment installations.

However, it lacks several vacuum-specific functionalities and field-objects. A convergence project is being established: a minimum set of field objects will be developed and deployed in Isolde complex. In parallel, the SCADA functionalities will be migrated to UNICOS, and progressively integrated. Also, the Time Stamp Push Protocol (TSPP) should soon be implemented.

Linac4 could be a test-bench for the full integration, before global deployment in all accelerators during LS2.

REFERENCES

[1] J.M. Jimenez et al., “LHC Vacuum Upgrade During LS1”, Chamonix 2012 Workshop on LHC Performance, Feb. 2012.
[2] P. Gomes et al., “The Control System of CERN Accelerators Vacuum [LS Activities and new Developments]”, ICALEPCS13, Grenoble, Oct 2011.
[3] G.Pigny et al, "Measurements, Alarms and Interlocks in the Vacuum Control System of the LHC", ICALEPCS15, Melbourne, Australia, Oct 2015.
[4] S. Blanchard et al, "Vacuum Pumping Group Controls based on PLC”, PCaPAC14, Karlsruhe, Germany, Oct 2014.
[5] R. Salemme et al., "Recommissioning of the COLDEX experiement at CERN", IPAC15, Richmond, VA, USA, May 2015.
[6] R. Ferreira et al., "Consolidation of the Controls System of a Chemical Polishing Machine", IEEE-CASE15, othenburg, Sweden, Aug 2015.
[7] F. Mateo et al., "The Upgrade of Contol Hardware of the CERN NA62 Beam Vacuum", ICALEPCS15, Melbourne, Australia, Oct 2015.
[8] S. Blanchard et al., "The new Control Software for the CERN NA62 Beam Vacuum", ICALEPCS15, Melbourne, Australia, Oct 2015.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

DEVICE CONTROL DATABASE TOOL (DCDB)

Pavel Maslov, Matej Komel, Miroslav Pavleski, Klemen Zagar, Cosylab, Ljubljana, Slovenia

Abstract

We have developed a control system configuration tool, which provides an easy-to-use interface for quick configuration of the entire facility. It uses Microsoft Excel as the front-end application and allows the user to quickly generate and deploy IOC configuration (EPICS start-up scripts, alarms and archive configuration) onto IOCs; start, stop and restart IOCs, alarm servers and archive engines, and more. The DCDB tool utilizes a relational database, which stores information about all the elements of the accelerator. The communication between the client, database and IOCs is realized by a REST server written in Python. The key feature of the DCDB tool is that the user does not need to recompile the source code. It is achieved by using a dynamic library loader [1], which automatically loads and links device support libraries. The DCDB tool is compliant with CODAC (used at ITER and ELI-NP), but can also be used in any other EPICS environment (e.g. it has been customized to work at ESS).

INTRODUCTION

In a physics facility containing numerous instruments, it is advantageous to reduce the effort and repetitiveness needed for changing the control system (CS) configuration: adding new devices, moving instruments from beamline to beamline, etc. We have developed the Device Control Database (DCDB) tool [2], which provides an easy-to-use interface for quick configuration of the control system for the entire facility. The DCDB-tool allows the user to quickly generate and deploy configuration for input/output controllers (IOCs) (EPICS start-up scripts, alarms and archive configuration) onto IOCs; start, stop and restart IOCs, alarm servers and archive engines, and more.

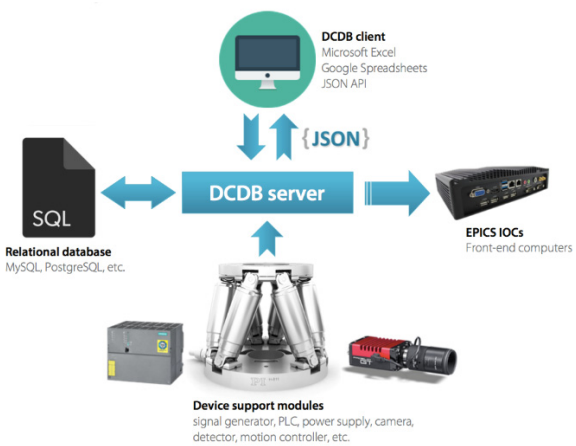


Figure 1: DCDB architecture.

DCDB ARCHITECTURE

The DCDB-tool uses a MySQL relational database. The backend is a typical web-server (Fig. 1), which is realized with a combination of the following Python modules: flask-restful (REST server), sqlalchemy and pymysql (database communication layer), and paramiko (ssh). The front-end is a Microsoft Excel plugin written in C# using .NET technology. IOCs are Linux machines running EPICS and procServ [3]. The client-server communication is based on the exchange of JSON objects (strings).

DEVICE SUPPORT MODULES

To start using the DCDB-tool, simply prepare EPICS device support modules and register them with the DCDB server. In general, device support modules are created in the standard EPICS way. The DCDB-tool introduces the idea of using dynamic macros in the startup scripts. Three additional files, namely: init.cmd, init-pre.cmd and init-post.cmd (Fig. 2) contain macro definitions to be stored in MySQL, and IOC shell commands that register and setup the support module before and after IOC initialization.

```
.
├── db
│   └── BeamPositionMonitor.db
├── dbd
│   └── BeamPositionMonitor.dbd
├── init.cmd
├── init-post.cmd
├── init-pre.cmd
└── lib
    └── linux-x86_64
        ├── libBeamPositionMonitor.a
        └── libBeamPositionMonitor.so
```

Figure 2: Files to deploy.

The procedure of creating device support modules for ITER using special extensions to the *mvn iter* plugin was described in [4]. In the latest version we have tailored DCDB to work in the new EPICS environment developed at ESS [5], since they have a similar approach to using dynamic library loading, and in addition to our setup support dependency resolution and have a simpler unit development workflow. The latter is achieved by creating a Makefile, adding source files and running make. If you want to locally install the module, run the shell command: ***sudo make install***. In order to register the module with the DCDB tool, run ***make dcdb.import***. To delete the module from DCDB, run ***make dcdb.delete***.

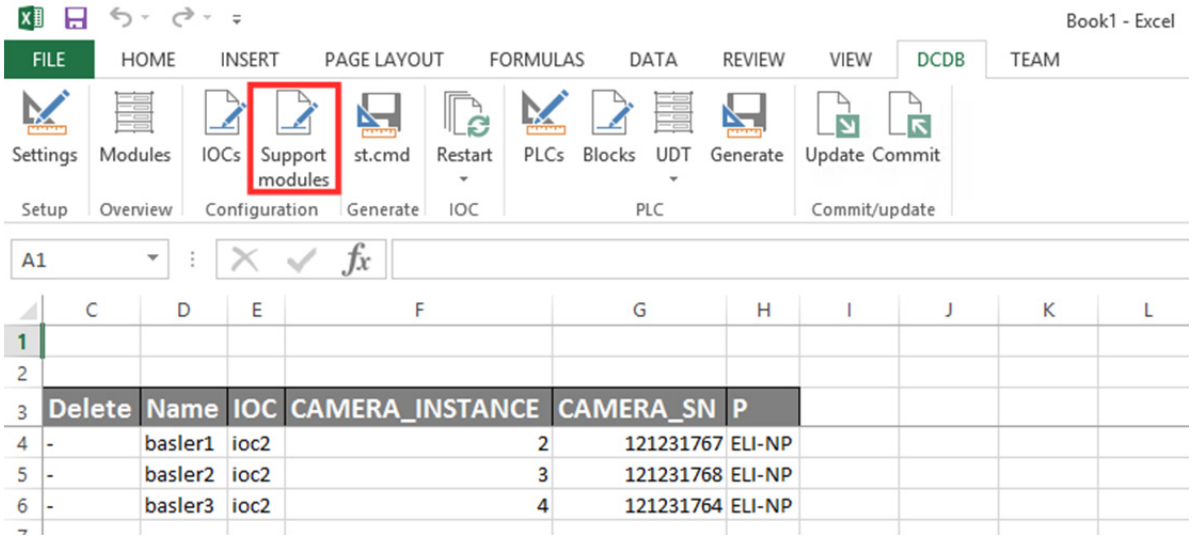
The last two commands require two environmental variables *$(DCDB_SERVER)* and *$(DCDB_PORT)* to correspond with the parameters of a running DCDB server.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

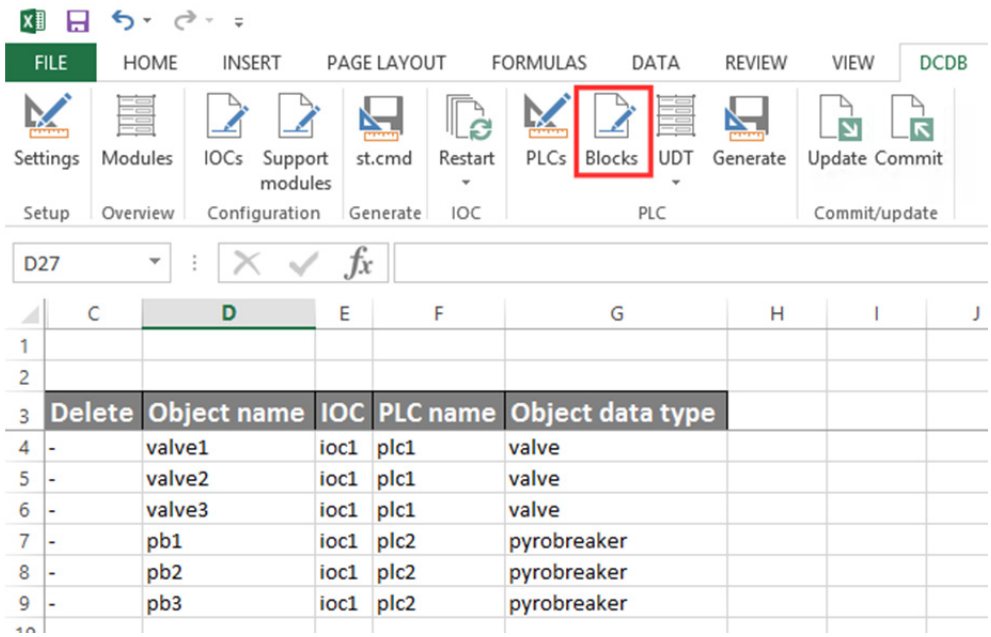
PLC SUPPORT

While doing EPICS integration of Siemens S7 PLCs, the developer spends a lot of time grouping PVs and calculating offsets in order to match EPICS PV types with STEP7 variable types. We have decided to simplify and automate this process using Excel and introduce this feature into the DCDB-tool. There are now sheets in the Excel file where you configure the hardware, generate UDTs and assemble DB blocks. As a result, the DCDB-tool generates an EPICS start-up script and two PLC configuration files to be deployed on the PLC.



	C	D	E	F	G	H	I	J	K	L
1										
2										
3	Delete	Name	IOC	CAMERA_INSTANCE	CAMERA_SN	P				
4	-	basler1	ioc2	2	121231767	ELI-NP				
5	-	basler2	ioc2	3	121231768	ELI-NP				
6	-	basler3	ioc2	4	121231764	ELI-NP				

Figure 3: Microsoft Excel client. Configuration of support modules instances.



	C	D	E	F	G	H	I	J
1								
2								
3	Delete	Object name	IOC	PLC name	Object data type			
4	-	valve1	ioc1	plc1	valve			
5	-	valve2	ioc1	plc1	valve			
6	-	valve3	ioc1	plc1	valve			
7	-	pb1	ioc1	plc2	pyrobreaker			
8	-	pb2	ioc1	plc2	pyrobreaker			
9	-	pb3	ioc1	plc2	pyrobreaker			

Figure 4: Microsoft Excel client. PLC blocks configuration.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

FRONT-END

The user communicates with the REST server via an HMI, which is realized as an Excel add-in (ribbon). It provides a set of buttons with which you can easily edit your support modules' configuration (Fig. 3), configure IOCs (Fig. 5), PLCs (Fig. 4); start/stop/restart IOCs, and more.

As a free alternative to Microsoft Excel, the DCDB-tool also has a Google Spreadsheets client (Fig. 5), which has been published to the Chrome webstore (Fig. 6). The client is written in HTML/Javascript and fully supports the functionality provided by the JSON API.

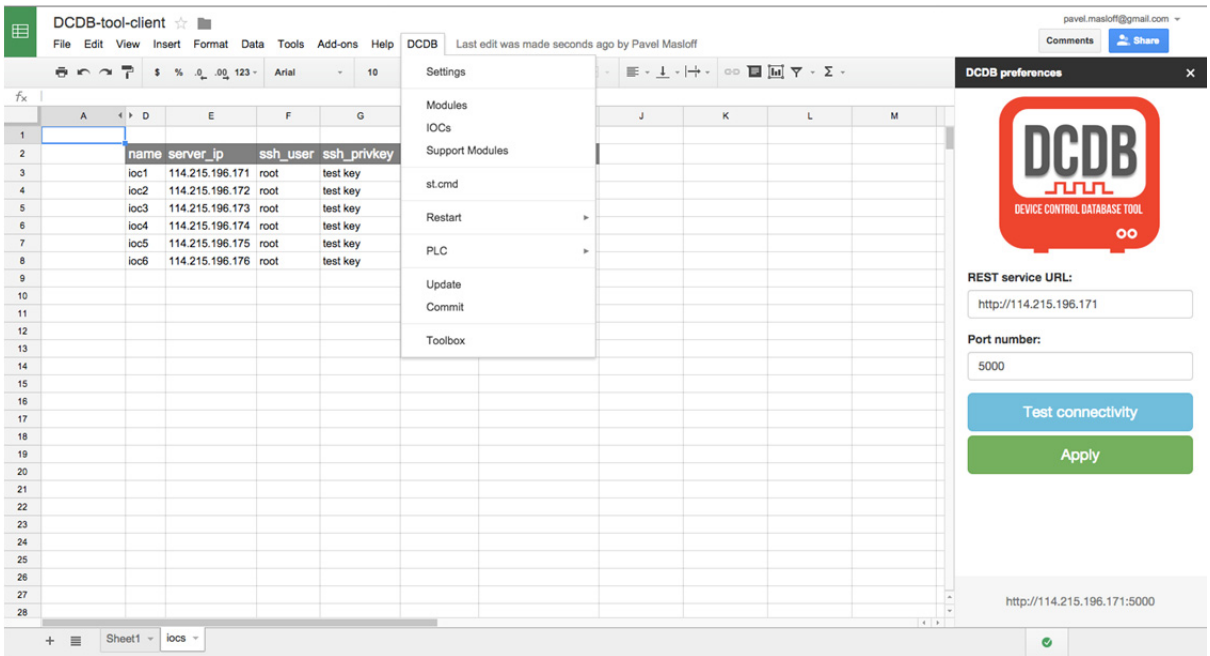


Figure 5: Google Spreadsheets client. IOC configuration sheet.

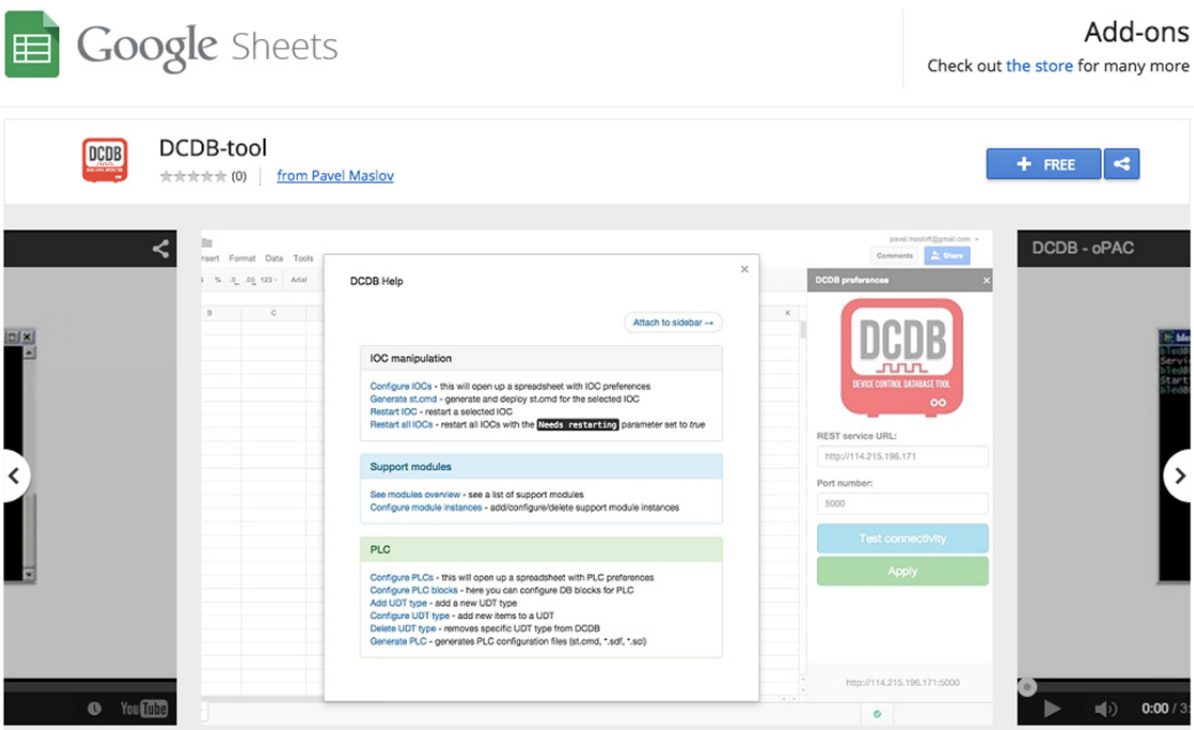


Figure 6: Google Spreadsheets client in the Chrome webstore.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SUMMARY

The DCDB-tool is a powerful control system configuration tool that reduces integration effort and thus, time. It was tested on different platforms including RHEL, Ubuntu, Scientific Linux, Mac OS X and Windows. It is developed with the best practices from the EPICS community, compliant with the CODAC Core System (used at ITER, ELI-NP), but can also be used in any other EPICS environment (i.e. ESS).

ACKNOWLEDGEMENT

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 289485.

REFERENCES

[1] The concept of dynamically loadable device support modules, including the require function is developed by Dirk Zimoch (PSI).
[2] DCDB-tool official web-page: http://users.cosylab.com/~pmaslov/dcdb/
[3] procServ (written by Ralph Lange): http://sourceforge.net/projects/procserv/
[4] DCDB tool, PCaPAC 2014: http://accelconf.web.cern.ch/AccelConf/PCaPAC2014/papers/fpo015.pdf
[5] ESS EPICS Environment: https://ess-ics.atlassian.net/wiki/display/HAR/EPICS+Environment

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DESIGN STRATEGIES IN THE DEVELOPMENT OF THE ITALIAN SINGLE-DISH CONTROL SYSTEM

A. Orlati, M. Bartolini, S. Righini, INAF-IRA, Bologna, Italy
M. Buttu, A. Fara, C. Migoni, S. Poppi, INAF-OAC, Cagliari, Italy

Abstract

The Italian National Institute for Astrophysics (INAF) manages three radio telescopes: the Medicina and Noto dishes and the newly-built SRT. In order to make their capabilities more valuable to the scientific community, we started the DISCOS (Development of the Italian Single-dish COntrol System) project. DISCOS is implemented according to a distributed Component-Container model and hides to the users the differences among the telescopes by presenting the same user interface and the same data format. The complexity of coping with three heterogeneous instruments was handled designing a software development infrastructure with a wide monolithic codebase (libraries, components and generic interfaces), which is completely shared among the three product lines. This design permits to produce new software components with a minimum effort and to set up the same test suites for all the environments, thus leading to an affordable development and maintenance process. In this paper we illustrate the design strategies and the development techniques used to realize and optimize this common control software. We also provide a description of the project status and future plans.

INTRODUCTION

The National Institute for Astrophysics (INAF) manages three radio telescopes in Italy: The Sardinia Radio Telescope (SRT) [1] and the Medicina and Noto 32-m dishes. The newly-built SRT, located in the Sardinia island, was inaugurated in 2013. Since the early stages of the project, staff from all the three telescopes has been involved in the development of the SRT control software. This forced us to cope with both the development of a brand-new system and the maintenance of the already existing telescopes, compelling the involved personnel to learn and enhance their competence in distinct and heterogeneous systems based on completely different technologies. Developing a control software to be installed at all the telescopes was the natural and straightforward approach to this problem.

This idea was formalized with the creation of the Development of Italian Single-dish COntrol System (DISCOS). The project aims to provide all three Italian radiotelescopes with a common infrastructure that increases their capabilities and the usability for the scientific community and, at the same time, optimizing the efforts made by the technical staff for its development and maintenance.

DISCOS is based on ALMA Common Software (ACS) developed at ESO for the ALMA project [2]. ACS implements a Component-Container model via CORBA (Common Object Request Broker Architecture). It provides a set of tools, libraries and development patterns that hide the complexity of CORBA. This approach permits to reduce the time required for coding and development. The framework also supports both real and non-real time platforms for C++, Python and Java.

The core of the project was completed during the main development stage. The SRT and Medicina installations are now fully operational and supervise all the telescope operations. The Noto antenna is also equipped with a preliminary version, which is going to be finalized in a few months.

In this paper we describe the salient design choices that allowed us to reuse the most part of the code for all the production lines and to hide the instrument complexity under a common infrastructure. We also illustrate the maintenance and development workflows, as they are fundamental part of the DISCOS design.

THE TELESCOPES

The INAF radiotelescopes share some common aspects but are in general different from one another. Medicina and Noto have passed through years of development and updates, while SRT is the result of an organic employment of state-of-the-art technologies. Table 1 recaps the main characteristics and the differences, listing all the major sections – servo systems, front-ends, back-ends. All the telescopes are presently participating in the VLBI network. Medicina and Noto are also offered to the scientific community as single-dish facilities (SRT will very soon be, as well).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Table 1: Fact Sheet of the Three Italian Radiotelescopes

	SRT	Medicina	Noto
Main mirror	shaped profile, 64 m	parabolic profile, 32 m	parabolic profile, 32 m
Optical configuration	Gregorian	Cassegrain	Cassegrain
Mount	Altazimuth, fully steerable 12 motors + cable wrap	Altazimuth, fully steerable 4 motors	Altazimuth, fully steerable 4 motors
Antenna Control Unit (main servo system)	Beckhoff PLC ethernet vendor protocol	VxWorks based PC ethernet vendor protocol	VxWorks based PC ethernet vendor protocol
Primary Focus	three degrees of freedom INAF protocol	three degrees of freedom INAF protocol	three degrees of freedom INAF protocol
Secondary Focus	six degrees of freedom ethernet INAF protocol	five degrees of freedom ethernet INAF protocol	five degrees of freedom RS232 vendor protocol
Active Surface	1008 aluminium panels 1116 actuators rs485/ethernet vendor protocol	Not available	240 aluminium panels 244 actuators rs232 vendor protocol
Receivers (RF bands*)	(0.305-0.410) (1.3-1.8) (5.7-7.7) (18.0-26.5), 7 feeds GPIB and ethernet INAF protocol	(1.35-1.45) (1.595-1.715) (2.2-2.36) (4.30-5.80) (5.90-7.10) (8.18-8.98) (18.0-26.5), 2 feeds GPIB, ethernet and RS232 various protocols	(0.317-0.320) (1.40-1.72) (2.20-2.36) (4.70-5.05) (8.18-8.58) (22.18-22.46) (39.0-43.3) GPIB and RS232 various protocols
Backends (Bandwidth*)	TotalPower [continuum] (up to 2.0), 1-1000 ms, 14 inputs XARCOS [spectro-polarimetry] (up to 0.125), 10 s, 2048 bins, 14 inputs Roach [spectro-polarimetry] (0.512), 10-1000 ms, 8192 bins, up to 14 inputs DFB [spectro-polarimetry] (1.024), 1-4000 ms, 8192 bins, 4 inputs	TotalPower [continuum] (up to 2.0), 1-1000 ms, 4 inputs XARCOS [spectro-polarimetry] (up to 0.125), 10 s, 2048 bins, 14 inputs	TotalPower [continuum] (up to 2.0), 1 ms, 4 inputs

* Frequencies are expressed in GHz.

A COMMON STRATEGY

Broad categories like development, operations and maintenance are very demanding, especially when highly qualified expertise is required. Our common infrastructure enhances the quality and throughput of our work.

Operations

Operations are immediately affected by this strategy; as operators can now manoeuvre the three telescopes facing very slight differences in the user interface, their require a single training. Similarly, astronomers are presented with the same observing modes and the same data format at every telescope. Operation manuals and documentation are written with a common effort, sharing most of the information among the infrastructures.

Maintenance

Maintenance consists in code debugging, configuration changes or even complete replacements of software components. Since regular telescope activities require to minimize the time devoted to maintenance, the relative work must be carried out with proper planning and efficiency. This is eased by the use of the same ticketing

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

system to track software bugs and to monitor the antenna status. A common workflow, involving every collaborator, is employed to fix malfunctions. As the needed tests can be performed at any telescope, we can choose the one providing an adequate scheduling to test patch releases.

Development

The development of new software features is a time-consuming activity that also requires testing time. The DISCOS common platform highly reduces these needs. New components can be deployed indifferently at all the antenna systems, so that separate units can independently carry out the development work. Moreover, once a new piece of hardware is commissioned and integrated into the software system of one of the telescopes, it can be replicated at the other telescopes with little effort.

DESIGN

The design of the DISCOS control software highly relies on the ACS patterns and services. In the ACS model, the basic unit performing a task is a component. It either controls a simple device or executes astronomical computations. Each component exposes an interface or list of capabilities and is individually configured to determine its exact behaviour inside the system. In our project the components are organized into subsystems or packages according to their functional affinity, thus forming system segments that are independent and capable to run as stand-alone groups. During the early development efforts this configuration turned out to be very valuable for our geographically-spread team. Each developer, in fact, could focus on one single subsystem, internal milestone or scheduled test, without being concerned by the delays in the development of the other packages.

Supporting three different telescopes is of course the most challenging goal of the project. This can be reached by expanding the common code, the part of our project that can be reused and deployed at all sites, as much as possible. The station-specific modules, then, consist essentially in the low-level and no-logic control of the devices and of the telescope hardware. According to this approach, a careful design was crucial to ensure that all the peculiarities of the telescopes fitted with the business logic at a higher level and with the common parts of the control software. The Italian telescopes differ in various features (see Table 1), which can be grouped into two categories:

- A given telescope functionality is implemented via different hardware devices at the three radiotelescopes: they could differ in vendor, working mode and communication protocol. The main servo systems of our antennas are an excellent example of this case; in our software implementation such devices were modelled according to an ideal or generic interface that defines the minimal set of attributes and methods required by the control logic. Every other component handles the device adopting this interface.
- An apparatus is installed at one or two telescopes, but it does not apply to other antennas. For example, the SRT and Noto antennas are equipped with an active surface, while the Medicina one is not. These categories are handled through configuration files describing the available components and functionalities; the system is then capable of detecting if an operation is allowed under the present configuration. If the operation is not fundamental for the running observation, no error is raised.

Development Workflow

In recent years we have tried to formalize the development workflow of new components, resulting in clearer and more maintainable code. As a first step we decided to add the ability to run components by simulating the hardware layer whenever possible. This led us to split the component development into:

- Development of a hardware simulation server
- Development of a standalone hardware communication library
- Development of the ACS component exploiting the library

The immediate benefit of this procedure is to have a simulated hardware environment that permits to run our tests automatically. We consequently adopted integration tests to validate component interfaces and regression tests when fixing bugs in the system. Furthermore, we set up an infrastructure - both Python and C++ are supported - that generates a testing skeleton for new components and runs unit tests collecting xUnit-formatted results.

A new approach was adopted also in the configuration of the software environment. The full stack deployment was formalized via code, resulting into the AZDORA project [3]. It uses virtualization and provisioning technologies in order to setup and run a fully configured DISCOS environment, which can be used both for development and production workstations. This uniformity among development, testing and production environments is crucial in assessing the software stability as much as possible before the on-field production installation.

The just-described features have made it possible to adopt Continuous Integration practices in the development process. A Jenkins server [4] is used in this case. At present we run nightly builds of the trunk branch of each telescope and of every still-maintained release. The proper developer is alerted in case problems are detected; occasional build or integration errors are fed into our bug-tracking system. Each build target results into a ready-to-install package that can be safely used in production environment. This infrastructure also permits test automation and to collect the results on the whole project. In the near future this will give access to new quality metrics and to the possibility of automatically

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

running acceptance tests upon every new code subscription.

Maintenance Workflow

System bugs are handled with regression tests: each time a bug is found, the developer writes a test that reproduces the buggy behaviour, then the fix is not committed into the patch release until the test executes correctly.

We also adopt a *major.minor.patch* nomenclature. The major version zero (0.y.z) is for initial development, when anything may change at any time: the public APIs, the operator input commands and the schedule grammar are not to be considered stable. New major versions are issued when non-backward-compatible changes are necessary and are planned way in advance. Minor versions are feature releases, so they add functionality in a backward-compatible manner. We also have non-production ready versions that get an additional qualifier: beta and release candidate (RC). The beta versions are to be tested by advanced users, while the RCs are aimed at being tested by a group of astronomers.

Release notes are written by developers in a user-friendly manner and can be found within our documentation portal [5], while technical release notes are automatically extracted from the bug-tracking and features system used to manage the development process [6].

FUTURE PLANS

The DISCOS user interface is currently mainly composed of unrelated GUI applications showing textual information about the main telescope components via *ncurses* based panels. Every interaction with the control software is performed via a command shell that serially reads telescope commands in a synchronous *Read-Evaluate-Print* loop. A remote access to the interface is offered via VNC connections. This setup is effective for early operations; in view of the telescopes future activities we are designing a user-friendly interface in the form of a web application. This will provide the users with a more homogeneous experience, efficiently enabling remote operations. This approach will likely require other software layers, implementing the authentication and authorization of the users and new parameters-publishing systems. This will also lead to a possible integration with other telescope management software tools, such as the proposal submissions handling system and the schedule creator. This work could also technologically enable the creation of a centralized operation centre for all the radiotelescopes.

Further development in the near future will involve the adoption of the most recent ACS version, the adoption of new technologies for high-rate data transfer, the extension of test coverage to a bigger portion of the code and the automation of tests during the build process.

CONCLUSIONS

DISCOS is serving the scientific community by enabling observations at the Italian INAF radiotelescopes, and while being still subject to some major changes, it is running and effective. The growth of the project and its dissemination among different sites has forced the adoption of a common development strategy and has led to the adoption of best practices from the software industry such as Test Driven Development and Continuous Integration in order to assess software quality and to ensure the stability of successive software releases. This also enables an effective maintenance process, which is a strong assurance for the coming years of development and operations.

REFERENCES

[1] G. Grueff et al, “Sardinia Radio Telescope: the new italian project”, Proc. SPIE vol 5489, p 773-783 (2004)

[2] G. Chiozzi et al., “The ALMA common software: a developer-friendly CORBA-based framework”, Proc. SPIE vol 6274, September 2004, p. 205 (2004)

[3] Azdora project github page: https://github.com/discos/azdora

[4] Jenkins project website: https://jenkins-ci.org/

[5] DISCOS documentation portal: http://discos.readthedocs.org/en/latest/developer/releasenotes/releasenotes.html

[6] DISCOS project bug tracking system: http://www.med.ira.inaf.it/mantisbt/roadmap_page.php

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TANGO INTEGRATION OF A SPECIFIC HARDWARE THROUGH HTTP-SERVER

A. Panov, A. Korepanov

The Budker Institute of Nuclear Physics, Novosibirsk, Russia

Abstract

MAX IV are new synchrotron third generation. MAX IV synchrotron consist of 1.5 GeV storage ring, 3.0 GeV storage ring and linac; it is located in Lund, Sweden. Structure of storage rings contains several pulse magnets (kicker and pinger). Control system of pulse power supplies based on LTR crate with several modules (ADC, DAC, input/output registers etc.). LTR crate is a product Russian firm L-CARD. In order to communicate with crate native LTR-server is used. LTR-server is a Windows application based on use of sockets. Control system of MAX IV and Solaris uses TANGO. For integration LTR-crates in final structure, special software gateway is used. This gateway is a set of several specific Windows applications implemented by using Qt5 libraries. Gateway allows communicating TANGO-server with crate through built-in HTTP-server.

SPECIFIC HARDWARE

The MAX IV facility included two storage rings for the production of synchrotron radiation: the 3 GeV and 1.5 GeV storage rings [1]. Both rings will be operated with top-up shots supplied by the 3.5 GeV MAX IV linac acting as a full-energy injector at up to 10 Hz repetition rate. At the commissioning phase, the single dipole kickers will be used for the injection in the both rings. In addition, pulsed vertical kickers (pingers) are needed for performing tests with the stored electron beam.

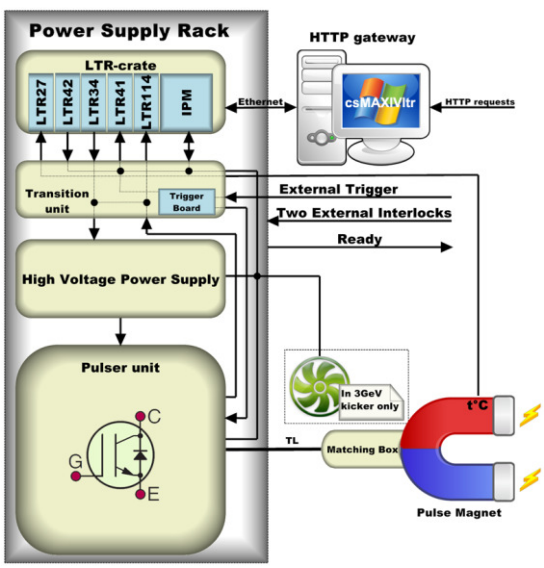


Figure 1: General pulse power supply structure.

Table 1: Pulse Magnet Systems Specifications

	1.5GeV		3GeV	
	kicker	pinger	kicker	pinger
Integrated field, Gm	143	75	450	110
Peak current, A	1060	2050	3600	2600
Pulse length, μs	0.64	0.64	3.5	3.5
Maximum rep. rate, Hz	10	1	10	1
Pulse to pulse timing,ns	±5	±5	±5	±5
Pulse to pulse repeatability, %	±0.1	±1	±0.1	±1

The requirements for the pulse magnet systems are listed in Table 1. Structure of all system (power supply with control electronics and magnet) are illustrated on Fig.1.

Control system of the pulse power supply based on 19" LTR-crate (LTR-EU-16). It is a product of a Russian firm L-CARD (http://lcard.ru). Base structure of the LTR-crate are represented on a Fig. 2.

This crate include FPGA Cyclon EP1C3 for

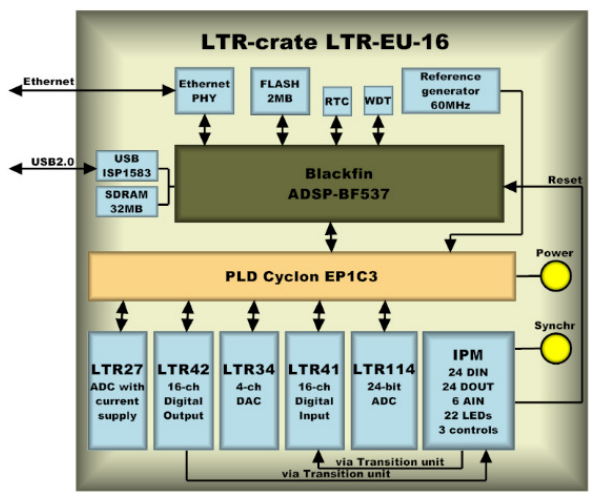


Figure 2: Power supply control system (LTR crate with modules) block diagram.

communication with installed in crate modules via proprietary interface and digital signal processor Blackfin ADSP-BF-537 for preliminary data processing. DSP operates under control of Visual DSP++ Kernel (VDK) [2]. Sources of a DSP firmware in free access and can be modified for individual requirements. Crate can communicate with PC over USB2.0 or Ethernet (TCP/IP) via L-CARD specific protocol.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The following modules are installed in LTR crate:

1. LTR41 (16 channels digital input module);
2. LTR42 (16 channels digital output module);
3. LTR114 (24 digits ADC module);
4. LTR34-4 (4 channels DAC module);
5. LTR27 (thermo ADC);
6. IPM (Interlock Processing Module);

All modules (except IPM) was produced by L-CARD. Interlock Processing Module was made in Budker Institute of Nuclear Physics. It communicate with LTR-crate over digital inputs/outputs by LTR41/42. Brief characteristics of modules and their targets of using are listed below.

LTR41 Digital Input Module

LTR41 is 16-channels input module for TTL/CMOS-signals (with 5 V-logic) and also for current logical signals (up to 25 mA) with per-channel galvanic isolation. It is used to synchronize and to get current states and interlocks values from Interlock Processing Module.

LTR42 Digital Output Module

LTR42 is a 16-channels output module with per-channel galvanic isolation. It is used to send control commands to the interlock processing module and to generate "heartbeat" signal, which is used by the algorithm of checking the status of connection to control PC.

ADC Module LTR114

LTR114 module is a 24-digits ADC with sampling frequency up to 4 kHz, differential input and dynamic commutation to 16 channels. It has software controlled input sub-ranges ±10 V, ±2 V, ±0.4 V independently for each channel. LTR114 is used to measure high voltage and to monitor settings of the HVPS (voltage and current).

DAC Module LTR34-4.

LTR34-4 module is a 4-channel 16-digits DAC with output frequency up to 500 kHz in voltage range from -10 V to +10 V. It is used to set up the HVPS output voltage and current values.

Thermo-ADC LTR27

Unit LTR27 is a mezzanine module with possibility to install up to eight submodules. In current configuration a single submodule H27-R250 for resistances measurement is used. Frequency of data yield is up to 100 Hz. The unit is used to monitor the temperature of the magnet's vacuum chamber by measuring resistance of thermoresistor Pt100.

CONTROL SYSTEM SOFTWARE

L-CARD produce software package for operate with crate and modules. First, this package include LTR-server (for Windows PC) and ltrd daemon (for Linux PC). These applications necessary for initializing operations, obtaining data from crate in uniform format and allocating data according to list of modules. In addition, software package include necessary library (C++/LabView) for implementation individual application for communicate with modules over LTR-server or ltrd. There is utility named UTS for test purposes and there is L-Graph2 application for collecting and visualization data from various ADC.

Set of modules is a low layer control and monitor system of pulse power supply. High layer is a Tango server with appropriate operator screen. Therefore middle layer for communicate Tango server with LTR-crate is necessary. This middle layer was realized in BINP and named csMAXIVltr. Main goal of this software part is be gateway between Tango and LTR. Moreover, csMAXIVltr provide several procedures for normal work of the power supply.

csMAXIVltr consist of four Windows application:

1. Starter, which start and control all other applications;
2. LTR-target – main application, which communicate with LTR-server;
3. HTTP-server – for communicate with Tango server;
4. LTR-server – L-CARD application for communicate with LTR-crate.

Basic structure of the csMAXIVltr are presented at Fig. 3.

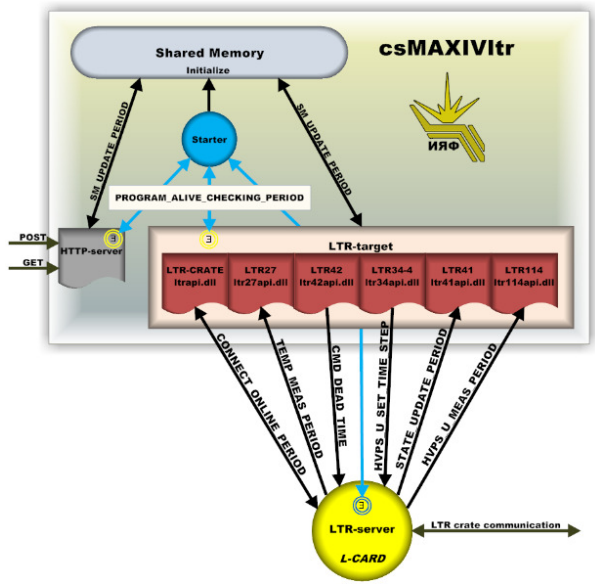


Figure 3: Basic structure of software gateway.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

All BINP parts of the csMAXIVltr was realized with Qt5 framework for operate on Windows machine. Starter, LTR-target and http-server interact together over shared memory with specific period (SM_UPDATE_PERIOD) which amount approximately 100 ms. Each module in LTR-target operate in its thread and interact together with QT native SIGNAL/SLOT mechanism.

Http-server

Http-server based on QHttpServer by Nikhil Marathe [3]. This csMAXIVltr part provide access Tango server to main actions (switch on/off, reset), settings (current/voltage), states and interlocks. Procedures for writing and reading of csMAXIVltr internal settings were added in last version.

Http-server ensure the fulfilment of two basic methods: POST and GET. POST method are used for control pulse power supply: switch on/off high voltage power supply, reset interlocks, disable or enable trigger and write voltage and current settings to HVPS. Each POST-request should contains single command. So if it is necessary to send a number of commands to the system then the sequence of POST-requests should be generated. If POST-request contains a few commands then only the first command is processed and other are ignored.

GET method are used for reading various measurements (voltage from divider, feedback levels, temperature), states and interlocks. Moreover, by GET method operator can modify internal csMAXIVltr settings (IP address, timeouts, average size and many other).

Http-server responding with identical answer on POST or GET request. This response consist of two parts: simple html-page (yellow on the Fig. 4) and last line with all required parameters (green on the Fig. 4). “Yellow” part required only for debugging or testing without Tango operator screen. “Green” part are implemented in POST data format – …&ParameterName=ParameterValue&… This part formatted like comment in html page.

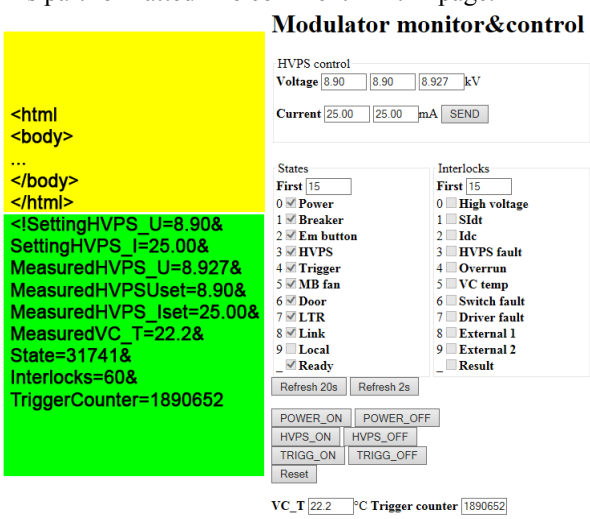


Figure 4: Typical Http-server response.

Read request structure of a csMAXIVltr parameter via GET are look like http:\\x.x.x.x\~PATH??

Write request structure of a csMAXIVltr parameter - http:\\x.x.x.x\~PATH=VALUE! where PATH is variable combined from APPLICATION_NAME and PARAMETER_NAME.

In additional http-server supports authenticate header for security operate with control system. Login and password can be changed by configuring csMAXIVltr.

Tests

Several basic tests was performed for operability checking of the system. Apache Benchmark output are represented at Listing 1.

```
This is ApacheBench, Version 2.3 <$Revision: 1663405 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
Benchmarking 192.168.1.6 (be patient)
Server Software:
Server Hostname:        192.168.1.6
Server Port:            80
Document Path:          /
Document Length:        0 bytes
Concurrency Level:      10
Time taken for tests:   3.627 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      110000 bytes
HTML transferred:       0 bytes
Requests per second:    275.73 [#/sec] (mean)
Time per request:       36.267 [ms] (mean)
Time per request:       3.627 [ms] (mean, across all concurrent requests)
Transfer rate:          29.62 [Kbytes/sec] received
Connection Times (ms)
              min mean[+/-sd] median   max
Connect:        0    0   0.4      0       1
Processing:    32   36   1.8     36      44
Waiting:       32   36   1.8     35      44
Total:         33   36   1.8     36      44
Percentage of the requests served within a certain time (ms)
  50%     36
  66%     36
  75%     37
  80%     37
  90%     38
  95%     39
  98%     42
  99%     43
 100%     44 (longest request)
```

Listing 1:Apache Benchmark output.

Control test results are represented on a Fig. 5. Histogram corresponds to result of write “action” command tests. Delta is time between operator action (mouse button click on “Reset” button at html page) and actual control system operation (reset interlocks by IPM). This time summarize three large latency: shared memory update time by http-server (≤80ms), read time from shared memory by LTR-target (≤80ms) and send time to IPM over digital output module by using special robust protocol (≤160ms). These times can be decreased by configuring csMAXIVltr, but this action increase CPU load.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

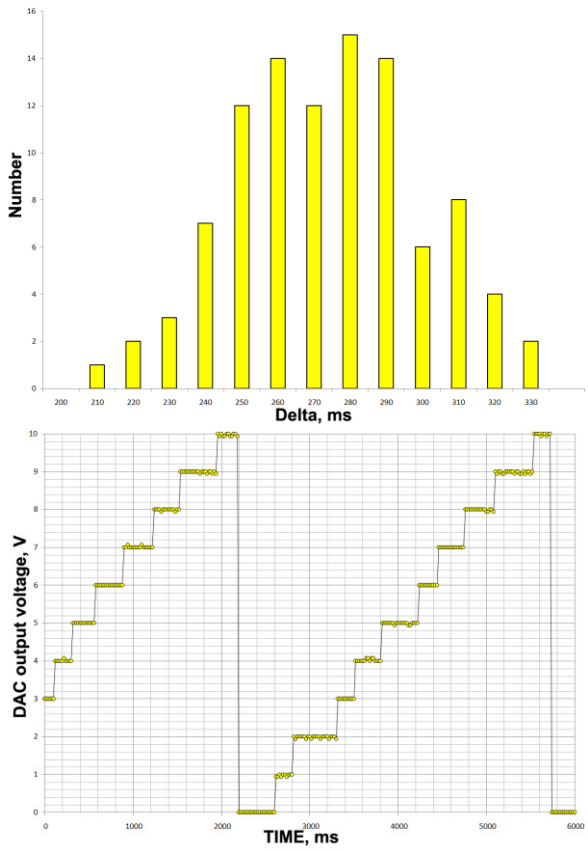


Figure 5: Write command test results.

Upstairs graph represent continuous sequence of voltage setpoints from zero to 10 Volts with 300 ms time step.

TANGO VIEW OF PULSE POWER SUPPLY

Tango-server for pulse power supply was realized by Cosylab. It contains 29 attributes for operator display level and additional 35 attributes for expert. Majority of operator view attributes (20) are Boolean which indicate state or interlock.

Power supply control state machine are illustrated at Fig. 6. This diagram not include UNKNOWN and INIT states because they define only internal Tango server implementation. All other state defining from status of the pulse power supply.

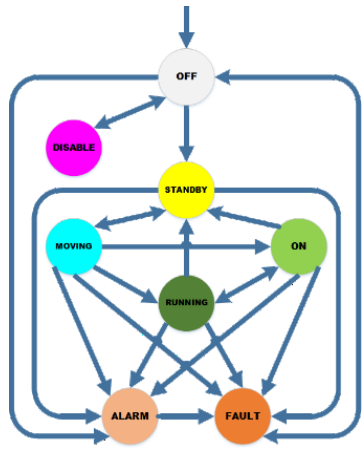


Figure 6: Power supply state machine in Tango view.

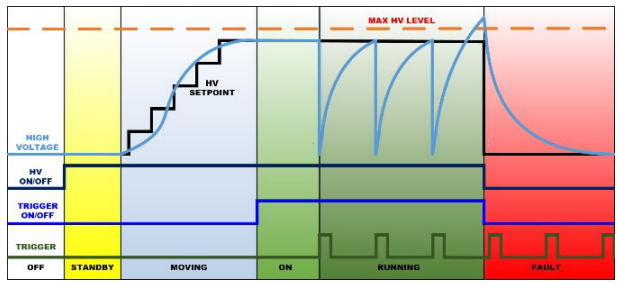


Figure 7: Basic conditions for Tango state definition.

At the Fig. 7 are represented basic conditions for Tango main states definition. Trigger pulses are external synchro signal for power supply start. FAULT state condition came when high voltage exceed legal level for example.

Software gateway based on the very simple http-server allow to link LTR-crate with Tango approach for successfully control of pulse magnet power supplies.

REFERENCES

[1] S. C. Leemann, A. Andersson, M. Eriksson, L.-J. Lindgren, E. Walle´n, J. Bengtsson, and A. Streun, Phys. Rev. ST Accel. Beams 12, 120701 (2009).

[2] Visual DSP++ Kernel (VDK) description http://www. analog.com/media/en/dsp-documentation/legacy- software- manuals/50_vdk_man.3.1.pdf

[3] Q H ttp S e r v e r d e s c r i p t i o n a n d t e s t s http://blog.nikhilism.com/2011/02/qhttpserver-web-apps-in-qt.html

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MEASUREMENTS, ALARMS AND INTERLOCKS IN THE VACUUM CONTROL SYSTEM OF THE LHC

G. Pigny, F. Antoniotti, J.P. Boivin, N. Chatzigeorgiou, J. Gama, P. Gomes, P. Krakowski, H. Pereira, H. Vestergard
CERN, Geneva, Switzerland

Abstract

In the LHC beam pipes and cryostats, the pressure measurement covers a wide range, from 1500 mbar down to 10^{-11} mbar and even lower. If vacuum deteriorates, alarm signals are generated and sent to other systems, like cryogenics, accelerating cavities, kicker magnets. In addition, an unacceptable pressure rise in beam pipes generates interlocks to close the adjacent sector valves, thus isolating the sector, so that the pressure rise does not propagate. This pressure interlock is simultaneously forwarded to the beam interlock system.

This paper describes the instrumentation, the interlocks and alarms logic used in the vacuum control system of the LHC. We analyze the possible signal degradation caused by ionizing radiation or due to cable length, shielding and grounding. During the first LHC long shut down, several corrections were applied to mitigate radiation effects and improve signal integrity. The tests performed for the vacuum control commissioning and LHC machine checkout are also presented.

VACUUM MEASUREMENT

Vacuum gauges in accelerators have to cover about 15 decades of pressure. The direct pressure gauges, which measure the force per unit area, are gas-type independent. They cover the rough and medium vacuum ranges. The indirect pressure gauges, which measure a pressure-dependent property of the gas, are gas-type dependent. They can cover a wider range of vacuum, from the rough to the ultra-high vacuum range [1].

The membrane gauge covers the range from 0 to 2 000 mbar. It is composed by a thin silicon crystal layer used as a membrane, whose deformation induces a piezo-resistive effect on the four resistances of a Wheatstone bridge. The resistance variation is a function of the pressure. The signal to be read across the bridge is in the mV range.

The Pirani gauge covers the range from 1 000 to 10^{-4} mbar. The measurement is based on the dependence of the thermal conductivity on the gas pressure. A filament is heated and maintained at a constant temperature. The current needed to maintain this temperature depends on the gas conductivity and therefore is a measure of the pressure. The signal delivered by the amplifier is in the Volt range. Above 1 mbar the measurement becomes non-linear, and depends on the gas species

The Penning gauge covers the range from 10^{-5} down to 10^{-11} mbar; it is also called inverted magnetron cold cathode gauge. It is composed by a thin cylindrical anode rod, surrounded by a cathode. A high voltage (HV) potential of 3kV creates a high electric field between the two electrodes, which favors the ionization of gas molecules. Around the electrodes, a permanent magnet produces a field that forces the free electrons into spiral paths, thus increasing the probability of ionizing more molecules. The ionization current is a function of the pressure. In the low range of measurement, the current can be as low as the pA; a high quality HV triaxial cable is used for signal transmission.

In the range from 10^{-5} down to 10^{-9} mbar, the Sputter Ion Pump can be used to measure the pressure, in addition to its pumping function. It is composed by several Penning cells, with the particularity of Titanium cathodes and a potential of 7 kV. The principle of vacuum measurement is the same as for the Penning gauge. In the low range of measurement, the current can be as low as 100 nA, and high quality HV triaxial cable is used for signal transmission.

Finally, to measure down to 10^{-12} mbar, the Bayard-Alpert gauge is used. A heated filament emits electrons which are accelerated by the grid potential (+150 V). On their path, these electrons ionize gas molecules, which are gathered by a collector. The collector current is a measure of the ionization and therefore of the gas pressure. It can be as low as 100 fA. High quality triaxial cable is used for this measurement.

INTERLOCKS AND ALARMS

Alarms are binary signals sent to the operators' console, e-mail or SMS, to draw their attention to particular equipment status or situations. There are also alarms directly cabled to other systems, for information or to be used in their control logic.

Interlocks are binary signals used within the vacuum system to prevent entering undesired states. They have a higher priority than the normal process logic or the operator's commands. Apart from the beam permit interlock, the vacuum control system does not interlock any other external systems directly.

The detection of a pressure rise, above a predefined threshold, can be used to produce alarms or interlocks. The gauges used for alarms or interlocks can be the Membrane, Pirani, Penning and Ion Pump [2].

Interlocks for the Vacuum Sector Valves

The beam pipes are divided in "vacuum sectors" that can be isolated by "sector valves" (VVS), to avoid the propagation of leaks over a large volume.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Upon the detection of a pressure rise above a predefined level, these valves will close in 1 to 3 s, depending on the model.

The LHC itself is divided in 8 "machine sectors". Each one consists of a curved section (ARC + DS) of about 2.8 km where all magnets are cryogenic and kept at 1.9 K in a continuous cryostat. At each side of the continuous cryostat, there is a long straight section (LSS) of some 300 m. Here, a sequence of cryogenic stand-alone magnets (SAM) alternates with sectors of bare beam pipe at room temperature, before arriving to the experimental or technical caverns.

The continuous cryostat, the room temperature sectors and the SAMs have vacuum sector valves at their extremities. During the recent long shutdown (LS1), one of the targets was to minimize the risk of a pressure wave created in the ARC to propagate and damage the SAMs or other equipment.

The sector valves at the end of the continuous cryostat were improved for a faster closing (less than 1 or 2 s). New Penning gauges were added further upstream (Q12, Q13) of the valves, to interlock them on an early detection of the pressure wave front.

In general, a sector valve is interlocked by several neighboring pressure measurements, either to make it close or to enable its aperture. The pressure thresholds can be set individually for each gauge, but preference is given to an uniformity of values all around the machine. A valve closed by a pressure interlock will also propagate it to its 2 neighbor valves, to further isolate the faulty sector.

Figure 1 shows the typical interlock configuration between two vacuum sectors at room temperature.

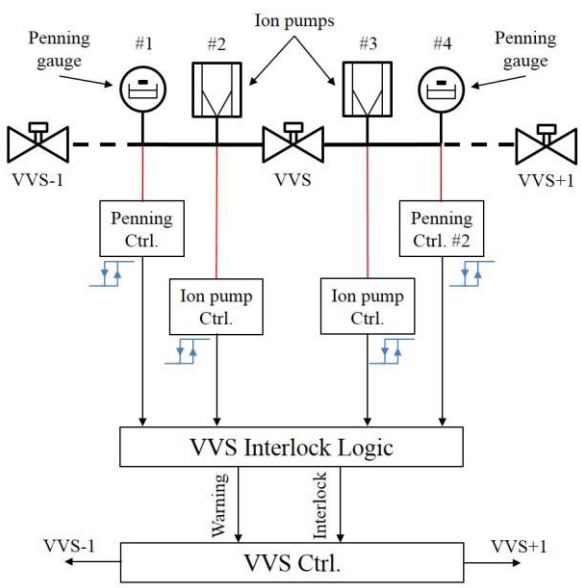


Figure 1: VVS interlock configuration.

Table 1 summarizes, for different configurations, the thresholds and the fraction of neighboring gauges needed OK, to allow or keep valve opening. The valves at the end of the continuous cryostat have an additional condition on the 2 new gauges.

Table 1: Thresholds and Logic for the Sector Valve Interlocks

	Rising [mbar]	Falling [mbar]	# to allow open	# to force close
continuous cryostat	2.10^{-6}	1.10^{-6}	4 OK /4 + 2	3 bad /4 + 2
room temperature	2.10^{-6}	1.10^{-6}	4 OK /4	3bad /4
SAM	2.10^{-6}	1.10^{-6}	3 OK /3	2 bad /3
dump lines	7.10^{-5}	5.10^{-5}	4 OK /4	3 bad /4

Interlocks to the Beam Interlock System (BIS)

As a closed sector valve would be in the beam path, and thus damaged, the beam must be stopped before that may happen. Indeed, the beam can be extracted from the LHC and dumped in less than 300 μs, which is much faster than any valve closing (1 - 3 s).

Therefore, a pressure interlock triggering a valve closing is simultaneously forwarded to the BIS [3]; and so is the information of any valve losing the open state, due to a neighbor valve closing or for any other reason.

Alarms to the Cryogenic System

If the pressure rises, the thermal isolation degrades and the cryogenic system might need to prevent the start of some cryo-compressors or even to stop them.

Each insulation volume around the cryogenic distribution line (QRL) or the cryo-magnets is equipped with a triplet of gauges (Membrane, Pirani and Penning). The first two are used to generate alarms to the cryogenic system, at 500 and 0.1 mbar, respectively.

Alarms to the RF & ADT Systems

If the vacuum degrades inside a radiofrequency (RF) accelerating cavity, the high electrical field may ionize the residual gas and damage the cavity.

For each of the 8 cavities in each beamline, the RF system has its own Penning gauges, to follow the pressure inside the cavities, and to produce the necessary interlocks. These are not available on the vacuum system.

The pressure in the beam pipes outside of an RF module (group of 4 cavities) is used to produce an alarm, combining one Penning gauge ($1 - 4.10^{-7}$ mbar) and two ion pumps (1.10^{-6} mbar).

For the transverse dumpers (ADT), located upstream and downstream of the modules, the RF system receives the analog reading and a Penning alarm ($1 - 5.10^{-7}$ mbar).

Alarms to the Kickers Systems

The injection kicker magnets (MKI) insert the beam coming from the transfer lines into the LHC orbit; there are 4 MKI tanks at point 2 and 4 other at point 8.

The dilution kicker magnets (MKB) sweep the beam in an 'e' shape, distributing it over a wide dump surface; there are 5 MKB tanks on each side of point 6.

As the kicker magnets are powered by high-voltage pulses, if vacuum degrades, galvanic isolation becomes a concern and the kickers may have to be stopped. If pressure rises in the MKI, there cannot be injection into the LHC. If

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

pressure rises in the MKB, the beam should be dumped before the situation gets worse. Table 2 shows the thresholds and the fraction of neighboring gauges needed to send an alarm to the kicker systems.

Table 2: Vacuum Information Sent to the Kicker Systems

	Rising [mbar]	Falling [mbar]	alarm sources	# to send alarm
MKI (4 tanks / beam)	2.10^{-8}	1.10^{-8}	1 gauge	1 bad /1
MKB (5 tanks / beam)	2.10^{-5}	1.10^{-5}	1 gauge 2 pumps	2 bad /3

Their control systems receive binary alarms and analog pressure readings, from the associated Penning gauges and ion pumps.

RADIATION ENVIRONMENT

Service Areas

The majority of LHC vacuum controls are located in underground service areas, shielded from tunnel radiation. During the next 20 years, as the energy and intensity of the beam will increase, those service areas will see an increase of the High Energy Hadron (HEH) fluence rate by one order of magnitude. The electronics installed in those areas may suffer Single Event Effects (SEE).

Solutions, such as additional shielding, relocation or redundancy of critical vacuum controls must be taken into consideration. For example, due to the level of radiation caused by the collimation system at point 7, all vacuum controls equipment in UJ76 has been moved into the TZ76 gallery, during the LS1. This concerned 27 racks; the associated 280 cables were extended by up to 180 m. In this gallery, the HEH fluence rate will be much lower and the SEEs negligible ($< 10^6$ $cm^{-2}y^{-1}$) [4].

LHC Tunnel

All the gauges in the LHC tunnel are subject to ionizing radiation. Depending on their location, the levels of radiation may vary: in the ARCs, the Total Ionizing Dose (TID) will be less than 10 Gy/y; but in the LSS it will be much higher than that. The lifetime of the gauges must be better than 10 years.

Gauges in low radiation zones are “active”: the signal-conditioning electronics is incorporated in the gauge head. They are locally powered and send their readings on 0-10V through standard multi twisted-pair cables, over up to 1000 m, to the PLCs in the service areas.

The gauges installed in higher radiation zones are “passive”, with the electronics located in a service area, up to 500 m away. Power and signal are carried by special HV cables, according to the gauge type.

The cables in the tunnel accumulate radiation over the time, which progressively degrades their characteristics, eventually destroying them. Also, transient effects might be observed due to radiation-induced conductivity of the insulation of HV cables. Upcoming tests at CHARM facility (CERN) will give more information on radiation-induced effects on HV cables.

Irradiation Tests on Active Gauges

Priority has been given to test the gauges and electronics to be installed in the tunnel. This concerns the Membrane, Pirani, and Penning gauges, of which the first two are used to generate alarms for the cryogenic system, and are thus more critical for the machine operation.

Irradiation tests have been carried out in 2002 in the TCC2 facility at CERN, on a set of commercial Pirani/Penning active gauges. This facility provided a mixed field similar to that of the LHC.

The Pirani electronics provided correct readings up to 20 Gy, and saturated under-range at 50 Gy. For the Penning electronics, correct readings were obtained up to 40 Gy, saturating under-range at 60 Gy. Therefore, it was decided to use the commercial Penning electronics and to design a radiation tolerant Pirani electronics.

A set of commercial active Membrane gauges has been irradiated at the cyclotron of the University of Louvain (Belgium), in 2003. The source was a 60 MeV proton beam, with a flux of 8.10^7 $cm^{-2}s^{-1}$. The best gauge stood up to 200 Gy, with a good stability.

In 2015, a new irradiation test has been carried out at Fraunhofer Institute. A gamma radiation source of ^{60}Co has been used for 20 days to produce up to 500 Gy, which is higher than 10 years of operations in the arcs. Three samples of commercial Penning and CERN-designed Pirani electronics have been irradiated. One sample of each was placed outside the radioactive field as a reference while vacuum was simulated by resistors.

The worst Penning electronics sample was measuring pressure one decade too high already at 15 Gy, and two decades too high at 20 Gy, before falling quickly under-range. The Pirani electronics re-designed at CERN in 2002 stood up 500 Gy, with slight variations around the reference measurement.

The analysis of these tests are still ongoing and the results will contribute to define the strategy against radiation effects to electronics for future consolidations in the LHC.

SIGNAL INTEGRITY

Vacuum Measurement in the ARCs

The readout electronics for the Penning, Pirani and Membrane gauges in the ARCs are installed in a box placed above some of the dipoles. In one ARC, 28 boxes are installed. They are connected, through multi twisted-pair cables, to one of the two ARC PLCs, installed in service areas.

The Pirani and membrane electronics are locally powered, while the Penning electronics are powered from the PLC digital outputs. The 0-10V analog output of each gauge is routed through a twisted pair directly to the PLC analog input module. Moreover, each PLC analog input module provides an analog ground, isolated from the earth, used mainly by the Penning electronics.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The Penning and Pirani gauges located by the middle of the ARCs, where the cable lengths may reach 1 000 m, often shown malfunctions. Some Penning gauges could not be switched on for measurement. Moreover, by switching on or off a Penning gauge, the nearby Pirani and Penning gauges could be disturbed.

Corrective Actions

During the LS1, an investigation took place. Measurements on the output signal of the concerned Penning gauges showed a strong common mode voltage, unacceptable by the analog input module specifications.

The wire resistance on the shared powering return path was too high; the current consumption of the Penning signal-conditioning electronics brought up a voltage across the wire. As the reference for the Penning electronics is based on the common power lead, isolated from earth, this common-mode voltage was also present at the input of the analog input module of the PLC.

Moreover, the cause of measurement cross-talk between gauges installed at different places was explained by the fact that they shared the same analog ground.

An equivalent circuit has been simulated in PSpice, to understand the behavior of the interconnected analog grounds. The simulations indicated that the analog ground connections between two separated gauge electronics generated a current flowing from one to the other, depending of the status of each gauge. This created an additional DC offset to the measurement, of the order of some mV. In the range of measurement, this might correspond to one order of magnitude in pressure.

The solution finally implemented consisted in adding parallel wires, to decrease the resistance on the return path for the Penning supply. Thus, the common mode voltage has been successfully reduced within the manufacturer specifications and solved the first issue.

On the other hand, the analog ground connections have been removed, to avoid complex ground loops between different channels.

LHC COMMISSIONING

The commissioning of all vacuum measurement, alarm and interlock channels was carried out during the last months of the LS1. Each element of the chain has been tested with and without simulators. The vacuum sector valves were actuated and their status checked.

The signal transmission from the vacuum control system to the BIS has been checked in all points of the LHC. The interlocks and alarms have been triggered and checked for each system (vacuum sector valves, BIC, cryogenics, RF cavities, ADT, MKI and MKB).

The equipment involved in vacuum interlocks and alarms amounts to: 300 sector valves, 230 pairs of gauges, 590 Penning gauges, 350 ion pumps [5].

Machine Checkout

The systematic verification of interlocks and alarms has been performed during the LHC machine checkout phase [6]. The goal was to check all the interlock sources for the vacuum valves, and confirm that their status were sent correctly to the BIS, for dumping the beam.

An automated test sequence was written in the vacuum SCADA, and its results stored in xml files.

After all the vacuum valves have been opened to set the vacuum interlock to true in the corresponding BIS, each of the N (3 or 4) devices are switched off (and back on) one by one, to see if the valve open-enable is lost (and recovered).

Then, N-1 devices (3 or 2) are switched off to see if the corresponding valve closing-interlock is triggered. The time stamps are stored for: the valve closing-interlock; the loss of valve open-status; the reaching of the valve close-status; the vacuum user permit event, when the BIS receives the interlock from the vacuum. For each valve, the right sequence of all time-stamps is verified, as well as the valve travelling time, from opened to closed.

The test checks also if the corresponding alarms for Penning gauges, ion pumps, and sector valves are present in the LHC Alarm Service (LASER).

CONCLUSIONS

The vacuum measurement chains, alarms and interlocks must be reliable and are crucial to insure the good operation of the LHC. The effects of ionizing radiation and the issues with signal integrity must be understood and minimized, to improve the system reliability. Attention must be paid to the calibration, settings and tests of every measurement chain.

During the LS1, important actions have been performed in this direction: racks relocation against radiation effects; grounding and cabling modifications for signal integrity improvements; systematic test of each interlock and alarm, from front-end hardware to the SCADA. Additionally, some cabling and Database errors were found and corrected. The vacuum interlocks are part of the LHC machine protection; during machine checkout, they were fully verified up to the BIS.

REFERENCES

[1] P. Gomes et al., "The Control System of CERN Accelerators Vacuum [Current Status & Recent Improvements]", ICALEPCS11, Grenoble, Oct 2011.
[2] MPS Aspects of the Vacuum System Commissioning, CERN, EDMS 896391.
[3] User Interface to the Beam Interlock System, CERN, EDMS 636589.
[4] G. Pigny et al, "Mitigation of Radiation and EMI Effects on the Vacuum Control System of the LHC", TWEPP13, Perugia, Sep. 2013.
[5] Test Procedures for the LHC's Vacuum Control System Commissioning, CERN, EDMS 1405440.
[6] Test Procedures for Functionality Checks of the Vacuum Valves and the BIC during Machine Checkout, CERN, EDMS 1010244.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONTROLS AND INTERLOCKS FOR THE NEW ELETTRA SUPER CONDUCTING WIGGLER

L. Pivetta, F. Giacuzzo, G. Scalamera, D. Vittor,
Elettra Sincrotrone Trieste, Trieste, Italy

Abstract

During the last two years, triggered by the construction of the XRD2 beamline, and to comply with the topup operations, a complete refurbishment of the Elettra Super Conducting Wiggler (SCW) has been carried out. Alongside with the mechanical, cryogenic and electrical components, also the electronics, the control and interlock systems have been upgraded. The MVME5110 PowerPC single board computer, which is a standard in the Elettra control system, has been adopted, as well as RS232 communication modules, analog to digital converters and digital I/O lines. To cope with the high output power of the SCW, up to 18 KW, the interlock system, protecting both the wiggler and the beamline frontend, has been completely redesigned. The control system software has been rewritten from scratch using the TANGO software framework. The complete system has been tested during the second half of 2014 and is now fully operational.

INTRODUCTION

A 3.5 T superconducting wiggler, designed and built by the Budker Institute of Nuclear Physics (BINP), has been installed in the Elettra storage ring as a photon source for the X-ray diffraction beamline XRD2 in 2002 [1, 2]. Due to the lack of funding for the beamline construction, the SCW didn't came in operation in the subsequent years. Then, in 2011, a collaboration agreement between the Indian government and Elettra Sincrotrone Trieste provided the funds required for the beamline construction and the SCW refurbishment. During 2012 and the first months of 2013, at the BINP laboratory in Novosibirsk, the SCW has been provided with a new cryostat aimed at decreasing the helium consumption and increasing the reliability. Then, in July 2013, after successful site acceptance tests, the SCW has been installed in the Elettra storage ring [3]. A picture of the installed wiggler is shown in Fig. 1.

Simplifying, the main components of the SCW are the chassis, the cryostat, the liner and the superconducting coils; the principle of operation of the SCW consists in keeping the cryostat temperature near to the 4.5 K, using liquid helium, and feeding the superconducting coils with the appropriate currents in order to reach the desired magnetic field.

ELECTRONICS

The original Danfysik MPS883 power supplies, powering the central and the side superconducting coils, after being revised have been kept in operation, as well as the four existing Leybold Coolpack 6000 compressors, provided with new Coolpower cooling heads. A new junction box, shown in Fig. 2, has been designed by BINP to replace both the old cabled box and the VME-based signal conditioning and acquisition board. The new junction box, based on an Atmel ATmega128 microprocessor, acquires the signals coming from the temperature and pressure sensors installed into the cryostat and implements the quench interlock logics. If a quench occurs the junction box turns off the power supplies that feed the superconducting coils driving their external interlock input. A RS232 serial line is available for remote control and supervision.

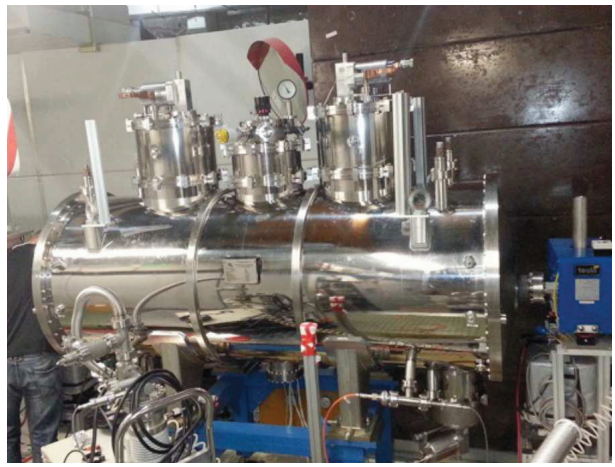


Figure 1: The refurbished super conducting wiggler installed in the Elettra storage ring.

Figure 2: The new Junction Box.

Ion pumps, powered by the DUAL high voltage power supplies manufactured by Varian, are also installed to provide the requested vacuum levels. A pirani vacuum gauge allows to monitor the pressure in the vessel insulation, whilst some penning gauges are used in the front-end vacuum chamber; they are all powered with two TPG300 power supplies.

CONTROLS

In conjunction with the refurbishment of the cryostat the SCW control system has also been updated. The legacy control system, based on a VME crate and a Motorola 68K microprocessor single board computer running VxWorks, provided by BINP as a turnkey system, has been dismantled. Also, the ELTEC BAB40 single board computer, run-

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ning Microware OS-9, which acted as gateway to the legacy RPC based control system has been removed.

Hardware

A new dedicated front-end computer, based on the Artesyn MVME5110 VME single board computer, which is a standard for the Elettra control system, has been installed in a VME64x crate. A number of RS232 serial lines are requested to interface the two Danfysik MPS883 power supplies, the four compressors, the junction box and the vacuum power supplies. Two TIP866 industry pack mezzanine cards, each providing 8 RS232 serial lines, are installed on a TVME220 VME carrier board. An in-house developed signal conditioning board, compliant to the VME64x rear I/O board specification, provides the opto-insulation on the serial lines. To acquire some additional analog signals, such as the current monitor of the MPS883, a TIP501 ADC industry pack mezzanine card is used. Also, a new peripheral of the machine interlock system has been installed to interface signals mandatory for the machine protection.

Software

All the control software has been moved to the TANGO Controls framework [4]. More in detail, one TANGO device server for each family of devices to be interfaced has been developed:

- Danfysik MPS883 power supplies;
- Leybold Coolpack 6000 compressors powering cryocooler models 4.2 GM and 10 MD;
- junction box.

The TANGO device servers for the DUAL and TPG300 high voltage power supplies of the FERMI vacuum subsystem have been reused.

In charge of the overall SCW supervision, one more TANGO device server, that interacts with the junction box device server and the power supply device servers, has been developed. Dedicated additional threads are in charge of:

- ramp the magnetic field;
- monitor the junction box status and:
  - ramp the SCW magnetic field to zero in case of helium level warning or temperature warning;
  - stop any magnetic field ramp whenever a quench, or other alarm, are detected;
- monitor the power supplies and compute the actual magnetic field from the currents; this thread is also in charge of pushing the reading events to the clients.

In order to follow a precise trajectory, a look-up table is used to compute the power supplies currents corresponding to the requested magnetic field, using linear interpolation. The look-up table also specifies the speed of the current ramps, in A/s, for a number of predefined magnetic field ranges: different intensities require different speeds. The power supply currents versus the magnetic field are plotted in Fig. 3.

Moreover, even when ramping up or down the field different speeds have to be used. The thread driving the ramps also checks that the actual magnetic field value moves along the predefined trajectory; if the tolerance is exceeded the current ramps are stopped and a new one started to move to an allowed value on the calibrated trajectory, following the shortest path.

The status of the MPS883 power supplies is also monitored and, should a fault occurr, the magnetic field ramp

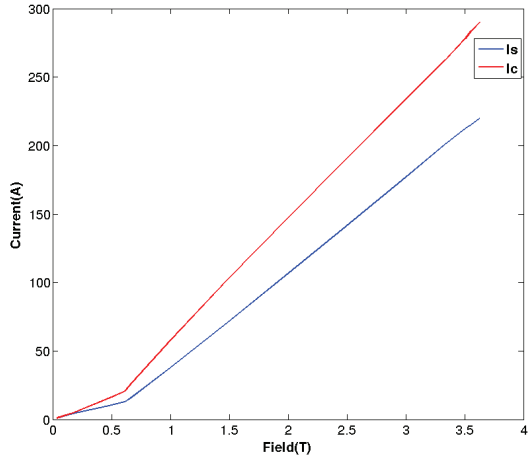


Figure 3: Side coil [blue] and central coil [red] power supplies currents versus magnetic field.

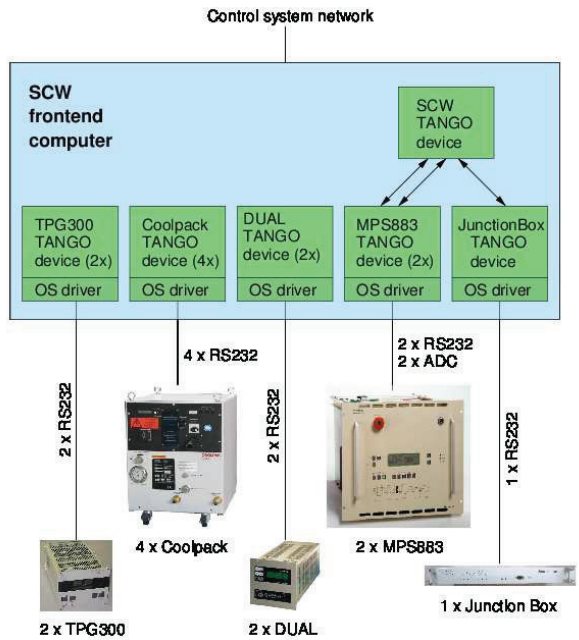


Figure 4: TANGO device servers layout on SCW front-end computer.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

stopped. The structure of the control software deployed on the SCW front-end computer is shown in Fig. 4.

All the relevant parameters of the SCW are continuously monitored and stored into the historical database of the accelerator, using the HDB++ archiving system, on both variation and periodic basis. A large number of values are stored during the magnetic field ramps, in fact at each setting of the power supplies currents; when the wiggler is not ramping just a few values are stored on a periodic basis.

Graphical User Interface

A graphical user interface, to be used by the control room operators and the insertion devices specialists, has been developed with Qtango and the Qt graphics library. All the parameters and the readings, such as temperatures, power supplies status and currents, helium level, junction box and compressors status and, last but not least, wiggler magnetic field are shown over a synoptic image of the SCW. In addition the pressure levels and the temperatures of the front-end are also displayed. Moreover, the operator can input the desired magnetic field value and start the ramp. A screenshot of the current SCW GUI is shown in Fig. 5.

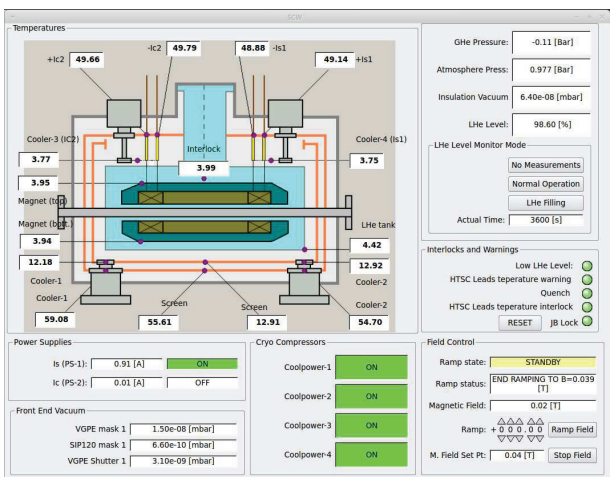


Figure 5: The SCW graphical user interface.

INTERLOCKS

The high output power of the 3.5 T superconducting wiggler introduced a novel issue on the accelerator light exit as well as the beamline front-end.

The machine protection interlock system of the Elettra storage ring is based on Siemens S7 programmable logic controllers (PLC). The interlock PLC, though operating independently, is connected to the control system via Ethernet and a dedicated TANGO device server provides all the process variables to the supervisory system, which is also based on TANGO.

A vacuum valve, protected by a beam shutter, usually separates the Elettra storage ring vacuum chamber from the beamline front-end vacuum. In case of increasing pressure on the beamline, the interlock PLC, in protection of the storage ring vacuum, closes the valve and, to protect the valve from the beam, the corresponding shutter. However, the standard beam shutter used at Elettra can not cope with the full SCW power. On the other hand, a 18 KW capable shutter is large and expensive, and an additional one is required for the beamline photon shutter. Therefore, the decision has been taken to install a new photon shutter, capable of holding 18 KW photon beam power and to implement a mechanism to protect the existing exit valve/shutter. Consequently, the storage ring interlock system for the XRD2 beamline has been completely redesigned to include the vacuum, the valve/shutter and the front-end protection. A new peripheral of the interlock system, connected via PROFIBUS to the nearest S7 CPU, acquires the SCW power supply currents by means of two dedicated DCCT converters, as well as a number of thermocouples, flowmeters and vacuum gauges, installed on the front-end. To define the operating status of the wiggler two thresholds have been introduced:

- beam shutter protection threshold;
- fast orbit interlock threshold.

The first one, usually set at 1.0 T, enables the interlock protection of the beam shutter/vacuum valve: if the valve is closed, hence the shutter, and the SCW magnetic field is higher than the threshold the electron beam is dumped. The second, usually set at 0.42 T, tells the fast orbit interlock system that the SCW is operating and the tolerance on permissible orbit distortion has to be lowered.

A large displacement of the superconducting coils power supplies currents from the reference trajectory values, possibly due to control software or hardware failure, can lead to heavy orbit distortion and, considered the big output power, to the damage of the vacuum chamber. The PLC monitors the actual power supply currents by means of two DCCTs; the values are compared to the theoretic values, computed from the current magnetic field by a reverse calculation, and, if the tolerance is exceeded, the electron beam is dumped.

The status of all the interlocks related to the SCW operation is summarized on a dedicated GUI, shown in Fig. 6.

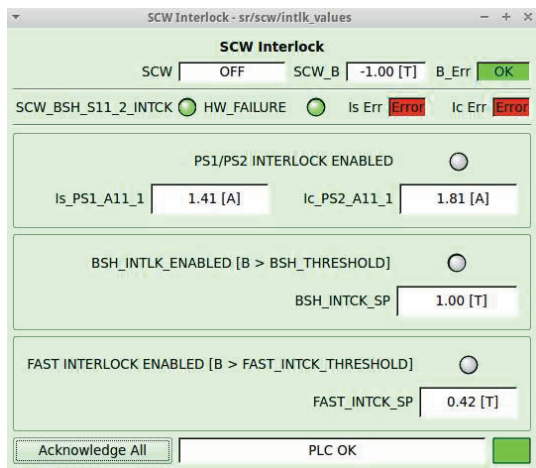


Figure 6: The SCW interlock GUI.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Thermocouples, flow meters and vacuum gauges installed on the front-end are also monitored and the appropriate interlock logics implemented. Currently, in case of low flow the electron beam is immediately dumped. Conversely, a two step mechanism is in place for the front-end temperatures and pressures. First, some alarms have been set up in the TANGO alarm system to alert the operator of possible problems with the temperatures or the vacuum. In case of increasing front-end temperatures, for instance, the operator can ramp down the SCW magnetic field decreasing the output power. Should the operator miss the action, the second step belongs to the interlock PLC that dumps the electron beam.

CONCLUSION

The control system for the new SCW is in operation since autumn 2014. Also the interlock system, updated in two steps, is now fully operational. A number of alarms are already in place to help the machine operator to supervise and operate the SCW at the best condition and, because of the need of electron beam dump in plight, to prevent possible issues.

REFERENCES

[1] A. Batrakov et al., "A Superconducting 3.5 T Multipole Wiggler for the ELETTRA Storage Ring", EPAC2002, Paris, France (2002).

[2] L. Tosi et al., "The ELETTRA Superconducting Wiggler", EPAC2004, Lucerne, Switzerland (2004).

[3] D. Zangrando et al., "The Elettra 3.5 T superconducting wiggler refurbishment", IPAC2014, Dresden, Germany (2014).

[4] TANGO Controls website: `http://www.tango-controls.org`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONTROLS INTERFACE INTO THE LOW-LEVEL RF SYSTEM IN THE ARIEL e-LINAC AT TRIUMF

J.J. Pon, K. Ezawa, R. Keitel, R.B. Nussbaumer, J.E. Richards, M.A. Rowe, P.J. Yogendran
TRIUMF, Vancouver, BC, Canada

Abstract

Phase I of TRIUMF Advanced Rare Isotope Laboratory (ARIEL) was completed in September 2014. At phase I, the Low-Level RF (LLRF) system of ARIEL's electron linear accelerator (e-Linac) consists of a buncher and a deflector, one single-cavity injector cryomodule and the first cavity of two dual-cavity accelerating cryomodules. The model for the e-Linac LLRF system is largely based on the experience gained from the fully-commissioned TRIUMF ISAC-II linear accelerator (linac). Similarly, the EPICS-based Controls for the e-Linac LLRF builds on the lessons learned with the linac LLRF Controls. This paper describes the interface between the ARIEL Control System (ACS) and the e-Linac LLRF using EPICS ASYN/StreamDevice and a SCPI-like protocol. Also discussed are the ACS EDM displays and future plans for LLRF Controls.

INTRODUCTION

The Advanced Rare Isotope Laboratory (ARIEL) is the latest project in the Rare Isotope Beam (RIB) program at TRIUMF. In conjunction with ISAC and ISAC-II, ARIEL is designed to triple RIB production and expand the range of exotic isotopes “for Nuclear Physics and Astrophysics, Nuclear Medicine and Materials Science” [1].

In September 2014, phase I of ARIEL was completed with the delivery of a 300KeV DC thermionic gun, a buncher and a deflector, a single-cavity injector cryomodule (ICM), and the first cavity of a dual-cavity accelerating cryomodule (ACM). These components constitute the first half of ARIEL's superconducting electron linear accelerator (e-Linac). The second half of the e-Linac is presently under way and comprises the second cavity of the first ACM, and a second dual-cavity ACM, for a total of five cavities in three superconducting cryomodules.

The design of the Low-Level RF (LLRF) system for ARIEL's e-Linac borrowed extensively from the model used in ISAC-II superconducting linear accelerator (linac) [2] – TRIUMF's first fully-commissioned linac. It was natural then, that the new ARIEL Control System (ACS) interface into the e-Linac LLRF also followed the model used in ISAC-II linac Controls. However, as in any new project, LLRF Controls of ARIEL's e-Linac presented an opportunity to improve on some of the unexpected drawbacks encountered in ISAC-II Controls.

INTERFACE

LLRF-ACS Interface

The overall framework used in ISAC-II LLRF Controls has been carried over into the LLRF-ACS interface. At one end, the LLRF system exchanges information with Controls via an EPICS IOC. At the other end, the IOC makes LLRF data available to EPICS clients in the form of process variables (PV). While this model has worked well at TRIUMF, the way it was implemented in ISAC-II LLRF Controls uncovered some drawbacks that were addressed in the LLRF-ACS interface.

In ISAC-II, the LLRF system interfaced with the Controls EPICS IOC using a shared memory model [3]. The implementation of this method brought in a number of firm constraints. First, inter-process communication via shared memory implies a high degree of coupling. The LLRF system and ISAC-II Controls had to adhere to a strict memory mapping scheme, was not very portable, and carried a moderate amount of complexity to maintain.

Another implication on the use of shared memory is that the IOC needs to run in the same LLRF system computer. The host machine is owned and maintained by the LLRF group. For simplicity and security reasons, LLRF computers are stand-alone machines and are not connected to the site's network. If Controls wanted to perform diagnosis or maintenance of the IOC it was necessary to inconvenience the LLRF group for physical access to the host computer. Even simple actions like viewing IOC log messages or machine crash dumps required local access to the machine.

Furthermore, the IOC had to be built to run on Microsoft Windows. But Controls have had limited experience developing for the Windows environment. Only one Controls programmer gained enough knowledge to build a heavily-customized Windows IOC based on the EPICS Portable Channel Access Server framework and make available to LLRF engineers a C++ API to read/write the shared memory. Plus, the programming was done in Visual Studio and involved the use of WIN32 DLLs [4], these development tools were also unfamiliar to Controls.

During the early stages of ARIEL phase I an alternative implementation for the LLRF-ACS interface was put forward. After some consultation with the relevant groups one of the main conclusions was to not proceed with the shared memory model. Instead, the ACS would use the EPICS ASYN/StreamDevice method of communication with LLRF. The Controls group is familiar with

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ASYN/StreamDevice. It has been used in previous occasions to interface with various kinds of peripheral equipment at TRIUMF so the knowledge and experience is well spread among Controls personnel.

In addition, the decision to not use shared memory gave Controls more freedom to choose where to run the EPICS IOC. Whereas in ISAC-II the Controls IOC was required to be hosted in the LLRF system computer, in ARIEL the IOC moved into a computer in the ACS domain. This setup gives Controls personnel freer access, without unnecessary inconvenience to LLRF staff, to peruse the logs and system messages for diagnosis, maintenance, and updates. Not to mention that there is a clearer separation of responsibilities. For example, in ISAC-II, the LLRF group was responsible for updating the Controls IOC database. Not so in ARIEL where the IOC database is out of the hands of LLRF and into ACS.

Another benefit of hosting on an ACS computer is that the EPICS IOC runs on a Linux machine. Controls have ample expertise on Linux - the standard operating system in the group [5]. And Controls is also very familiar with building, maintaining, and debugging EPICS IOCs on Linux. The additional knowledge in Windows shared memory, Visual Studio, or WIN32 DLLs is no longer required.

For its part, the ARIEL LLRF group committed to building a server application to respond to commands and queries from the ACS over a network connection. To preserve the security of LLRF computers, the ACS connection is done over a private network, shielded from the outside world. The LLRF system and ACS agreed to follow the guidelines for Standard Commands for Programmable Instruments (SCPI) for the structure of commands and queries [6]. However, our model does not fully adhere to the complete SCPI (pronounced “skippy”) specification. The intention was not to degrade the standard but to make it as simple as possible to meet our needs at the time. For example, the SCPI specification for command concatenation was deemed unnecessary and thus, not implemented. Similarly, command abbreviation was also not implemented. The set of allowed SCPI-like commands and queries was documented in a table on an internal TRIUMF web page [7] to facilitate separate implementations in both groups. Figure 1 shows the various protocols used and the separation of responsibilities in the model for the LLRF-ACS interface.

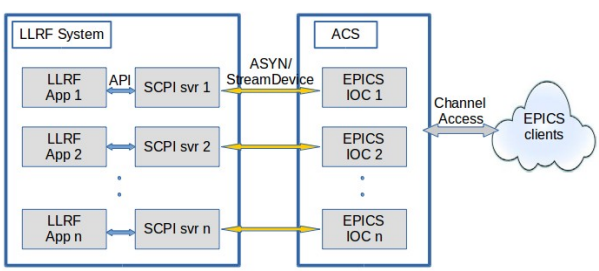


Figure 1: LLRF-ACS Interface.

Controls GUI

The ACS graphical displays did not see any major changes from those used in ISAC-I and ISAC-II Controls. In fact, the idea was to borrow or re-use as much as possible from earlier projects [5]. Development of the displays was done using the EPICS Extensible Display Manager (EDM). Colour schemes, size of objects, fonts, etc. followed the same established site guidelines. To preserve the same look-and-feel of ISAC-I and ISAC-II displays the placement of buttons, sliders, labels, etc. was also emulated in the ACS. Figure 2 shows a screenshot of the display for the e-Linac ICM LLRF Controls. Figure 3 shows the expert display, which is used mainly by the LLRF group for more detailed diagnostics.

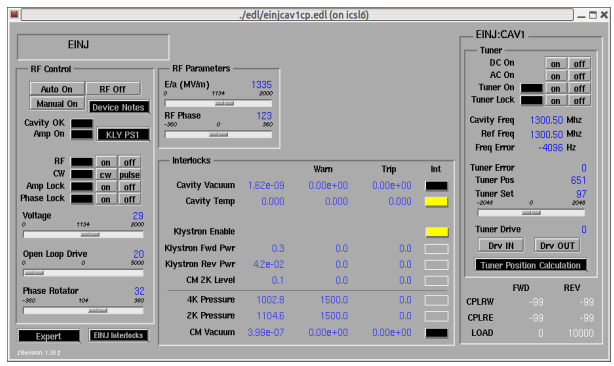


Figure 2: ACS Display of the ICM LLRF.

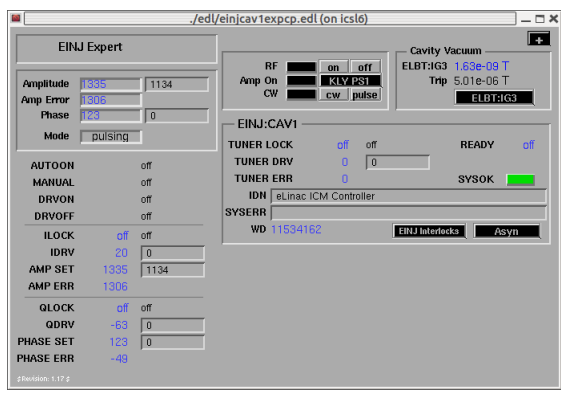


Figure 3: ACS Expert Display of the ICM LLRF.

One notable feature is the addition of a delay on fast, successive commands. In ISAC-II, there were cases when operating on an EDM slider object made the LLRF system unresponsive to remote commands and only a reboot of the LLRF computer would fix it. So far, this issue has not been resolved but we speculate that the fast storm of setpoint commands from the slider overwhelmed the LLRF system and sent it into a funny state. In the ACS, the IOC communicates changes in setpoint values to the LLRF SCPI server at 500ms intervals. Changes in setpoints in between this period are discarded. In other words, the IOC throttles the number of commands to

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

change setpoints to avoid overwhelming the SCPI server. Granted, this feature is implemented at the IOC level but it directly addresses a potential issue with the setpoint objects of the ACS displays.

FUTURE PLANS

After the implementation of the LLRF-ACS interface was completed in ARIEL phase I, the Operations group anticipated that the improvements would be backfitted to earlier projects. As of this writing, a retrofit of ISAC-II LLRF Controls is being commissioned. The changes have been transparent to Operations because the bulk of the modifications is at the IOC-LLRF interface level, whereas EPICS displays remain largely unchanged. It is hoped that issues like the slider control problem will not re-appear. There are also plans to apply these improvements to ISAC-I LLRF Controls.

It is also worth noting that the shared memory model was applied, to a minor degree, in specific sub-systems in the Laser Control System and Beam Diagnostics System [3]. There are plans to retrofit these sub-systems too but due to constraints in time and resources this may not occur in the immediate future.

SUMMARY

The model for ARIEL's LLRF-ACS interface is heavily based on the design of ISAC-II LLRF Controls. Although the overall framework is maintained, there are notable improvements from an implementation point of view. The decision to replace the shared memory model with ASYN/StreamDevice interface reduced the number of restrictive implementation constraints and allowed for a clearer division of responsibilities. As for the ACS GUI displays, the same guidelines and same look-and-feel of earlier projects were closely followed. ISAC-II operators would be hard pressed to notice much difference in the ACS displays from what they are used to seeing.

In conclusion, the improvements to the LLRF-ACS interface described in this paper have been well received and so far there are no major issues to contend. In fact, there are plans to apply these changes to existing Controls-related projects where the shared memory model is still in use.

ACKNOWLEDGEMENT

The authors would like to acknowledge the contribution of time and expertise from the SRF/RF group: Ken Fong, Mike Laverty for unraveling the requirements for LLRF Controls. Philipp Kolb, Zhongyuan Yao, Qiwen Zheng for spending endless hours testing the Controls interface.

Also, many thanks to all members of the Controls group for their support and guidance. In particular, Shawn Rapaz for helping setup the Controls hardware and Evgeniy Tikhomolov for sharing his experience of ISAC-II LLRF Controls development.

REFERENCES

[1] L. Merminga, et al., “ARIEL: TRIUMF's Advanced Rare IsotopE Laboratory”, IPAC2011, San Sebastian, Spain (2011), WEOBA01; www.JACoW.org

[2] M. Laverty, K. Fong, Q. Zheng, “TRIUMF e-Linac RF Control System Design”, PAC09, Vancouver, Canada (2009), WE5PFP099; www.JACoW.org

[3] E. Tikhomolov, “Interfacing of Peripheral Systems to EPICS Using Shared Memory”, ICALEPCS07, Knoxville, USA (2007), TTPA29; www.JACoW.org

[4] E. Tikhomolov, G. Waters, R. Keitel, “EPICS Portable Channel Access Server for Multiple Windows Applications”, ICALEPCS03, Gyeongu, South Korea (2003), WP574; www.JACoW.org

[5] R. Nussbaumer, et al., “ARIEL Control System at TRIUMF – Project Update”, ICALEPCS13, San Francisco, USA (2013), MOPPC094; www.JACoW.org

[6] J.J. Pon, K. Fong, R. Nussbaumer. “Specifications for the Interface Between Low-Level RF System and ARIEL Control System”, Internal TRIUMF Document 108110, 2014/03/20

[7] https://www.triumf.info/wiki/internal/isaccontrols/index.php/LLRF-ACS_SCPI_Commands

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

LABVIEW AS A NEW SUPERVISION SOLUTION FOR INDUSTRIAL CONTROL SYSTEMS

O.O Andreassen, F. Augrandjean, E. Blanco Vinuela, D. Abalo Miron, M. F. Gomez De La Cruz, A. Rijllart, CERN, Geneva, Switzerland

Abstract

To shorten the development time of industrial control applications, CERN has developed the Unified Industrial Control System (UNICOS) framework, which standardizes the programming of the front-ends and the configuration of the Supervisory Control and Data Acquisition (SCADA) layer. At CERN the SCADA system of choice is WinCC OA, but for some specific projects (small or initial prototypes not connected to accelerator operation or not located at CERN) a more customisable supervision using LabVIEW is an attractive alternative. Therefore, a system called UNICOS in LabVIEW (UiL), has been implemented. It provides a set of highly customisable re-usable components, devices and utilities. Because LabVIEW uses different programming methods than WinCC OA, the tools for automatic instantiation of devices on the supervision layer had to be re-developed, but the configuration files of the devices can be reused. This paper reports how the implementation was done, it describes the first project implemented in UiL and an outlook to other possible applications.

INTRODUCTION

The CERN Unified Industrial Control System (UNICOS) project proposes a standardized architecture for both, supervision and control layers. It also defines a set of tools for automatic instantiation of devices, in the two mentioned layers, employing re-usable components, devices and utilities.

However, for smaller lab-size solutions e.g. systems which are only expected to exist for a short time or when users do not have the time and resources to invest in a complete SCADA as WinCC Open Architecture (WinCC OA), a LabVIEW based solution would fill this gap [1][2].

UNICOS in LabVIEW (UiL) gives the user much of the same look & feel and functionality of WinCC OA as Human Machine Interface (HMI).

UiL generates automatically the supervision objects from the same configuration file provided by UNICOS Application Builder (UAB), which was developed to generate the code and configuration files of the different components of the UNICOS project (e.g. WinCC OA, Siemens PLCs etc.) [3][4]. These objects are the counterpart instances of the control objects deployed in the PLCs in the control layer.

Once the objects are instantiated in the UiL application, they can be used in the application synoptic by means of widgets, a reduced but still meaningful vision of its status, and faceplates, a full view of the object status and the place where the operator will interact with the control system.

In addition to the widgets and faceplates, UiL provides a trending tool, an alarm and an event list. All the components, widgets and user interaction in UiL are designed to resemble the UNICOS look and feel as much as possible; making sure that the transition from one SCADA tool to the other is effortless.

UNICOS

UNICOS was initially developed in 1998 as a need to develop the LHC cryogenics control system, but became a CERN de facto standard for industrial control systems. As shown in the figure 1, the framework connects, instantiates and interacts with both the supervision and control layers with commercial industrial components [1][5].

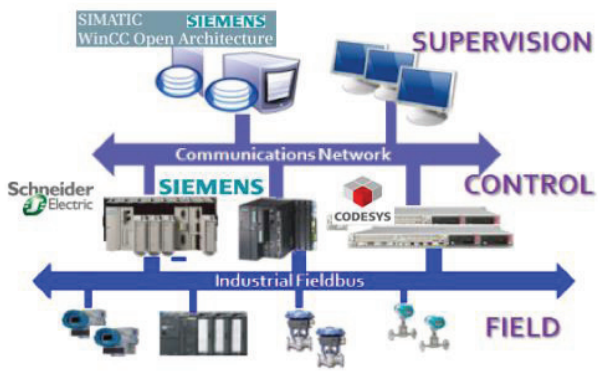


Figure 1: UNICOS architecture.

Currently UNICOS uses WinCC OA as a commercial SCADA system and both Siemens, Schneider, as Programmable Logic Controllers (PLC) [1][6]. A third possibility using Codesys extends the usage to industrial computers (IPC)

The possibilities and the application of the UNICOS framework extend to monitoring and supervision applications (e.g. Magnets alignment, Collimators, Quench protection systems...) where the control layer could be not based on UNICOS.

The UNICOS framework is based on an object oriented concept consisting of standard object classes featuring process control objects, controllers, valves, pumps, heaters, inputs, outputs etc. It also establishes how these objects interact and formalizes the control unit definition together with the way the specific control logic is implemented.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

UNICOS IN LABVIEW

Although deploying a UNICOS/WinCC OA SCADA solution is easier than starting from scratch with WinCC OA, for some applications this solution can still be too difficult, and/or too costly in terms of needed expertise knowledge. In some cases, the developers lack the time, knowledge, manpower and/or resources to invest in a complete SCADA as WinCC OA. Many times the developers are already familiar with LabVIEW as well. These are the basic justifications for the UNICOS in LabVIEW (UiL) project [1][7].

The objective of the UiL project is to offer a simple and affordable solution including most of the core functionality and tools found in UNICOS. The combination of a simplified UNICOS functionality and LabVIEW's graphical programming interface, makes it more suited for lab-size and prototype applications with a substantial reduction on engineering costs (Figure 2). In addition, the site-licensing scheme adopted at CERN for LabVIEW reduces the cost of ownership for smaller SCADA systems.

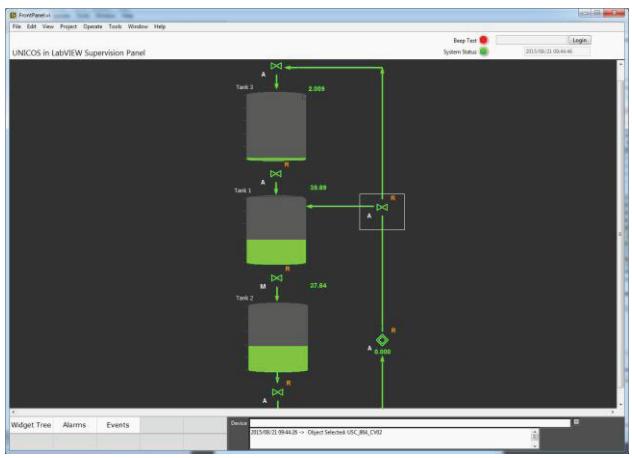


Figure 2: UiL example application

Architecture and Components

UiL is implemented using several commercial off-the-shelf components as LabVIEW, LabVIEW Datalogging and Supervisory Control Module (DSC) and a third party Python bindings application interface to MongoDB, an open-source document database designed to ease the development and scaling [8].

The general module architecture can be seen in Figure 3. The application revolves around a client, which subscribes, caches and re-serves data throughout the application using LabVIEW's internal notifiers and queues. The most resent updates are stored and broadcasted through an internal notifier, which can be read (by reference) in any sub module of the application. The larger data sets (arrays and vectors for trends and graphs) are stored in an internal queue, and indexed by object type identifiers.

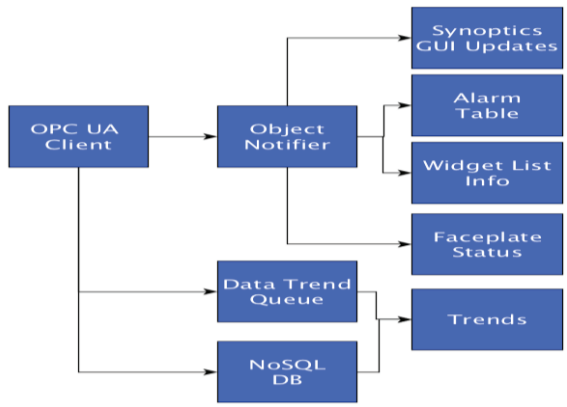


Figure 3: UiL modules.

Communication

The communication between UiL and the UNICOS based PLC's is based on OPC UA. OPC UA is a platform-independent, service-oriented architecture specification that allows the exchange of data in the industrial automation space [9]. The UNICOS based PLCs targeted are Siemens and Schneider. S7 and Modbus over Ethernet are used by those PLCs respectively. Moreover, they implement a CERN in-house protocol, the so-called Time Stamp Push Protocol (TSPP). This protocol is an event-based protocol; data is transmitted towards the supervision layer once it changes and it is sent together with the source time stamp [10].

A dedicated OPC UA server, encapsulating the TSPP protocol, has been developed at CERN. Having such server, any OPC UA client can connect to it and get the PLC data out, as it is the case of UiL, which includes an OPC UA client. This solution is very convenient as it simplifies the communication having only a single mechanism independent of the PLC (Figure 4).

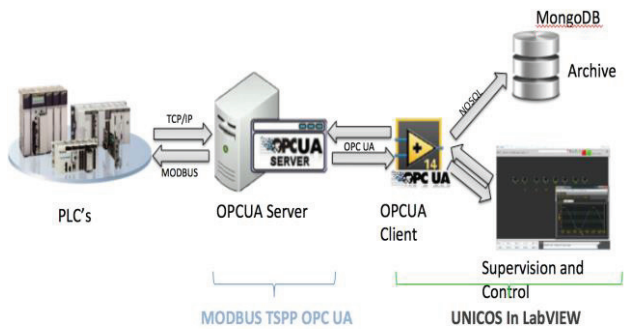


Figure 4: UiL communication.

Data Flow

Data in UiL is primarily stored and accessed through the use of internal LabVIEW notifiers. For trends, the data is queued, in order to accurately display all data from the PLC without losing data points in the graph. The

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

trends and events also access the NoSQL database for the viewing of historical data (Figure 4).

Archiving

UiL has the capability to archive all live data and settings. When the user selects this option in the UNICOS specification file for a particular device, the data is automatically cached and stored. UiL archives its data in MongoDB, a cross-platform document-oriented database. Classified as a NoSQL database. MongoDB eschews the traditional table-based relational database structure in favour of JSON-like documents with dynamic schemas, making the integration of data in applications such as UiL easier and faster [1][4]. The method used to archive data is class based and intended to support multiple sinks through class overrides in the future. This would make data retrieval possible directly from a WinCC OA archive, hence facilitating integration in between these to supervisors.

WORKFLOW AND DEVELOPMENT

The starting point for UiL configuration is the UAB generated configuration file [3][4]. This file contains the front-end, typically a PLC, parameters and the devices to instantiate. The user navigates to the file, selects a project name and location where to save the project (Figure 5). The IP address or hostname of the PLC is retrieved from the configuration file when the program runs after creation. The user can also manually set this if one wishes to connect to a simulated device.

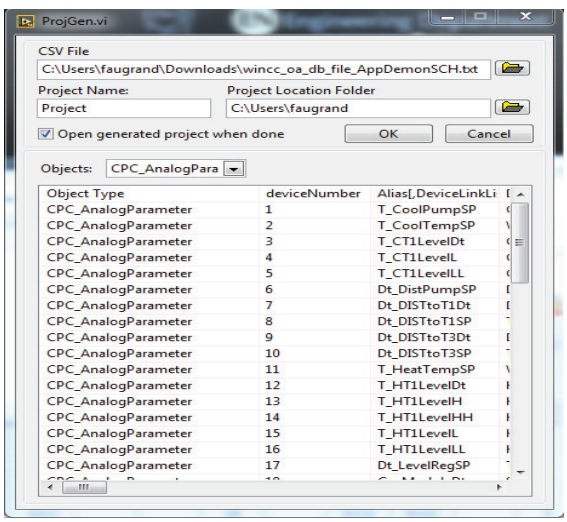


Figure 5: UiL Project generator.

Building the synoptic for a UiL application is drag and drop based. In the LabVIEW Project window, the 'Widgets' folder contains all object types as well as other graphical indicators and decorations. During the generation of the widgets, all items have been given a type identifier, which is used internally by the UiL's scripting engine. This facilitates all animations and user interactions with the widget.

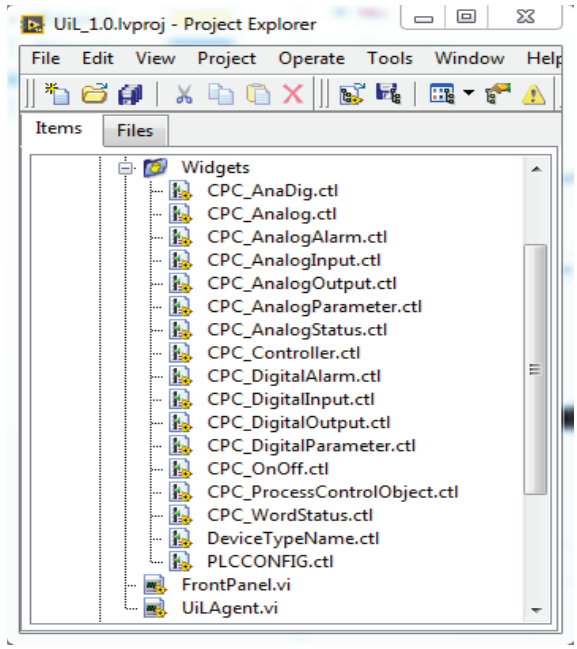


Figure 6: LabVIEW UiL project window.

To place specific devices into a synoptic, the user drag-and-drops a widget (Figure 6) onto the Front Panel which triggers a window where the user can select the object the widget must be linked to, choose an icon representation, and what device reference to connect to. Once configured, the widget is animated on the applications front panel.

VALIDATION

The UiL development has been initially tested in terms of connectivity with two OPC UA servers, interfacing with one PLC and one virtual device and up to three hundred channels. The test machine was a standard HP Compaq 8000 elite with 4Gb of RAM running Windows 7 64bit. The UiL LabVIEW instance was running in 32bit mode. The test bench had 200 active channels to the two servers and all the data was logged to its MongoDB archive. No significant increase in CPU (was at 30% when starting and stayed at 35% after the application ran) or Memory (used 200MB at start and had 350 Mb after 48H run) was noticed while running the test. The test provoked a random alarm every 30 minutes on random channels, and stored all the data while running. The update frequency was set to 1 point every half second, fixed.

The results in terms of animation were also very satisfactory having several dozens of devices in the same synoptic refreshing continuously. Interaction with the devices (via dedicated buttons) in the faceplates associated to the widgets was also smooth and efficient.

The current OPC UA implementation only communicates with one PLC, and some of the status indicators and alarms rely on live data from the OPC-UA server. This can cause confusion in case of communication loss by not having a proper diagnostic in the proposed architecture.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The current UiL release only supports a single database archive per application instance. This limits the cross application interaction although this will be addressed in next releases.

FUTURE PLANS

A consolidation phase is the first step to make the UiL a solid and deployable solution. Up to now all the concepts have been validated during the development and test phases. The plans include the development of a full palette of widgets and faceplates for the different sensors, actuators and control devices. The alarm and event managers will incorporate filtering capabilities and the diagnostics of the front-ends will be properly interpreted and shown to the user.

Support of multiple connections to PLCs will be added, but we still have to decide if this will be done at the OPC-UA server level or with the UiL client connecting to multiple servers.

Cross communication between different instances of UiL backends will be implemented in the next release.

CERN services will be included in future releases, these comprise the interface with LASER, the CERN central alarm system, the DB Logging, the CERN long term database and the CERN middleware (CMW) to exchange data from the UiL to a third party applications.

With the current class based data subscription model and in combination with the RADE framework, this could be done without much effort [7][11].

CONCLUSION

UiL has been successfully developed and is being used for prototypes at CERN. The combination of LabVIEW's intuitive drag and drop-based interaction, together with the similar look and feel of WinCC OA makes UiL a good choice for small to medium sized UNICOS applications as a cost effective supervisor. UNICOS is a widely used and supported standard for industrial control systems at CERN. Using UNICOS instead of a custom solution will benefit CERN users, as it will speed up development time and their support is ensured

UiL has most of the core UNICOS features, such as the generator, widgets, faceplates, contextual buttons, the alarm and event list, the trend tool and panel navigation tools.

OPC UA has been chosen as the communication layer between LabVIEW and the PLCs. Our tests show that both the Siemens and Schneider PLCs communicate well using the OPC UA interface and scales well with UiL [2][5]. This is accomplished without needing to do any changes to the UiL OPC UA client.

Our performance tests show that UiL can handle several hundred widgets running simultaneously without any significant load to the CPU.

REFERENCES

[1] P. Gayet et al. "UNICOS a framework to build industry like control systems", ICALEPCS 2005, Geneva, Switzerland, (2005)

[2] Siemens "SIMATIC WinCC", (1996), https://en.wikipedia.org/wiki/WinCC

[3] I. Prieto Barreiro et al. "UAB CONTINUOUS INTEGRATION FOR AUTOMATED CODE GENERATION TOOLS", ICALEPCS 2013, San Francisco, USA, (2013)

[4] I. Prieto Barreiro "UAB Bootstrap", CERN 2012, Geneva, Switzerland, (2012)

[5] R.Barillère et al. "A HOMOGENEOUS APPROACH FOR THE CONTROL OF THE LHC EXPERIMENTS GAS SYSTEMS", ICALEPCS 2003, Gyeongju, Korea, (2003)

[6] B. Farnham et al. "Migration from OPC-DA to OPC-UA", ICALEPCS 2011, Grenoble, France, (2011)

[7] O. Ø. Andreassen et al. "The LabVIEW RADE framework distributed architecture", ICALEPCS 2011, Grenoble, France, (2011)

[8] K. Banker "MONGODB IN ACTION", p. 375, ISBN 9781935182870, (2011)

[9] W. Mahnke "OPC UNIFIED ARCHITECHTURE" OPC UA, (2009), https://library.e.abb.com/public/75d70c47268d78bfc125762d00481f78/56-61 3M903_ENG72dpi.pdf

[10] J. Ortolá Vidal et al. "AN EVENT DRIVEN COMMUNICATION PROTOCOL FOR PROCESS CONTROL: PRFORMANCE EVALUATION AND REDUNDANT CAPABILITIES", ICALEPCS 2013, San Francisco, USA, (2013)

[11] A. Dworak et al. "THE NEW CERN CONTROLS MIDDLEWARE", CHEP 2012, New York, USA, (2012)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE CONTROL SYSTEM FOR TRIM-COIL RELAY-SELECTORS IN J-PARC MR

K. C. Sato*, S. Igarashi, N. Kamikubota, N. Yamamoto,
J-PARC, KEK & JAEA, Ibaraki-ken, Japan
S. Yoshida, Kanto Information Service (KIS), Accelerator Group, Ibaraki, Japan

Abstract

In J-PARC main ring, each of the main magnets (Bending, Quadrupole and Sextupole) has a trim-coil. The primary aim of trim-coil is to correct small deviation of each magnetic field. In addition, we use them for other purposes such as: (1) In Beam-Based-Alignment studies, (2) as flux monitors, and (3) to make a short-circuit to reduce ripples of magnetic field. At a moment, each trim-coil can be used for only one purpose. We developed a relay-selector system which enables selection of connection to equipment depending on purpose. When we switch the connection, we have to change 1,200 on-site relays manually, distributed in three buildings. Thus, a control system for trim-coil relay-selectors have been developed in 2014-2015. EPICS tools and environment are used to develop the system. The system comprises PLC I/O modules with a controller running EPICS on Linux. Its operation begun in April, 2015. By using the system, we expect much easier switching of relay-selectors and reduce overheads than before.

INTRODUCTION

J-PARC (Japan Proton Accelerator Research Complex) is a high-intensity proton accelerator facility. It has been operated collaboratively by Japan Atomic Energy Agency (JAEA) and High Energy Accelerator Research Organization (KEK). It consists of three accelerators: a linear accelerator (LINAC), a Rapid Cycling Synchrotron (RCS), and a Main Ring (MR). MR started beam operation in 2008 [1]. MR is an accelerator with circumstance of about 1570 meter. This paper aim to main magnets accelerator components.

There are 3 types in the MR main magnets: bending magnet (BM), quadrupole magnet (QM) and sextupole magnet (SM). The number of BM (QM, SM) is 96 (216, 72). They are arranged with a 3-fold symmetry in the MR tunnel. The main power supplies are installed in the MR power-supply buildings. As an example, the cabling layout of a typical family "QFN" is shown in figure 1. The power supply for QFN is located in the second power-supply building, and QFN magnets are wired clockwise. Each of these magnets has an independent trim-coil.

The trim-coil can be connected to four different equipment such as:

- "Harmonic" - to correct small deviation of each magnetic field.
- "BBA" - in Beam-Based-Alignment studies.

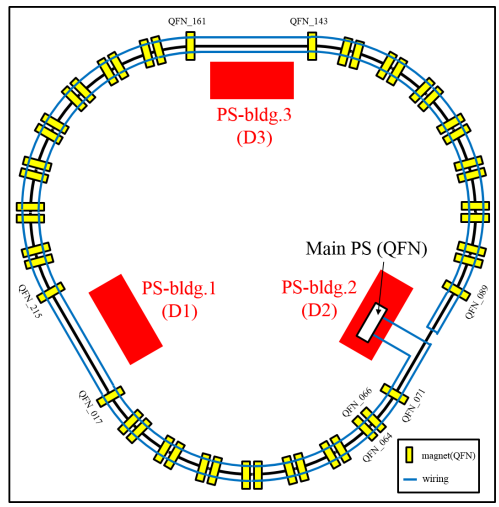


Figure 1: The layout of QFN power supply cabling.

- "Flux" - as magnetic flux monitors.
- "Short" - to make a short-circuit to reduce ripples of magnetic field [2].

In 2014-2015, we have developed the relay-selector system. Before the relay-selector system is installed, we have to switch connections between a trim-coil and a corresponding equipment by hands. All trim-coil connectors are placed in MR three power-supply buildings. So we need a lot of time to change connections. This relay-selectors system can reduce these overhead time since need not to switch the connection by hands.

By the way, the relay-selector system is applies to trim-coils of QM and SM. Since the BM trim-coils are used only for the "Short" purpose, it is independent from the system.

TRIM-COIL RELAY-SERECTORS SYSTEM

Each trim-coil need 4 relays. As shown in Figure 2 and 3, a relay-unit has 8 relays for 2 trim-coils. An aggregation-unit gathers cables from 8 relay-units. A control panel with buttons and fault indications is mounted in a rack. A relay-selector system, which consists of three rack, has 6 aggregation-units. Thus, a relay-selector system has to control 384 relays. J-PARC MR have three power-supply buildings, and each building has one relay-selector system. In total, this control system need to switch 1152 relays (equal to 288 trim-coils x4).

* kenichi.sato@j-parc.jp

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

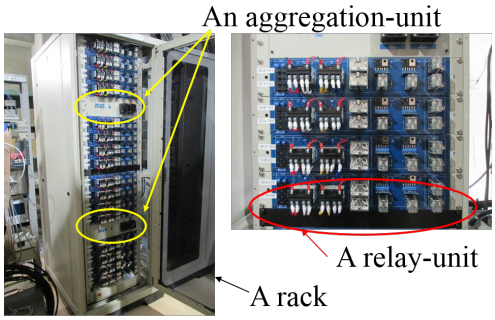


Figure 2: The photo of a relay-selector rack and a relay-unit.

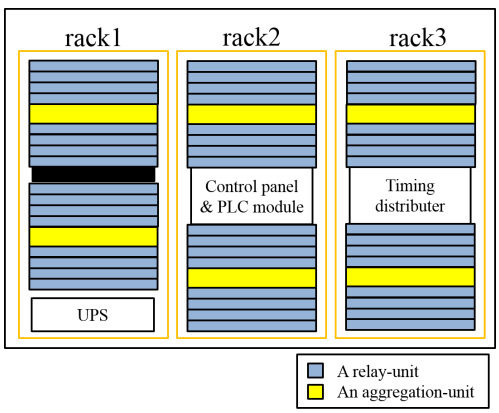


Figure 3: The layout of relay-selector.

EPICS PV CONFIGURATION

PLC Controller and I/O Modules

In J-PARC, we have been using an EPICS-based control system for the accelerator equipment [3] [4]. In J-PARC MR, a PLC-type CPU module running embedded Linux and PLC I/O modules (Yokogawa FA-M3 series) [5] [6] are standard for general I/O purposes. Relay-selectors are connected to PLC I/O via their aggregation-units. The buttons and fault indications of a panel are connected also to PLC I/O modules. Module cabling layout is shown in Figure 4.

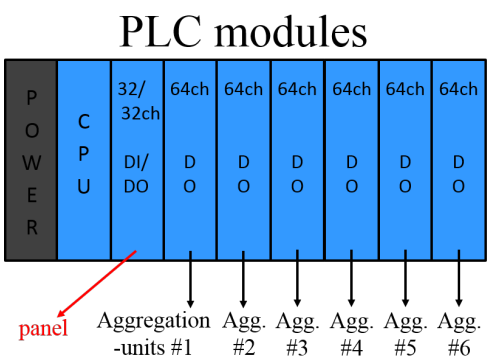


Figure 4: The layout of PLC modules and cabling.

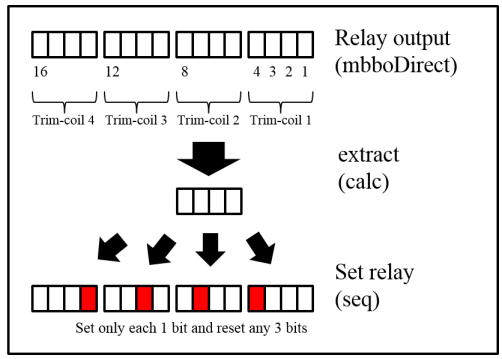


Figure 5: The footnote about PVs of relay output.

Design of EPICS PVs

Created PVs (Process Variables, a signal unit defined in EPICS) and numbers of them per one relay-selector system are shown in Table 1. The lower part of Table 1 concerns bit-oriented relay-output controls. Relationships of control PVs are shown in Figure 5.

Table 1: The Number of EPICS PVs of a Relay-Selector System

PVs	EPICS-type	number	explanation
Status	bi	5	system power, trigger
Interlock	bi	12	temperature, fan
Operation	bo	6	power and remote
Test switch	bo	7	test sw, trigger disable
Relay output	mbboDirect	24	see Figure 5
Extract	calc	96	see Figure 5
Set relay	seq	384	see Figure 5
Reset relay	seq	96	clear all

One “relay-output” PV, which is a 16-bit signal, contains 4 purpose of 4 trim-coils. The 4-bits are extracted as the “extract” PV, corresponds to one of four trim-coils. This PV shows the status of the selected purpose of the trim-coil. Before changing a trim-coil purpose, four relays are cleared by the “reset relay” PV. The reason is to avoid multi-bit-on status. Then, one of four relays, corresponds to the target purpose, are set by “set relay” PV.

GUI DEVELOPMENT

The total system, consists of three relay-selector systems, has 1152 relays. We have developed a top screen and four screens correspond to four purposes.

Figure 6 shows a top screen for total control. Using this screen, main power-switches are controllable. The status of remote-enable switches and fault indicators (temperature

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

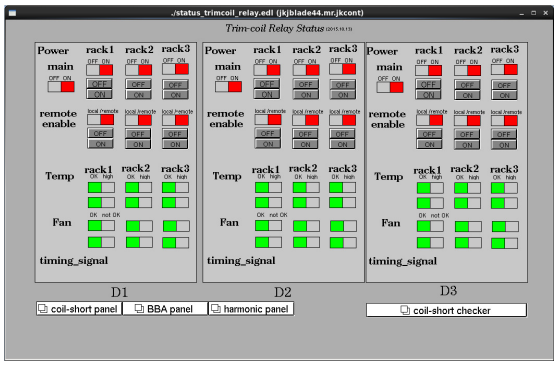


Figure 6: Top screen for the total control.

and fan) can be checked. Moreover, the screen has "open-new display" buttons.

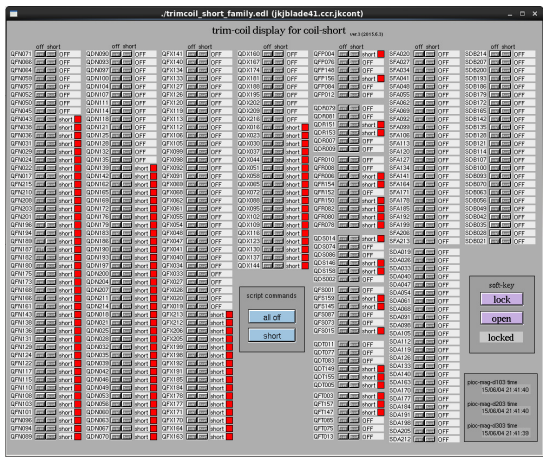


Figure 7: The control screen for switching to "short".

In the following, two screens of four purposes are shown. Figure 7 is the "Short" screen. As the distance from the power supply increases, ripples of magnetic field also increase. Thus, the trim-coils far from the power-supply are often set "Short". As shown in Figure 7, we can select any combination of "Short" trim-coils using this screen. The red indication in Figure 7 means that "Short" is selected.

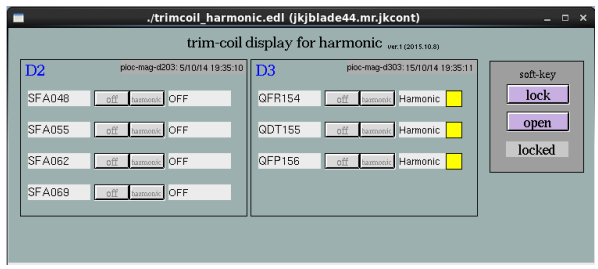


Figure 8: The control screen for switching to "harmonic".

Figure 8 is the "Harmonic" screen. The "Harmonic" purpose needs only 7 trim-coils. Set and reset of each trim-coil is possible. The yellow indication in the Figure 8 means that "Harmonic" is selected.

CONCLUSION

The three setups of the control system for the trim-coil relay-selector were developed. Operation started in April 2015. By using the system, we reduce the overhead time to change relay connections considerably than before.

REFERENCES

[1] T. Koseki, et al., "Beam Commissioning and Operation of the J-PARC Main Ring Synchrotron", Progress of Theoretical and Experimental Physics (PTEP) 2012, 02B004, 10.1093/ptep/pts071.

[2] S. Igarashi, et al., "Magnetic Field Ripple Reduction of Main Magnets of the J-PARC Main Ring using Trim Coils", Proceedings of IPAC'10, Kyoto, Japan, pp301-303, 2010.

[3] N. Kamikubota, et al., "J-PARC Control toward Future Reliable Operation", ICALEPCS 2011, Grenoble, France, Oct. 10-14, 2011, MOPMS026, pp. 378-381.

[4] http://www.aps.anl.gov/epics/

[5] http://www.yokogawa.co.jp/rtos/rtos-index-ja.htm

[6] J.-I. Odagiri, et al., "Application of EPICS on F3RP61 to Accelerator Control", ICALEPCS 2009, Kobe, Japan, Oct. 12-16, 2009, THD005, pp. 916-918.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DESIGN AND DEVELOPMENT OF THE ECR ION SOURCE CONTROL SYSTEM

Hyungjoo Son, Sangil Lee, Chang Wook Son, Hyojae Jang, IBS, Daejeon, S.Korea

Abstract

The Rare Isotope Science Project at the Institute for Basic Science constructs a heavy ion accelerator (RAON) facility in South Korea. The stable ion beam for the RAON accelerator could be generated by ECR ion source system. Therefore, it is necessary to build an ECR ion source control system that could be integrated into an accelerator control system easily. The vacuum control system is divided several parts because of one vacuum chamber among three different voltage stages (ground, 50 kV, and 80 kV). In this report, we will present the preliminary design and implementation of vacuum control system for the ECR ion source. We plan to use a Programmable Logic Controller (PLC) in order to control the vacuum system through interlock logic program. The PLC system has two major components: a digital I/O module that provides power to each component and standard RS-232 modules to connect the gauge and pump controllers. In addition, we will discuss its extension plan to integrate the vacuum control system into the RAON accelerator control system based on the EPICS framework.

SYSTEM CONFIGURATION

The driver linac injector of the RAON consists of a 28-GHz superconducting Electron Cyclotron Radiation (ECR) ion source, the LEBT (low energy beam transport), the 500-keV/u RFQ (radio-frequency quadrupole) and the MEBT (medium energy beam transport). For the ECR ion source, superconducting magnets and dual high power RF sources of 28 GHz and 18 GHz are used to improve its performance [1]. The high voltage ion sources could get from two different high voltage platforms (50kV and 80kV). The Vacuum control system for the ECR ion source is consisted of Allen-Bradley PLC (AB PLC) modules. The AB PLC chassis consists of four chassis and are installed each of the electrical potentials racks. Each vacuum control devices are connected with AB PLC modules to control turbo pumps and to read pressure of the vacuum chamber. Vacuum gauge controller (XGS-600) is used to read pressure and to communicate with AB PLC through serial cable using RS232 protocol. Similarly, OSAKA turbo pump controller and LAYBOLD turbo pump controller are used to operate turbo pumps with AB PLC through serial cable. In order to construct the network system for connection among multi-voltage stages, we used remote IO modules 1756-AN2TR and 1734-AENTR of AB PLC. 175 6-AN2TR and 1734-AENTR modules are used to connect each of two racks on ground state through LAN cable by MOXA switch. And 1783-ETAP2F modules has used to connect among one rack of ground state and two racks installed on high voltage stages (50 kV and 80 kV) directly through optical fibers because communication failure occurred when LAN cable is used. The basic configuration of the control system is indicated by the network diagram shown in Fig. 1. Dashed lines are optical fibers and solid lines are LAN cables. Each chassis are installed at each platform as below Fig. 2. Internet Protocol (IP) address is assigned to two areas 192.168.1.* (area A) and 100.100.100.* (area B) according to voltage platforms level to reduce the risks from high voltage difference. The area 'A' is connected to total network of the test facility that included the ECR ion source facility. The area 'B' is local network that connects between remote IO modules of the AB PLC only. Because the IP address is not enough when configure the total network system of the test facility. The control system performs the interlocks for the vacuum system of the ECR ion source. And this system will be integrated with the Experimental Physics and Industrial Control System (EPICS) to operate the system record the parameter values by EPICS Input Output Controller (IOC) using "process variables" in real-time.

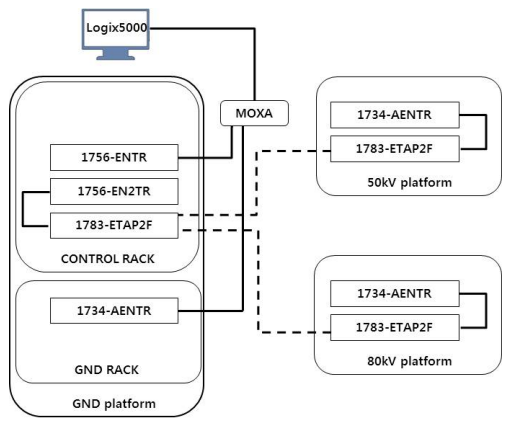


Figure 1: Diagram of network system for ECR ion source control system. 1783-ETAP2F modules for optical communication are used to connect between the high voltage platform and the control module that installed on ground platform.

VACUUM CONTROL SYSTEM

Vacuum control system of the ECR ion source consists of several turbo molecular pumps and controllers. They are installed at each voltage platforms. TC 353 turbo pump controller is installed at the ground platform and TC 2403 is installed at the 50kV platform. Both are made by OSAKA vacuum. TD 20 manufactured by LAYBOLD vacuum company is installed at 80kV platform. Each controller communicates with serial communication module (1734-232ASC) of AB PLC through serial cable. An AB PLC CPU 1756-L71 is used to control entire system of the ECR ion source and is installed at ground platform control rack of area 'A'

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Figure 2: Installed chassis on each platform to configure the vacuum interlock system of the ECR ion source.

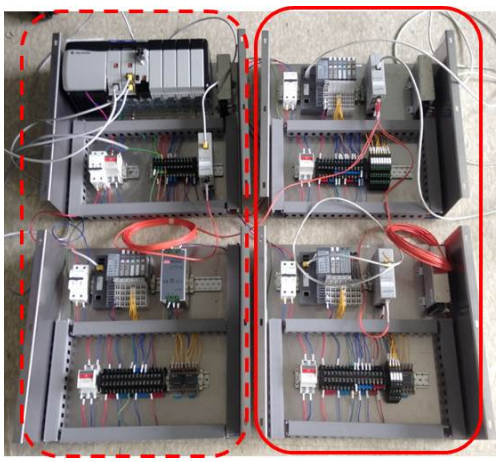


Figure 3: Preliminary test setup for confirming the network and I/O wiring. Left dashed box in figure is area 'A', Right solid box is area 'B'.

(Fig. 3). As can be seen from the Fig. 3, four slots of the control rack are emptied to install MODBUS communication module to integrate on the EPICS framework after local control test. We performed communication test among the AB PLC and the OSAKA turbo pump controller (TC353) as non-handshake method, the vacuum gauge controller (XGS600). TC353 comes with local and remote capabilities but we can't select these modes at the same time during the serial communication mode. TC353 and XGS600 are coupled with each other to control the vacuum system. We used Logix5000 software to send the command message and to receive the return message with each controller through RS-232 protocol. We completed communication test successfully to set each value of the parameters and to operate vacuum device. The electric wiring with vacuum components for IO control is proceeding after communication test. Presently, PLC modules are installed at each rack as shown in Table 1. The ladder logic programming for sequence control of the PLC is developed with Logix5000 software using Window based PC. The RSLinx software is used for setting network environment of the AB PLC. The ladder logic program will be designed independently to protect vacuum devices of the ECR ion source. The interlock program designed by PLC could perform protection for the vacuum system even when upper control system is shut down as system fail.

Table 1: PLC Modules

RACK	Module	Q'ty	Function
Control	1756-PA72	1ea	Power supply
	1756-ENTR	1ea	Networking
	1756-EN2TR	1ea	Networking
	1783-ETAP2F	1ea	CPU
GND	1734-AENTR	1ea	Networking
	1734-232ASC	2ea	Communication
	1734-IB4	1ea	Digital Input
	1734-OB8	1ea	textDigital Output
	1734-IE2C	1ea	Analog Input
50kV	1783-ETAP2F	1ea	Networking
	1734-AENTR	1ea	Networking
	1734-232ASC	2ea	Communication
	1734-IB4	1ea	Digital Input
	1734-OB8	1ea	Digital Output
	1734-IE2C	1ea	Digital Input
80kV	1783-ETAP2F	1ea	Networking
	1734-AENTR	1ea	Networking
	1734-232ASC	2ea	Communication
	1734-IB4	1ea	Digital Input
	1734-OB8	1ea	Digital Output
	1734-IE2C	1ea	Digital Input

We configured demo vacuum control system like the vacuum system of the ECR ion source to test serial communication for turbo pump controller and vacuum gauge controller as Fig. 4. Because ECR experiment group are performing test for developing the ECR ion source manually. We performed communication test among AB PLC and the OSAKA turbo pump controller (TC353) as non-handshake method, the vacuum gauge controller (XGS600). TC353 comes with local and remote capabilities but we can't select these modes at the same time during the serial communication mode. TC353 and XGS600 are coupled with each other to control the vacuum system. We used Logix5000 software to send the command message and to receive the return message with each controller through RS-232 protocol. We completed communication test successfully to set each value of the parameters and to operate vacuum device.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

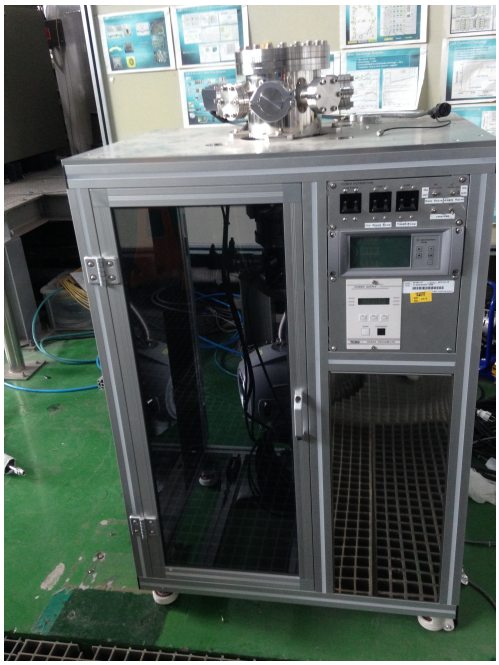


Figure 4: Demo vacuum control system with turbo pump and several vacuum gauges.

SUMMARY

We completed control test to operate vacuum pump and gauges through demo vacuum control system for the ECR ion source vacuum control. We will modify the vacuum control system of the ECR ion source to operate the system automatically through hard wiring between devices and PLC modules that are operated currently manually.

The system will be integrated with EPICS framework through Modbus TCP/IP module or Ether-IP module of the AB PLC. We are developing the EPICS IOC to control the vacuum system in real-time using EPICS drivers. The User Interface (UI) for monitoring and operating the system will be developed by the Control System Studio (CSS) software to provide easy control environment for the users. The vacuum control system of the ECR ion source is finally designed by the ladder logic program to perform the interlock checks continuously without data from the EPICS IOC so that the PLC can perform its protection functions even when the IOC is shut down [2].

ACKNOWLEDGMENT

This work is supported by the Rare Isotope Science Project funded by Ministry of Science, ICT and Future Planning(**MISP**) and National Research Foundation(**NRF**) of Korea(Project No. 2011-0032011).

REFERENCES

[1] D. JEON et al., “Design of the RAON Accelerator System”, Journal of the Korean Physical Society, Vol. 65, No. 7, October 2014, pp. 1010 ~1019.

[2] M. E. Bannister†, F.W. Meyer, and J. Sinclair, ORNL, Oak Ridge, TN 37831-6372, USA.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CAN OVER ETHERNET GATEWAYS - A CONVENIENT AND FLEXIBLE SOLUTION TO ACCESS LOW LEVEL CONTROL DEVICES

G.Thomas, D.Davids, CERN, Geneva, Switzerland
O.Holme, ETH Zurich, Switzerland

Abstract

CAN bus is a recommended fieldbus at CERN. It is widely used in the control systems of the experiments to control and monitor large amounts of equipment (IO devices, front-end electronics, power supplies). CAN nodes are distributed over buses that are interfaced to the computers via PCI or USB CAN interfaces. These interfaces limit the possible evolution of the Detector Control Systems (DCS). For instance, PCI cards are not compatible with all computer hardware and new requirements for virtualization and redundancy require dynamic reallocation of CAN bus interfaces to different computers. Additionally, these interfaces cannot be installed at a different location than the front-end computers. Ethernet based CAN interfaces resolve these issues, providing network access to the field buses. The Ethernet-CAN gateways from Analytica (GmbH) were evaluated to determine if they meet the hardware and software specifications of CERN. This paper presents the evaluation methodology and results as well as highlighting the benefits of using such gateways in experiment production environments. Preliminary experience with the Analytica interfaces in the DCS of the CMS experiment is presented.

INTRODUCTION

The control systems of CERN require the monitoring and control of a wide range of front-end equipment, much of which is installed in technically challenging locations with exposure to ionising radiation, strong magnetic fields and significant sources of electrical noise. A range of fieldbuses, including CAN bus [1], have been evaluated and selected by CERN to enable robust communication with the front-end hardware, leading to their widespread adoption in the LHC accelerator and experiments. As the readout computers and corresponding bus interfaces must be located outside of the hostile environment, the field buses are often more than 100 metres in length.

In the case of CAN bus, the communication on the bus is commonly driven by an OPC Data Access (OPC DA) server that runs on a computer and exposes the monitored hardware process variables and commands via the standard OPC interface protocol. The OPC server enables any OPC compliant client to connect and access the front-end data, independently of the details of the low level protocol used on the CAN bus for each type of hardware. A standard CERN solution for building the supervisory control system layer is to use the commercial SIMATIC WinCC Open Architecture (WinCC OA) [2] control system toolkit which natively supports OPC communication.

CURRENT STATUS AND LIMITATIONS

At the time of the construction of the LHC accelerator and experiment control systems, the connection between the control system computers and CAN buses was typically provided by internal PCI or PCIe cards or standalone USB connected interfaces. These CAN readout systems worked reliably throughout the first running period of LHC.

Changes in the computers running the control system software have been a significant motivating factor to re-evaluate the choice of CAN interface. For instance, the use of blade servers offers opportunities to minimize the rack space used by control system computers. The most compact blade servers do not provide the possibility to add PCIe expansion cards and therefore the PCIe-CAN interfaces are no longer a viable option in this context.

Due to the ubiquity of USB interfaces on modern computer hardware, the USB-CAN interfaces continue to provide a method of connecting CAN buses to any server. However, the limitation of maximum USB cabling length means that the computer and USB-CAN interface are typically installed either in the same rack or closely neighbouring racks. So with both PCIe and USB interfaces, the server and CAN bus interface must be co-located requiring that the control software servers are installed in or near to the racks where the CAN buses currently terminate. This imposes strong limitations on the possible re-organisation of rack hardware. With the current USB or PCIe interfaces, any significant change of server location would involve the costly installation of new CAN cabling.

Furthermore, there is increased interest in the use of redundancy at the control server level and the use of server virtualisation. Using the redundancy mechanism offered by WinCC OA, each control application runs on two computers. For robustness against power failures and network issues, it is beneficial to locate the two servers in different physical locations. As both computers need to access the front-end hardware, each server must be able to connect to the CAN bus interface, which makes it impossible to rely on the existing PCIe CAN or USB CAN devices used previously. Likewise, in the case of server virtualisation, the physical location of the server that runs the control software is not necessarily known in advance so it is essential to have a CAN interface which can be contacted from any location, independently of the distance between the server and interface hardware.

Offering network access to the CAN interface by Ethernet delivers a solution to these issues. This concept was demonstrated at CERN in the TOTEM and CMS experiments by coupling existing USB-CAN interfaces to Ethernet connected USB hubs. Using this approach the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CMS DCS was able to relocate and reduce the server rack space and to implement control server redundancy, without needing to change the existing CAN cabling. From operational experience it became apparent that, while providing the required functionality, the readout chain with both the Ethernet-USB and USB-CAN conversions was complex and difficult to debug in case of problems. For this reason, CERN initiated an evaluation to find a direct Ethernet-CAN interface that would meet the existing and future requirements of the organisation.

The evaluation identified the Anagate Ethernet-CAN Gateway family of devices from Analytica GmbH [3] as a candidate for a standard CERN Ethernet-CAN interface. These devices are based on Linux running on ARM CPUs and are currently available in two generations of hardware, the first being based on a CAN controller chip from Microchip and the second using an FPGA to drive the CAN buses. The first generation hardware is available in models with 1, 2 and 4 CAN ports, packaged in stand-alone boxes. Additionally there is a 1U rack mountable unit which can contain up to three 4-port devices, giving a maximum of 12 CAN ports. The second generation hardware is DIN rail mountable and is offered in 2, 4 and 8 CAN port variants.

INTERFACE EVALUATION METHODS

It is essential to have stable CAN interface hardware in any production environment because any failure will prevent communication between the front-end hardware and the control system software. This can generate downtime of the experiments and creates frustration amongst the support teams. To minimise the issues and purchase the best hardware that would meet the end user's requirements, the following methods were used to evaluate the Anagate Ethernet-CAN gateways.

Defining CAN Interface Test Specifications

The first priority was to define a test specification list describing the objectives, procedures and pass/fail criteria to check that the Ethernet-CAN interface met the user requirements. The tests were grouped into 4 categories; namely Functionality, Usability, Performance and Stress. Only Performance and Stress test categories are described here in detail. The primary goal of this exercise was to define a unit test for each category that specified its objective, a testing procedure and the pass condition. This test case catalogue could then be used to evaluate the hardware and to report the test results.

- Performance Testing Objectives

One of the first questions when evaluating a new hardware is inevitably going to be: "How does it perform?" Alternative questions might be: What maximum throughput can it sustain? Can it handle high load? How fast does it respond? There are many different ways to ask the question but all of them refer to the performance of a product. In order to be able to answer these questions and to determine how effectively it can perform under our environmental conditions and constraints, a unit test specification was defined per performance test criteria. These unit tests included: throughput, endurance and latency. Throughput unit tests were devised to measure the maximum number of CAN frames per second the interface could handle per CAN bus before CAN frames started to be discarded by the interface. Tests were performed for each possible baud rate. Endurance unit tests involved running the interface with a significant load over a long duration to ensure that the throughput and response time did not degrade with long-term sustained load. Latency unit tests were designed to characterise the delay associated with sending CAN data through the interface hardware.

- Stress (resilience) Testing Objectives

It is also important to know how well the hardware behaves and recovers beyond its normal operation conditions. As the CAN interface is a key element in the readout chain, it is essential that it can recover quickly and automatically from failure conditions. It should not require a physical human intervention to act on the interface, since the hardware is distributed over distances of several kilometres and located in areas that are not always fully accessible during the operation of the experiment.

Stress testing involved putting the CAN hardware under exaggerated levels of stress to evaluate its stability in the non-ideal conditions encountered in the production environment. Three types of major failures were identified in order to assess the robustness of the interface and check how well it was able to recover.

Power failure unit tests involved powering off and on the interface via hardware and or software reboot during periods of active CAN communication and then analysing how the interface recovered from this state.

LAN failures tests consisted of making a physical disconnection of the network equipment (i.e. Ethernet cabling, Network switch, etc…) for varying durations during operation and analysing the recovery mechanism.

CAN network failure tests aimed at acting on the CAN bus to simulate errors such as putting a CAN port into BUS OFF and verifying that the interface could reset the CAN port without any negative impact on the system.

Writing of Software Tools

As described earlier, the control and monitoring data of the devices on the CAN buses is handled by WinCC OA, which communicates via a native OPC client to a vendor specific OPC server. This OPC server then communicates over CAN with the physical device concerned. The OPC servers were originally written for a CERN defined set of CAN interfaces, mostly from Kvaser AB [4], making use of the vendor specific hardware APIs.

To be able to use and evaluate the Anagate Ethernet-CAN gateways within the CERN control context, it was necessary to write a wrapper DLL that would map between the original vendor (Vendor 1) API used in the OPC server and the new hardware API from Analytica (Vendor 2). This is illustrated in Fig. 1. From the OPC server point of view, the wrapper DLL behaves in exactly the same way as the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Vendor 1 API, avoiding the need to rewrite parts of the OPC server when changing CAN interface hardware.

The wrapper DLL was implemented in C as a re-entrant multi-threaded library.

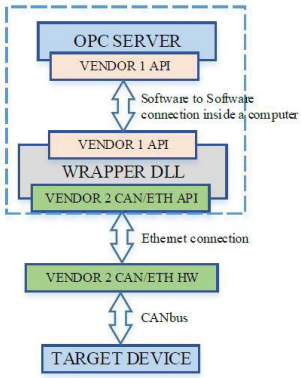


Figure 1: OPC-wrapper-CAN API software layout.

As the Vendor 1 API accesses the hardware via the CAN-port number and has no notion of how to access the Anagate hardware over the Ethernet network, a static configuration file is needed to associate hardware CAN-port numbers with the Anagate IP-addresses and IP-port numbers. To overcome the three types of major failures described in the stress testing section, a complex recovery mechanism was implemented in the wrapper DLL such that those problems become transparent for the OPC servers.

To facilitate the unit testing, a wide-ranging application for interactive testing was written in C. The test application was developed using the Kvaser API and can therefore use the wrapper DLLs to adapt to other CAN interfaces. The test tool can be called from batch scripts, enabling the automation of the required performance and stress tests. Additionally a set of PERL scripts were implemented to assist with the analysis of data generated during the tests.

The test application allows the execution of individual commands or scenarios of consecutive commands. For each CAN port, it is possible to instantiate independent sending and receiving threads. The traffic pattern of the sending thread can be controlled via three parameters: the number of CAN frames per burst, the time between bursts and the duration of the test in terms of the total number of bursts as shown in Fig. 2.

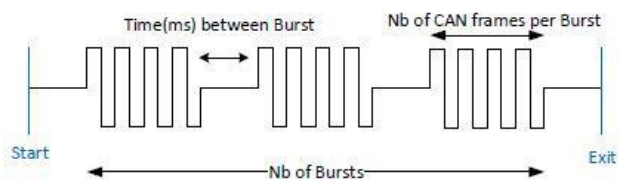


Figure 2: CAN frames transmit pattern.

An additional feature of the test application is that it can uniquely label CAN frames and record the delay between sending and receiving a given frame. By sending data on one CAN port and receiving it on a second port, it is possible to measure the latency of the round trip to the CAN bus and back, passing twice through the software and hardware interface layers. The latency statistics are written to file in the form of a frequency distribution of the delays.

TESTING AND RESULTS

The testing of the Anagate Ethernet-CAN gateway lasted for several months, with the aim of evaluating a maximum number of software and hardware aspects. Firstly, this involved checking the quality of the interface in terms of hardware parts (i.e. quality of CAN connectors, interface housing, etc…) and software elements (i.e. ease of installation, API readability, quality of documentation, etc…). Secondly, the unit tests described in the test specification document were carried out. Thirdly, the interfaces were installed in an experiment production environment to evaluate the stability and scalability over a period of several weeks with hundreds of CAN nodes distributed over several CAN buses. Throughout the testing period, the quality and responsiveness of the support from the vendor were evaluated.

Hardware and Software Product Inspection

After purchasing, the Anagate gateways were installed in the lab and the following criteria were evaluated:

- The quality of the housing and the integrated connectors (CAN ports, internet and powering) indicated that the hardware was robust.
- The ease and rapid installation and configuration of the interface demonstrated a documentation and software of good quality
- Testing of the native API methods could be rapidly evaluated with little programming effort.

Performance Measurements

Knowing the maximum throughput the interface could handle in receiving and transmitting CAN frames (before CAN frames were discarded) was the first performance criteria to be measured.

To achieve this, we used two Windows 64-bit computers and 3 Anagate CAN Quattro gateways mounted in a 1U case. We installed the hardware in a rack and connected two of the four CAN ports of the first Quattro gateway (which was the device under test) to each of the other two Quattro gateways. For each CAN baud rate (10kb/sec up to 1Mbit/sec) both receive and transmit CAN traffic was generated and passed through the device under test in order to determine the maximum CAN load before frames started to be discarded. The results are shown in Fig. 3.

The findings correspond to tests with a duration of five minutes. The maximum load percentages are the thresholds at which no frames are discarded. When those thresholds are exceeded, frames may be discarded by the Anagate hardware. As can be seen, the maximum load percentage diminishes for high CAN baud rates and when all CAN buses are used. These limitations correspond to the maximum throughput of the Anagate gateway that is consistent with data provided by Analytica.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

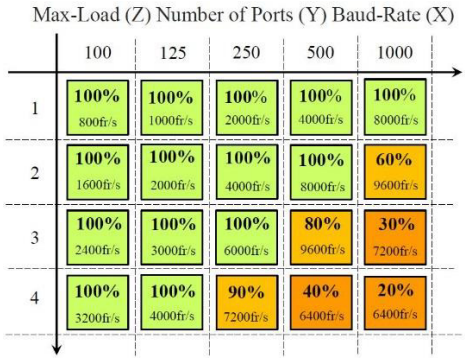


Figure 3: Throughput measurement results.

Latency Measurements

Several tests under different network topologies (with routers and switches) were performed to quantify the latency introduced by the Anagate hardware. The latency measured includes the time it takes to send and read back a CAN frame across the various layers (software, hardware and LAN).

The results in Figure 4 show the distribution of latency introduced by the Anagate hardware in comparison to a USB-CAN interface, when operating with low load. The average latency of the Anagate interface is slightly higher, but more significantly, the latency distribution is wider, with a long tail including longer delays.

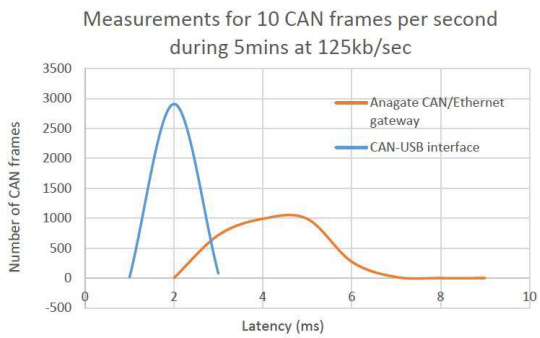


Figure 4: Latency measurements results.

Stress Testing Results

The Anagate interfaces were evaluated against several failure types that are encountered in a production environment. With recovery features included in the wrapper DLL, the Anagate interfaces automatically recovered from different type of failures such as power cut, network disconnection and CAN bus errors.

CMS Production Environment Tests

To test the Anagate hardware in a production environment, the Analytica interfaces were installed in CMS and integrated into the DCS in place of the previous USB-CAN interfaces. Tests covered the main types of CAN-based hardware to be controlled in CMS, namely power supplies and VME crates from WIENER Plein & Baus GmbH [5] and Embedded Local Monitor Boards (ELMBs) [6] designed by a collaboration between CERN, NIKHEF and PNPI. Large CMS CAN installations were selected to test the Analytica interfaces in the most demanding applications. Each test in Table 1 was executed for one week to evaluate performance and stability.

Table 1: CMS CAN Bus Test Applications

Hardware type	Devices	Buses	Baud rate
Wiener VME Crate	88	8	500 kb/s
Wiener Power Supply	136	10	100 kb/s
ELMB	104	8	125 kb/s

Initially, the Wiener OPC server encountered problems due to the additional latency of the Ethernet-CAN interface. The extra latency caused delays in the hardware response to OPC server requests, exceeding the expected round trip time and causing OPC item data to be marked as invalid. After discussion with Wiener, the problem was solved by modifying the OPC server to tolerate a user configurable additional latency in the readout chain.

The week long tests ran smoothly, indicating that the Anagate interface can be successfully integrated as a reliable component in the CMS DCS. An additional long term test was performed where the Wiener VME crate system ran for four months without any negative impact on the control system, in which time, more than 54 billion CAN frames were processed by the interface.

CONCLUSION

The Anagate Ethernet-CAN gateways from Analytica passed the unit tests including those which evaluated the functional, performance and robustness criteria. Furthermore, the tests in CMS demonstrated that the CAN interface could operate robustly in a production environment and that it provided the required quality of service for the experiment control system.

Throughout the testing period most of the issues reported to Analytica were analysed and fixed within a period of few weeks which demonstrated a very satisfactory level of collaboration and support.

The Anagate gateways offer a viable alternative to PCI and USB interface types, which enables the evolution of the CERN control systems towards virtualization and redundancy. There are ongoing studies for the integration of this interface into future OPC Unified Architecture based CERN control systems.

REFERENCES

[1] Q. King et al, “Fieldbuses for Control Systems at CERN”, June 2013, https://edms.cern.ch/document/1262875
[2] WinCC OA; http://www.etm.at/index_e.asp
[3] Analytica; http://www.anagate.com/index.html
[4] Kvaser AB; http://www.kvaser.com
[5] Wiener Electronics for Research & Industry; http://www.wiener-d.com
[6] Open Hardware Repository: CERN ELMB; http://www.ohwr.org/projects/cern-elmb/wiki

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

STRIPPING FOIL DISPLACEMENT UNIT CONTROL FOR H- INJECTION IN PSB AT CERN

P. Van Trappen*, R. Noulibos, W.J.M. Weterings, CERN, Geneva, Switzerland

Abstract

For CERN's Linac4 (L4) Proton Synchrotron Booster (PSB) injection scheme, slices of the 160 MeV H- beam will be distributed to the 4 superposed synchrotron rings of the PSB. The beam will then be injected horizontally into the PSB by means of an H- charge-exchange injection system using a graphite stripping foil to strip the electrons from the H- ions. The foil and its positioning mechanism will be housed under vacuum inside a stripping foil unit, containing a set of six foils that can be mechanically rotated into the beam aperture. The band with mounted foils is controlled by a stepping motor while a resolver, micro-switches and a membrane potentiometer provide foil position feedback. The vicinity of the ionizing beam and vacuum requirements have constrained the selection of the above mentioned control system parts. The positioning and interlocking logic is implemented in an industrial Programmable Logic Controller (PLC). This paper describes the design of the stripping foil unit electronics and controls and presents the first results obtained from a test bench unit which will be installed in the Linac4 transfer line by the end of the 2015 for foil tests with beam.

INTRODUCTION

As part of the LHC Injectors Upgrade (LIU), CERN has planned the replacement of the current Linac2 by the Linac4 as injector to the PSB. This new linear accelerator (linac) is expected to increase the beam brightness of the PSB by a factor of 2, making a Large Hadron Collider (LHC) injectors' upgrade possible for higher intensity and eventually a luminosity increase [1]. A combination of bending, kicker and septum magnets will distribute the 160 MeV H- ions to the four superposed PSB synchrotron rings. The beam will subsequently be injected horizontally into the PSB using the stripping foil. The orbit of the circulating beam is displaced by ~81 mm, using two independent closed orbit bump systems, to meet the incoming beam [2]. The first injection bump uses four pulsed dipole magnets (BSW) magnets while a series of 4 horizontal kicker magnets (KSW), located in the PSB circulating orbit, will produce an additional closed orbit dump with varying amplitude to accomplish transverse phase space painting to the required emittance. This is illustrated in Fig. 1 [3]. The injected beam will be stripped by a H- charge exchange injection system (i.e. stripping foil) where the ions will be converted to protons. Partially stripped H- and ~1% H- missing the foil will be directed to an internal dump.

In order to reduce machine downtime it will be possible to remotely interchange the carbon foil (~1 μm thick) when deemed necessary, e.g. in case of rupture or decreased performance. For that reason a foil interchange mechanism (FIM) has been designed, incorporating a rotating band with six interchangeable foils. This paper discusses the electronic actuators, sensors and control system that make a foil exchange possible.

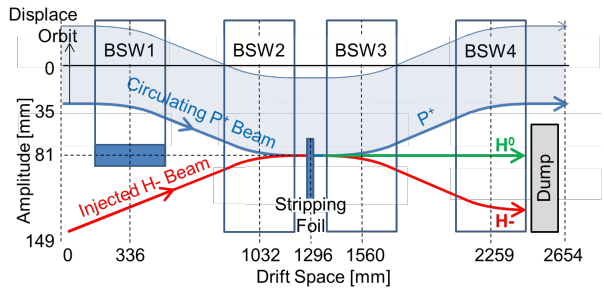


Figure 1: PSB injection region.

SENSORS & ACTUATORS

The FIM consists of a band, inside a vacuum chamber, that rotates over two pulleys so to displace a foil into the beam aperture. This will allow for a perpetual rotation so that each of the six foils can be reselected. Each foil is attached to a frame that has two important positions: *foil in* and *foil out*. Each of these positions allow for a 4 mm fine-tuning in order to find an optimum position for operation.

A retractable beam observation TV monitor (BTV) screen for beam position measurements on the selected foil is installed at the other side of the vacuum chamber, as shown in Fig. 2. When inserted it will be parallel next to *foil in* frame. To prevent BTV screen and FIM frame collisions, no FIM movement will be allowed when the BTV screen is in and at the same time any BTV movement is interlocked when the frame is not in the above mentioned positions. It is important to emphasise that a collision would result in major downtime because machine access and vacuum breakage will be required to replace the FIM. It is thus vital to know at all times the absolute position of the frame close to the beam aperture.

The choice of sensors is driven by obtaining that exact frame position and furthermore deal with the stringent vacuum pressure and radiation dose constraints. Furthermore the requested positioning reproducibility of 200 μm from the specification has to be taken into account.

Component Selection

The below electrical components are used to displace the band of ~50 cm so that one of the six foil frames is positioned into the injected beam aperture. The motor is the

* CERN TE-ABT-EC, pieter.van.trappen@cern.ch

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

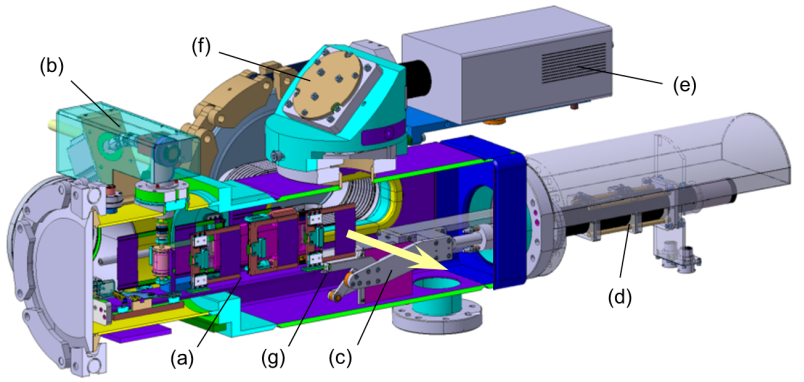


Figure 2: Cross section of the full design with (a) the FIM, (b) stepping motor, gearbox and vacuum feed-through, (c) BTV screen in retracted position, (d) the BTV motorisation, (e) BTV radiation hard camera, (f) mirror and optical filters unit and (g) the mirror positioned below the beam. The arrow indicates the beam direction.

sole actioning component, the other components are used for position readout.

Motor and Resolver The band pulley with a radius of 12 mm has its shaft connected to a mechanical vacuum feed-through. Outside of the vacuum chamber the feed-through is connected to a stepping motor through a 10:1 worm and wheel gearbox. Although some commercial off-the-shelf (COTS) stepping motors are made for operation inside the vacuum, placing the motor inside the chamber would endanger vacuum acceptance tests and complicate signal integrity and motor replacement.

The 1.8°stepping motor is microstepping driven which yields a higher positioning resolution and smoother frame movement. A microstepping factor of 8 steps per full-step has been chosen taking into account the microstepping disadvantages of torque variation and resulting loss of positioning accuracy. Taken all numbers into account the step resolution yields:

$$R = \frac{2\times12\times10^{3}\mu m\times\pi}{200\ \text{steps}\times8\ \text{microstepping}\times10\ \text{gearbox}} = 4.7\mu\text{m/step} \quad (1)$$

This is well below the required 200µm positioning accuracy and hence allows for several missed motor steps before the frame will be considered mispositioned. Although a stepping motor can function without feedback, i.e. open-loop control, a resolver has been added to provide command feedback. When the control system detects a difference between the motor command (i.e. the requested amount of steps) and the actual movement seen by the resolver, one can consider missed steps (steps loss). In this application the main cause of steps loss is mechanical friction that cannot be overcome by the motor's torque. Detecting this is crucial because it will result in mispositioning. Furthermore the small step resolution versus required accuracy is important because the asynchronous behaviour of the motor drive and resolver readout module means that several missed steps during positioning should be allowed, as it will be explained later.

For this application a resolver was chosen over an encoder because the elevated radiation doses don't allow for the electronics that are implemented in all but mechanical encoders. Mechanical encoders don't provide the required resolution for this application. Although a final motor has not been chosen yet, several suppliers offer motor-resolver combinations that are radiation hardened.

Microswitch The use of microswitches has several advantages because of their simple construction and operation. Several COTS switches, mainly for use in space, are made to minimize vacuum outgassing and resist radiation. Furthermore the open/close contacts accept a wide range of voltages and can be interpreted by any control system. From a mechanical point of view it is more challenging to implement these switches in the limited space of the vacuum chamber and route the cables through an electrical vacuum feed-through. The production of dedicated frame and microswitch supports has allowed us to detect the *foil in* and *foil out* positions over the 4 mm range. A microswitch is also used for calibration as explained later. After lengthy research, the Honeywell 17HM6 has been selected and successfully tested for outgassing. For each position two switches have been installed for redundancy. Figure 3 shows the two switches that indicate the *foil in* position.

Potentiometer While the closed-loop stepping motor and microswitches are able to provide frame position information, it was seen necessary to include an additional position measurement for reasons of reliability. Furthermore the stepping motor needs to be calibrated and while the calibration can be done using a microswitch, that method doesn't allow for actual position verification. Also microswitch degradation, caused by radiation, could displace its triggering point, hence altering the calibration point. Several sensors such as inductive and capacitive position sensors were evaluated but either they were not vacuum compatible

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

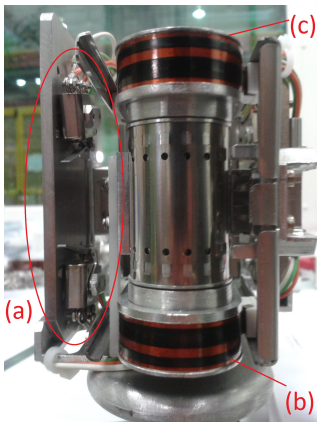


Figure 3: Front view with (a) microswitches and (b) (c) membrane potentiometers.

or they could not be physically fitted in the limited vacuum chamber space.

Because of the particular band movement it is clear that any linear position measurement won't cover the full range. However with a potentiometer that measures a single frame, preferably the one closest to the beam aperture, and with frame counter mechanism (see further in **Calibration**) an absolute position can be deducted. One has to take into account the limited chamber space and the fact that the frame closest to the beam aperture makes a rotational movement. These considerations have led to the choice of a membrane potentiometer. If the potentiometer consists of Kapton®, a polyimide film, and conductive ink it will experience limited outgassing and furthermore it can be bent around the pulley to follow the frame's movement. Germany-based Hoffmann+Krippner has produced a membrane potentiometer to our design. Figure 3 shows two of these potentiometers on the first test stand unit. The thicker line is the 20kΩ resistance while the smaller line makes up the conductor. Each frame has small wipers that short a certain part of the resistance with the conductor line. The measured resistance can then be translated to an absolute foil position.

Radiation Levels

The vicinity of the beam dump, mentioned in the introduction, is the main source of elevated levels or radiation. FLUKA [4] calculations show that some components can accumulate doses up to 10MGy which is why the microswitches and potentiometers will be sent to Fraunhofer INT for radiation tests. It is important to understand the degrading of the components as this might negatively impact the calibration which relies on for instance the potentiometer's linearity. The test results will also allow us to implement preventive replacement of the components when required. The foils themselves will become activated [5] so any unnecessary or unplanned intervention is to be avoided. The stepping motor and resolver are further away from the dump and can be commercially purchased to be radiation hardened so they will not undergo additional radiation testing.

Vacuum Cabling

For vacuum chamber cabling, no plastic-insulated cables will be used because the outgassing property of polymers cannot be accepted in the PSB vacuum. PEEK and Kapton® are to be avoided so all the cabling will be done by pure, uninsulated copper wire. The amount of microswitches and potentiometers result in a high number of cables that need to be insulated from each other. For this purpose COTS ceramic beads and tubes are used. All electrical contacts are passed through an electrical 26 pin High Density (HD) D-Sub vacuum feed-through.

CONTROL SYSTEM

The discussed components are all wired to the control system that is responsible for positioning the foils by actioning the stepping motor and reading the microswitches, potentiometer and resolver. There is no need for fast control so a PLC was chosen as main controller. PLCs, in general, provide out of the box axis control for positioning but due to the specific nature of the stripping foil this could not be used, as described below.

Hardware

The functional logic is implemented in a Siemens 1515F CPU with decentralised Inputs/Outputs (I/O) over PROFINET. For the final PSB installation there will be one master crate with the CPU and local touchscreen while each PSB ring will have a dedicated control crate with the I/Os. A safety PLC was chosen taking into account the machine downtime that a collision between a BTV screen and stripping foil could cause and not because of risk of human life which is normally the main reason. The implemented safety functions are surveilling the non-equivalent microswitch contacts and will open a contactor to open the motor phases in case of a detected discrepancy. Dedicated modules, for the resolver readout and stepping motor control & drive, have been integrated in the control chassis.

Positioning Algorithm

Calibration Positioning of the foil frames with a 200 µm precision relies mainly on the calibration of the band. Calibrating means finding a point on the band, the zero-point, which is considered to have the absolute position of 0 µm and from where all other positions are referenced. In case of a full turn the absolute position will set to zero again at that point. We know from Eq. 1 that the step resolution is small enough for accurate positioning, as long as the zero-point is found with an accuracy of better than 200 µm. The zero-point calibration error is however added to the existing calibration error with each full turn. Tests have shown that this accumulative error will result in exceeding the required precision after a few band turns. For reasons of consistency it was chosen to recalibrate with each full turn which only takes an additional few seconds. This is acceptable because of the band's slow motion, chosen so not to damage the fragile carbon foils.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Calibration Methods Three different calibration methods have been selected for evaluation, the most accurate two will be kept for reasons of redundancy. All three methods rely on detecting the zero-point at nominal speed, returning a small distance and moving forward with reduced speed for a more precise zero-point calibration.

The first method uses an additional microswitch that can only be triggered by one of the six frames, called the zero-frame, because of a special support on that frame. The second method uses the same microswitch for indicating the zero-frame but it will then use one of the two potentiometers to calibrate the system at a fixed resistance value. Finally the third method uses a linear actuator to mechanically block the band and relies on the resolver to detect missed steps. This method is the least favourite because it requires the motor's torque to be high enough to make the band turn, but low enough so that a blocked band is detected as quick as possible to have an accurate zero-point calibration. A stepping motor's torque can easily be controlled by limiting the current through the stator phases but this is a trial-and-error method and because of slight mechanical construction and friction differences this is expected to be different for every unit constructed.

Absolute Reference Axis A PLC or stepper driver has a certain functionality to allow for automated positioning and axis control. In order to use that functionality however the system assumes certain basic configurations and a band that allows full band turns and recalibration is not one of them. That's why for this project the motor driver is used in relative positioning mode and an algorithm to allow for absolute positioning has been developed for the PLC. A well-known methodology called *finite state machine* has been used to have precise control over the (re)calibration and positioning methods and asynchronous commands to the motor driver. The algorithm also deals with the asynchronous command and readout of the motor drive and resolver modules, as these operations are not synchronous with the PLC program cycle. This asynchronous behaviour imposes that during movement a certain discrepancy between the motor command and resolver position needs to be allowed. The small step resolution ensures that even in this situation the required positioning accuracy is within specification.

Test Results

The implemented positioning algorithm and the positioning accuracy, based on the zero-point recalibration, needs statistical validation so a test-run functionality has been implemented in the PLC. A laser distance sensor with a resolution of 20 µm and linearity of 100 µm has been purchased for position measurement and verification of the potentiometer's linearity in the test bench.

During a test-run the system will position itself to a configurable amount of positions on which all sensor's data is written to an comma-separated values (csv) file. That file can be read in by spreadsheet software for statistical analysis and graphical representation. A test-run that requested 500 successive 100 µm positions has been used to evaluate the potentiometer's linearity, as can bee seen in Fig. 4. At the moment of writing several test runs are being made to evaluate the three calibration methods and currently a standard deviation of 250 µm has been compiled from the data. The reason for that high positioning error has been found due to the mechanical axis feed-through, which allows for a small amount of slack, which is amplified by the pulley's diameter connecting the axis to the band. Work is ongoing to reduce this mechanical slack.

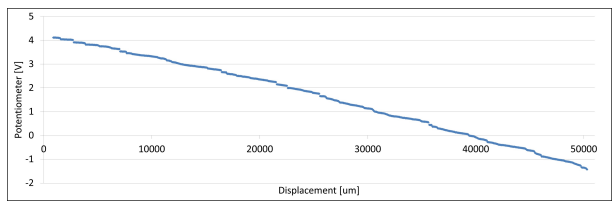


Figure 4: Potentiometer linearity test result, displacement (x) versus voltage (y).

STATUS AND FUTURE

The first unit will be installed at the end of the Linac4 Tunnel (L4T) for stripping foil performance tests. Currently that unit is being prepared for commissioning by the end of 2015 and is actually used as a test bench for the final validation of the mechanical and electrical system. As part of the Linac4 injection scheme for the PSB, the stripping foil exchange units need to be ready by the end of 2016 for an installation during LS2 (Long Shutdown 2) at the latest. The installation of two units, a definite one in the (L4T) and a temporary one as part of the half-sector test [6] scheduled in 2016, will provide us with useful experience that might require some improvements.

REFERENCES

[1] Gerigk F, Vretenar M et al, Linac4 technical design report, CERN-AB-2006-084.

[2] W. Weterings, B. Balhan et al, Status of the 160 MeV H^- Injection into the CERN PSB, CERN-ATS-2012-212.

[3] W. Weterings, C. Bracco et al, The stripping foil test stand in the Linac4 transfer line, J Radioanal Nucl Chem (2015).

[4] A. Ferrari, P.R. Sala, A. Fasso, and J. Ranft, FLUKA: a multi-particle transport code, CERN-2005-10 (2005), INFN/TC_05/11, SLAC-R-773.

[5] R. Froeschl, Activation of the stripping foils of the future H^- charge exchange injection into the PS Booster, CERN–RP–2015–048–REPORTS–TN.

[6] Chamonix 2014 Workshop on LHC Performance, Chamonix, France, 22-25 September 1014, CERN-2015-002.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A FAST INTERLOCK DETECTION SYSTEM FOR HIGH-POWER SWITCH PROTECTION

P. Van Trappen*, E. Carlier, S. Uyttenhove, CERN, Geneva, Switzerland

Abstract

Fast pulsed kicker magnet systems are powered by high-voltage and high-current pulse generators with adjustable pulse length and amplitude. To deliver this power, fast high-voltage switches such as thyratrons and GTOs are used to control the fast discharge of pre-stored energy. To protect the machine and the generator itself against internal failures of these switches several types of fast interlocks systems are used at TE-ABT (CERN Technology department, Accelerator Beam Transfer). To get rid of this heterogeneous situation, a modular digital Fast Interlock Detection System (FIDS) has been developed in order to replace the existing fast interlocks systems. In addition to the existing functionality, the FIDS system will offer new functionalities such as extended flexibility, improved modularity, increased surveillance and diagnostics, contemporary communication protocols and automated card parametrization. A Xilinx Zynq®-7000 SoC has been selected for implementation of the required functionalities so that the FPGA (Field Programmable Gate Array) can hold the fast detection and interlocking logic while the ARM® processors allow for a flexible integration in CERN's Front-End Software Architecture (FESA) [1] framework, advanced diagnostics and automated self-parametrization.

PRINCIPLE

In Fig. 1 a simplified schematic of a kicker system is shown. In general one can assume a system with matched impedances. A typical operation consists of charging the Pulse Forming Network or Line (PFN/PFL) to a voltage V by a Direct Current or Resonant Current Power Supply (DCPS/RCPS). When the main switch closes, a pulse of magnitude V/2 (if matched) propagates through the transmission line to the magnet. The pulse's energy is dissipated in the terminating resistor and full-field is established in the kicker magnet once the pulse reaches this resistor. Note that the magnet termination can also be a short-circuit, which doubles the magnet current because of reflection but also doubles the required fill-time of the magnet [2]. The pulse length in the magnet can be controlled by adjusting the timing of the dump switch relative to the main switch. When a sharp falling edge of the magnetic field is required for systems using a PFN, a clipper switch to ground is added between the PFN and the main switch. Typical values are 50 kV and 10 kA that traverse the magnet with pulse lengths between 500 ns and 600 µs.

During operation the system might be interlocked because of malfunctioning of one of the High Voltage (HV) switches. These malfunctioning causes are called *fast interlocks*, which

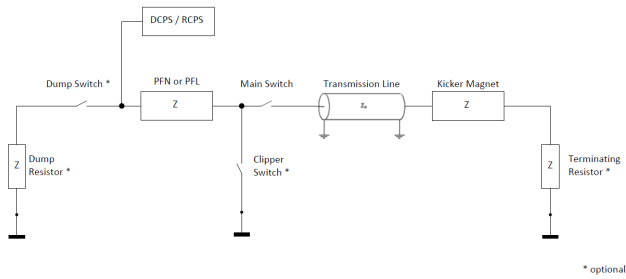


Figure 1: Simplified kicker system.

are detected over a large range of PFN voltages (i.e. the system's dynamic range). In addition to interlocking, they have often also a corrective action such as pulsing other magnets in order to fully deflect the beam and to avoid beam losses, i.e. to protect the machine. The following list gives a basic overview of the different types of fast interlock events:

- **Normal conduction** is not a fast interlock per se but for statistical reason it is useful to count the normal conduction cases.
- **Missing conduction** is registered when a trigger occurs but the switch is not pulsed. Lack of current in a HV switch or a missing load current after a certain amount of time after the trigger will result in a missing detection.
- **Erratic conduction** is the spontaneous conduction of an HV switch caused by tube or semiconductor malfunctioning as opposed to conduction following the normal triggering action. The FIDS mechanism registers an erratic when the switch conducts with no corresponding trigger pulse present.
- **Short-circuits** might happen at several locations, starting from the PFN up to a Terminating Resistor (TR). Depending on the magnet configuration several detection techniques exist, e.g. for short-circuited magnets one can compare the delay between the forward Main Switch (MS) pulse and the Dump Switch (DS) inverse current which, for normal operation, should represent the two-way transmission delay through the cable and magnet plus the single-way delay through the PFN.
- **No dump switch forward current** is the case when, for short-circuited magnets or systems with a Clipper Switch (CS) only, the short-circuit will reflect the travelling wave and the DS will have to withstand an inverse current (i.e. current from cathode to anode). In case of hollow-anode thyratron types, it is vital for the lifetime of the thyratron that it will first conduct forward current (i.e. anode to cathode) before the arrival of the inverse current.

* CERN TE-ABT-EC, pieter.van.trappen@cern.ch

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

A generator has several pick-up probes installed for analysis of the current and/or voltage. Resistive and capacitive voltage dividers are used as well as current transformers. The idea behind the FIDS is to sample these analog signals and their related trigger signals and perform an analysis to detect as fast as possible one of the above mentioned events. In case of an erratic conduction for example, we often need to pulse other magnets within 500 ns to reduce beam losses. After a certain amount of occurrences we would want to stop, i.e. interlock, the generator for expert intervention. Important in the analog signal sampling is a high signal-to-noise ratio (SNR), especially because many kicker generators have a wide dynamic range. In a simplified view the FIDS thus generates pulse outputs based on the relative timing of its input signals and communicates events to the upper-layer control system.

PROTOTYPE

A wide variety of electronic modules had been produced in the past and the FIDS principle of operation is based on an existing module, called Thyratron Protection Unit (TPU), by moving most of the functionality from analog discrete components to digital logic in an FPGA. This digitization will allow for more flexibility and an automated parametrization while adding improved communication possibilities. A prototype has been designed to verify the principle of operation and to perform several tests on dedicated installations available for development and testing. An FPGA was a logical choice to implement most of the digital logic because of the required reaction time (<500 ns) and the flexibility for designing hardware logic that differs per installation.

Figure 2 shows a block diagram that explains the prototype's functionality and the chosen hardware. All analog input signals need to be converted to digital values so that pulse edges and pulse lengths can easily be compared in time to detect a fast-interlock event. A possible method is to sample all signals with an Analog-to-Digital Converter (ADC) and use digital bit vectors inside the logic. Because of the high bandwidth of the pick-up signals (100 MHz) however a different approach was chosen. Analog comparators will be used to produce a 1-bit signal that indicate the presence of a current or voltage. This simplification will not limit the FIDS' performance because the signal amplitude itself is generally of no importance. It is important however that the SNR is taken into account for the full dynamic range of the input signals. This can be done by having the comparator reference follow the PFN's amplitude to control the comparators switching level at low PFN voltages. An additional advantage of the comparator-based approach is that additional latency due to ADC conversion and communication is avoided.

To implement the variable comparator reference the bipolar PFN voltage is sampled by an ADC after passing through an instrumentation amplifier and low-pass filter. Charging a PFN takes a considerable amount of time so a 10 kHz low-pass filters out transients and harmonics before digitalisation.

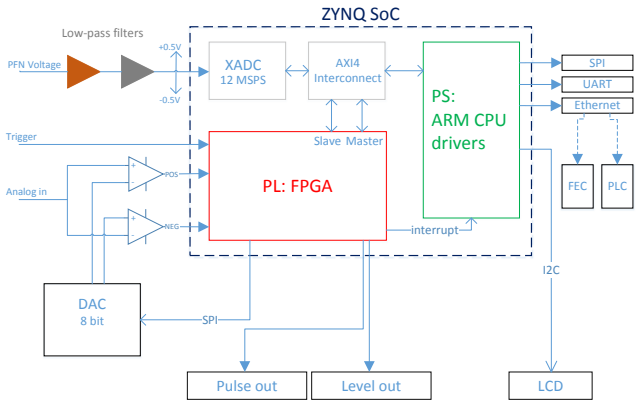


Figure 2: Prototype block diagram.

The digitalised value is used to set the comparator reference after applying a factor multiplication and offset addition. This factor and offset are user-selectable and depend on the kicker installation. A Digital-to-Analog Converter (DAC) translates the calculated value to an analog output that is used as comparator reference. Each bipolar analog input signal passes through two comparators so that a positive and negative amplitude can be detected, as often the case for the DS.

Hardware

Several FPGA evaluation boards were considered to reduce prototyping time. Taking into account the lifetime of the FIDS a recent FPGA family is preferred. Historically Xilinx® has been used in TE-ABT and conveniently their relatively new 7 Series FPGA family has two ADCs (12 MSPS) included - an ideal fit for sampling the PFN voltage. Furthermore a rather recent FPGA manufacturer direction is a System on a Chip (SoC) where an FPGA, ARM® cores and Input/Output (I/O) peripherals, such as I^2C and Gigabit Ethernet are included in one package. An internal interconnecting Advanced eXtensible Interface (AXI) bus can be used to link all components together - supported by the CPU cores as well. For the FIDS the Xilinx® Zynq®-7000 is an ideal fit for the prototype because the SoC can lead to significant development time gains. Finally the MicroZed™ low-cost development board was chosen. It includes the XC7Z010 SoC and several peripherals.

Software

The development time gain is mainly achieved by running a bare metal application on the CPU that allows to interact with the FPGA without much effort. The bare metal application is written in Embedded C and can easily access memory-mapped registers in the designed FIDS Intellectual Property (IP) core through the AXI4-Lite protocol. Thanks to the SoC implemented UART peripheral, user I/O is as easy as using printf() and getchar() functions. In the case of a fast-interlock event the IP core will trigger a CPU in-

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

terrupt which will put all data on the serial UART, such as the acquired PFN voltage and calculated comparator references. Thanks to the implemented I^2C peripheral driver and C library it was no problem to quickly add a 2x16 character LCD to let the prototype run for days without needing a laptop for FIDS output visualisation using the UART.

In a second stage the SoC is used by deploying an Embedded GNU/Linux distribution, crafted together by using the Xilinx'®PetaLinux tools. The main reason for doing so was to use the kernel's TCP IP stack to easily interface the integrated MAC controller. The idea is to integrate the FIDS seamlessly in CERN's control topology by running a FESA class on the Embedded GNU/Linux distribution. Currently FESA is not available for the ARM® architecture and a port has yet to be studied and/or implemented. At the moment of writing the FIDS prototype has a user-space program that uses a free software library to communicate with Siemens PLCs over Ethernet. This is a future implementation that will often be the case for a kicker installation. Implementing GNU/Linux allows for more flexibility and better control but to properly register the FIDS IP core's interrupts and memory a kernel driver had to be written. Hence additional design time has to be taken into account for the implementation and configuration of Das U-Boot bootloader, the rootfs and the device tree.

Gateware

The gateware for the FPGA is written in VHDL-2008, allowing for more generic packages and modules, and has two main functionalities:

- Interface with the XADC (Xilinx ADC), use of a linear formula to calculate the comparator reference and send it over SPI to the DAC.
- Detect the digitalised (single bit) pulse signals and generate fast-interlock events based on their relative timing and pulse length.

The implemented functionality was tested using a VHDL test bench with additional verification using the Property Specification Language (PSL) which is now part of the VHDL-2008 standard and supported by ModelSim.

Inside the design multiple clock-domains are used and to boost pulse measurement precision and pulse output latency, DSP48 slices have been used to clock the pulse length counters with a 250 MHz frequency. In a later stage the precision should be improved to 1 ns for which a more performant Zynq® SoC needs to be used so we can clock the ISERDES2 primitive at 1 GHz.

Preliminary Test Results

After successful lab tests, the prototype has been tested on two different types of dedicated test installations: the 50kV SPS Beam Dump System (SBDS) and the -60kV Transfer Kicker (TK). These installations are installed in different accelerators and have different PFN lengths and charging methodologies (DCPS versus RCPS). Also the magnet configuration is different because the TK has a terminated magnet with MS and DS configuration while the SBDS has a MS that consists of a parallel combination of a thyratron with series ignitrons.

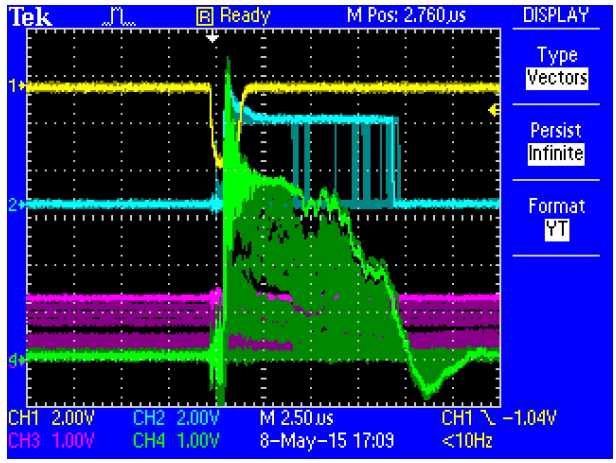


Figure 3: SBDS tests scope image; Ch1 (yellow): trigger, Ch2 (blue): comparator output, Ch3 (magenta): comparator reference, Ch4 (green): switch pick-up.

On Fig. 3 the oscilloscope waveforms of the SBDS thyratron switch are shown after a few hours of operation with infinite waveform persistence activated. The operating principle of the FIDS is clearly visible in this image because the installation was operated over its full dynamic range, i.e. from 5 kV to 45 kV. This PFN voltage, measured as a 0 to 2 V voltage from a resistive divider, is sampled by the XADC and used as comparator reference (Ch3 in magenta). When the PFN voltage increases, so does the reference according to the user parameters set by an FPGA register. In this example the linear formula used is y=1x+5 where the offset is controllable with 23 mV steps. A higher PFN voltage will result in an increased current through the thyratron (Ch4 in green). The comparator output, a digital signal, hence reflects the length of the current through the thyratron switch. This pulse is then used with reference to the trigger signal (Ch1 in yellow) to distinguish the fast-interlock event.

FULL DESIGN

After successful concept verification, a strategy to develop a board, suitable for every existing and future installation, had to be found. Taking into account the projected lifetime it is important not to select technologies or protocols that will soon become obsolete. The form factor and scalability are equally important because of the variable size of kicker installations with single to tens of HV switches. Finally after thoroughly analysing all requirements and multiple discussions with other groups at CERN, the FPGA Mezzanine Card (FMC) approach was chosen. This ANSI/VITA 57.1 standard specifies an I/O mezzanine module that works in conjunction with an FPGA processing device. The focus is to obtain mezzanine modules that minimize the handling and formatting of the transceived data while conveniently not forcing interface protocols [3]. This allows for example to route the comparator outputs with low-latency differential

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

signals directly to the FPGA while at the same time using a different subset of pins for DAC serial communication. The two main advantages of the use of the FMC standard for the FIDS are

- **Flexibility:** as mentioned before, kicker installations are complex and use several switches per generator and multiple generators per installation. Using a carrier with FMCs approach allows for adding functionality by simply plugging in an existing FMC.
- **Independency:** at the moment it is unclear what will be CERN's future CPU-bus and once chosen it is certain that this will change again, following the technology trends of the moment. Implementing the core functionality on a mezzanine allows for independency since it is easy to adapt to a new CPU bus by using the according carrier card.

In the following subsections the two basic parts of the system are detailed, being the FIDS FMC DIO 10i 80 module and the FIDS Carrier.

FIDS FMC DIO 10i 8o Module

While adhering to the FMC standard has many advantages, one should not forgot the disadvantages of increased design time, small physical size and limited freedom for the Power Distribution Network (PDN). An important advantage however is that at CERN several carriers for different CPU-buses exist and a working FMC can be quickly put in use by using one of these carriers, which are open hardware [4]. FMC is a widely accepted standard and also at CERN different cards with a wide range of functionalities have been developed. In general the additional time spent for fitting a design into the FMC standard pays off if there is enough demand for the card. A high demand, thanks to a general design, could encourage the industry to include it in their catalogue which will yield smaller lead times and lower prices. Also the more users incorporate the card, the faster design mistakes can be found and resolved. Below in Table 1 the card specification is listed and it shows that the analog input stage, including the comparator reference producing DACs, are included. While still a perfect suit for the FIDS, it can easily be used by other users as a general purpose I/O card that offers more user I/O than any other card available today. The card will be made available on the open hardware repository as well [4].

FIDS Carrier

As mentioned before at CERN several carriers such as the Simple VME FMC Carrier (SVEC) exist and the FIDS could make use of one of these to reduce overall design time. It was found however that none of the existing carriers included all of the required functionality so a specific carrier will be designed. The main features are two FMC slots, 6 analog inputs, fail-safe functionality and two Ethernet ports combined with the Zynq SoC. The full specification can be found in Table 2. They will be placed horizontally in a 19"rack and occupy a 1U slot.

Table 1: FIDS FMC DIO 10i 8o Specification

Parameter	Value
Inputs	10 analog comparator
Outputs	8 TTL 1 ns rise time
Number of comparators	20 (2 per input channel)
Comparator reference	20 (1 per comparator)
Comparator input bandwidth	1 GHz
Comparator input levels	± 5V
Comparators to FPGA	LVDS
FMC to carrier interface	Low Pin Count (LPC) connector
Input impedance	High-Z or 50 Ohm
Programmable threshold	DAC with 5mV precision
DAC resolution	12 bit
DAC Sampling rate	166 kHz

Table 2: FIDS Carrier Specification

Parameter	Value
2 FMC LPC slots	VADJ fixed to 2.5V for slot-1 and 1.8V for slot-2
6 analogue inputs	2 MSPS ADC, 1 MHz BW, 12-bit, lemo-00, ±10 V input
4 isolated outputs	9-pin D-Sub male connector
4 outputs	TTL level, rise-time < 10 ns
4 high-power outputs	15V, rise-time 20 ns, 1A peak output current
user-button	Momentary NO push-button
reset-button	Momentary NO push-button
8 bi-colour user LEDs	4 FPGA- and 4 CPU-controlled
Serial interface	I2C bus for connecting external LCDs and I/O expanders
JTAG header	Xilinx Platform Cable style
SPI Flash	Bootable, 2x 128 Mbit QSPI
SD card slot	For booting, OS mounting and persistent storage
Fail-safe functionality	FPGA and CPU watchdogs; voltage supervision; pulse output feedback
2 Ethernet RJ45 ports	Ethernet switch; 1 SoC Ethernet MAC; 10/100 Mbps

REFERENCES

[1] M. Arruat et al., "Front-End Software Architecture", ICALEPCS '07, Knoxville, USA, October 2007, WOPA04, p. 310 (2007).

[2] M.J. Barnes, Injection & Extraction Magnets II: Kickers, CAS 2009, Brugges, Belgium.

[3] ANSI/VITA 57.1-2008, American National Standard for FPGA Mezzanine Card (FMC) Standard, Approved July 2008.

[4] Open Hardware Repository, CERN, http://www.ohwr.org.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

UPGRADES OF TEMPERATURE MEASUREMENTS AND INTERLOCK SYSTEM FOR THE PRODUCTION TARGET AT J-PARC HADRON EXPERIMENTAL FACILITY

K. Agari#, Y. Sato, A. Toyoda, Y. Morino, KEK, Ibaraki, Japan

Abstract

Hadron experimental facility is designed to handle intense slow-extraction proton beam (750kW-15μA) from Main Ring of Japan Proton Accelerator Research Complex (J-PARC). On May 23, 2013, due to malfunction of the slow-extraction system in Main Ring, the production target in Hadron experimental facility was locally damaged because of overheat by absorbing proton beam in extremely short period (5 milliseconds). After the accident we have improved the monitoring system of temperature of the target with 100 milliseconds sampling rate in order to detect damage to the production target as soon as possible. The monitoring system has been operated without failure after the accident. This manuscript reports the upgrades of the temperature measurements at the J-PARC Hadron experimental facility.

INTRODUCTION

The Hadron experimental facility [1] (HD-hall) at Japan Particle Accelerator Complex (J-PARC) shown in Figure 1 is designed to handle intense slow-extraction proton beam from Main Ring (MR). The period of beam extraction from the MR to the HD-hall is 2 seconds and the operation cycle is 6 seconds.

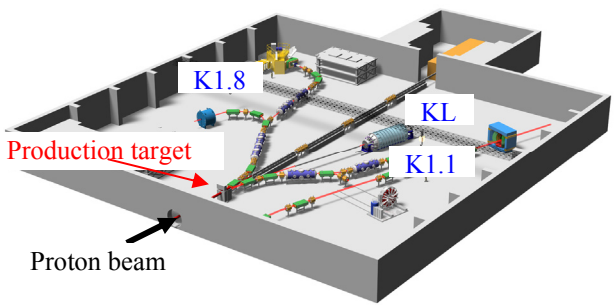


Figure 1: Illustration of the HD-hall.

On May 23th, 2013, 2×10^{13} proton beams were rapidly extracted to the HD-hall in 5 milliseconds due to malfunction of power supply of Extraction Quadrapole magnet in the MR. The production target [2] in the HD-hall was locally damaged because of rapid rise of temperature by beam deposit in extreme short period.

In order to detect the damage to the production target, the requirements of temperature measurement system are as follows.

- Upgrade read-out system of the production target temperature.
  - Hundred milliseconds sampling
  - Synchronization with beam extraction
- Make trend graphs and the waveform spectra of the target temperature as a function of time to tell the operators the state of the production target.

THE PRODUCTION TARGET

The production target at the HD-hall is made of gold and a copper block with coolant stainless pipes. Gold is chosen for high density, high thermal conductivity and good chemical stability. The total size of a gold structure is $15^{W} \times 6^{H} \times 66^{L}$ [mm]. The gold structure was divided into 6 pieces to reduce thermal stress. The water coolant pipes, embedded in the copper block, were made of stainless steel to avoid erosion and corrosion. The gold structure, the copper block and the coolant pipes are bonded by a Hot Isostatic Pressing (HIP) process.

The production target is located in an airtight chamber filled with helium gas. The beam entry and exit are covered with beam windows. Twenty thermocouples were attached at the surface of the gold pieces (12), the copper structure (2), the water and helium gas pipes (4) and the edge of the beam windows (2). The production target is designed to be capable for up to 50-kW proton beams.

MEASUREMENT AND CONTROL DEVICE

Before the accident we had measured the temperatures of the production target with 1-second cycle with a Programmable Logic Controller (PLC). The PLC had consisted of one EPICS-CPU, which is an embedded EPICS IOC on Yokogawa's FA-M3 PLC platform [3], three temperature monitor and one output modules. However, the temperature-measurement cycle of 1 second was too slow to detect the rapid temperature rise of the production target. Therefore we have upgraded the measurement system of the target temperature. Measurements of temperature with 100 milliseconds sampling are controlled with the sequence-CPU. The sampled data are stored in the sequent memory of the sequence-CPU. Then, an EPICS-IOC, running on the adjoining CPU module, installed next to the sequence-CPU, can take data from the sequence CPU via shared memory. Finally, the operators can monitor waveform spectra of the temperature rise on the production target, via the EPICS waveform record.

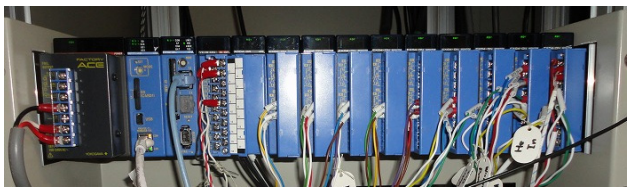
#agari@post.kek.jp

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

In the sequence-CPU, the temperature data of the production target are continuously compared to the thresholds for each thermocouple. An interlock signal is issued when the temperature exceeds the threshold. The interlock signal is transferred to the accelerator and the next beam extraction is safely stopped.

The start and stop timing of the measurement is synchronized with the beam extraction cycle, using the gate signal synchronized to the accelerator operation. The system of the PLC is shown in Figure 2.



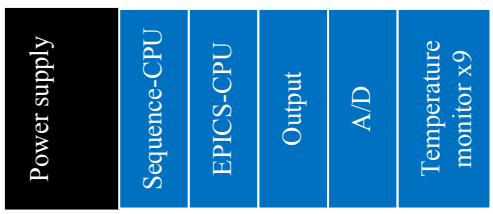


Figure 2: Photograph and illustration of the PLC.

METHOD TO MEASURE TEMPERATURES SYMCHRONIZED WITH BEAM EXTRACTION

We have developed the system of the temperatures synchronized with beam extraction to display the time structure of the target temperature. The way of the system is described in Figure 3 and as follows.

I The gate signal synchronized with beam extraction (ON-Beam) is detected by the A/D module in the PLC.

II When the leading edge of the gate signal is detected, the measurement starts with 100 milliseconds sampling rate, and ends after 5.8 seconds. The data are stored in the shared memory on the sequence-CPU.

III After the completion of each temperature measurement, the sequence-CPU issues the signal to the adjoining CPU module for data-taking.

IV The adjoining CPU starts to read temperature data via the shared memory, and store data into EPICS waveform records.

V Finally, the system waits for the timing gate of the next beam extraction after the completion of the temperature measurement.

The EPICS sequencer [4] has been used in order to operate the system synchronized with the beam extraction.

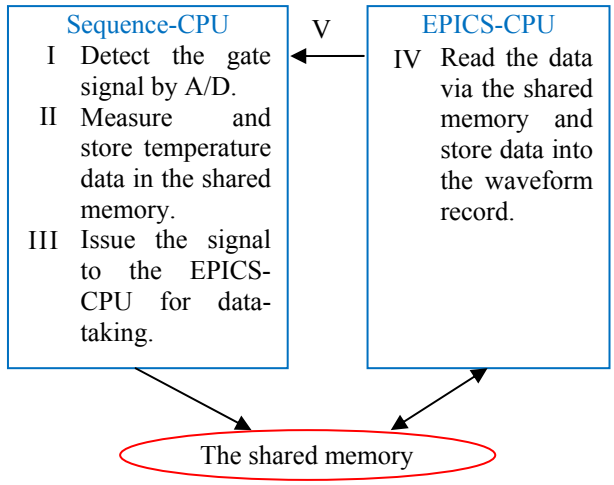


Figure 3: Diagram of how to measure the temperatures synchronized with beam extraction.

INTERLOCK AND ALARM SYSTEM

The interlock signals of the beam line components are used in the Machine Protection System [5] (MPS) in J-PARC. The MPS automatically stops beam operation when the MPS status is broken by the interlock signals from the accelerator and the beam line components. The EPICS Alarm Handler [6] has been introduced to notice the operators the warnings from the equipments. As for the production target, two-step thresholds ("Warning" and "Alert") for all temperature data are defined, using EPICS ALARM field (HIGH, HIHI, etc.).

The MPS signals from the target system are assigned to be "Alert".

DISPLAYS

Temperature data synchronized with beam extraction, trend graphs and MPS status are displayed in a control room for the operators. These displays are mentioned as follows.

Temperature Data Synchronized with Beam Extraction

A GUI program for the temperature data synchronized with beam extraction has been developed with wxPython [7]. WxPython is a GUI toolkit for the Python programming language. The operators can monitor the current data via this GUI program and inspect the past data if necessary. Figure 4 shows the typical waveform spectra of the target temperatures during the normal beam operation of 33kW. A smooth temperature rise during the beam extraction is measured from 0.6 to 2.6 seconds. The smooth decrease of temperature continues to 5.8 seconds from the beginning of the measurement. The maximum temperature of the gold pieces was 250.0 degree Celsius at the forth from the front piece. The temperature rises of the copper block and the coolant pipes were not more rapid than that of the gold pieces. The maximum temperatures and the averaged temperatures for last five

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

shots are displayed on the upper of this graph for beam commissioning.

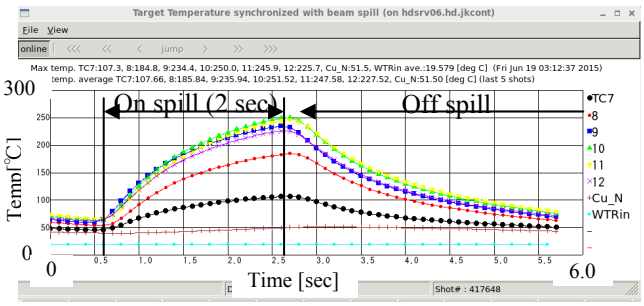


Figure 4: Display of temperature data synchronized with beam extraction during 33-kW beam operation. Its vertical axis is for the temperature [degree Celsius], and the horizontal axis is for the time [second].

Trend Graphs

The trend of temperature data are displayed by Strip Tool [8] as shown in Figure 5. The target temperature and the thresholds of "Warning" and "Alert" are shown in the same window. Figure 5 shows the typical beam operation continued to operate for 30 minutes without beam stop.

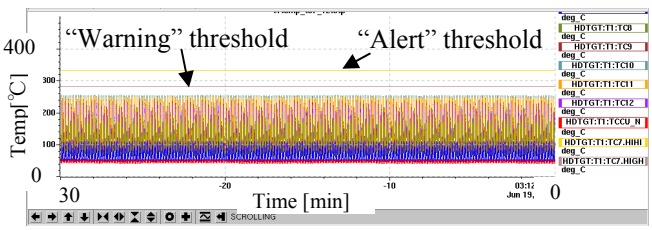


Figure 5: Display of trend graphs. The beam condition of this figure was the same as that of Figure 4. Its vertical axis is for the temperature [degree Celsius], and the horizontal axis is for the time [minute].

The Current Values and the Interlock Status

A GUI program for the current values, thresholds, and the interlock status of temperature has been developed with MEDM [9]. Figure 6 illustrates a top view of the production target. The operator can recognize the current values and the interlock status at a glance.

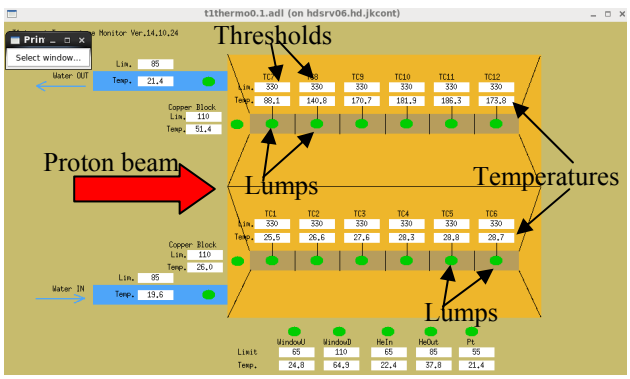


Figure 6: Display of interlock status.

SUMMARY

- The measurement and interlock system of the target temperature has been developed with the PLC. The temperature data and the interlock signals are handled and controlled with the sequence-CPU module. The EPICS-CPU module can handle the data from the sequence-CPU via the shared memory. The waveform records of temperature as a function of time can be referred on the EPICS-CPU.
- The measuring system operates to synchronize with the beam extraction. The operator can monitor the waveform spectra as a function of time in every beam extraction.
- The trend graphs, the current data of temperature, and the interlock status are shown visually for the operator.
- The two-step alarm system has been introduced for the operators, using ALH.
- The upgraded system has been successfully and stably operated with up to 33-kW proton beams.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 26800153.

REFERENCES

[1] K. Agari, et al., "Secondary charged beam lines at the J-PARC hadron experimental hall" , Progress of Theoretical and Experimental Physics (PTEP), (2012)

[2] H. Takahashi, et al, "Indirectly water-cooled production target at J-PARC hadron facility", Journal of Radioanalytical and Nuclear Chemistry, September 2015, Volume 305, Issue 3, pp 803-809.

[3] EPICS Device and Driver Support for Yokogawa's F3RP61 website: http://www-linac.kek.jp/cont/epics/f3rp61/

[4] EPICS Sequencer website: http://www-csr.bessy.de/control/SoftDist/sequencer/

[5] Y Sato, et al., "CONTROL SYSTEM FOR J-PARC HADRON EXPERIMENTAL FACILITY", Proceedings of ICALEPCS2009, Kobe, Japan, pp.331-333 (2009)

[6] EPICS Alarm Handler website: http://www.aps.anl.gov/epics/extensions/alh/

[7] wxPython.org website: http://www.wxpython.org/

[8] Strip Tool website: http://www.aps.anl.gov/epics/extensions/StripTool/

[9] MEDM website: http://www.aps.anl.gov/epics/extensions/medm/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE GENERAL INTERLOCK SYSTEM (GIS) FOR FAIR

F. Ameil, C. Betz, GSI Helmholtz Centre for Heavy Ion Research, Darmstadt, Germany
G. Cuk, I. Verstovsek, Cosylab d.d., Ljubljana, Slovenia

Abstract

The Interlock System for FAIR, called the General Interlock System (GIS), is part of the Machine Protection System which protects the accelerator from damage by misled beams.

The GIS collects various hardware interlock signals from up to 60 distributed remote I/O stations through PROFINET to a central PLC CPU. Thus a bit-field is built and sent to the interlock processor (PC) via a simple Ethernet point-to-point connection. Additional software Interlock sources can be picked up by the Interlock Processor via UDP/IP protocol.

The Interlock System for FAIR was divided into two development phases [1]:

- Phase A: interlock signal gathering (HW and SW) and a status viewer.
- Phase B: fully functional interlock logic with support for dynamic configuration, interface with Timing System, interlock signal acknowledging, interlock signal masking, archiving and logging.

The realization of the phase A will be presented here.

INTRODUCTION

The Interlock System for FAIR, called the General Interlock System (GIS), is part of the Machine Protection System which protects the accelerator from damage by misled beams.

The GIS will be used as a slower part of the Machine Protection System for all machines comprising the FAIR accelerator complex. Three components of the Interlock System can be identified [2] (Figure 1):

- the operation layer,
- the processing layer, and
- the signal pickup layer.

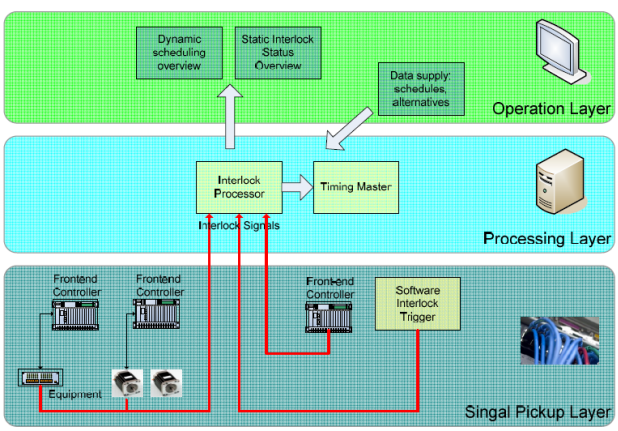


Figure 1: Layers of the interlock system.

The system must be able to collect hardware as well as software interlock signals from up-to 60 remote locations. Each remote station is capable of connecting to between 8 and 192 hardware interlock signals. Interlock transportation and processing must not take more than about 100 ms even when the system is extended to its limits - 4000 interlock signals. Based on these requirements, a combination of a PLC + interlock processor (PC) solution was chosen.

The purpose of Phase A was to confirm the selected architecture and components and to verify that such a solution is scalable and can reach the required performances. The solution will be installed and tested at the CRYRING accelerator in GSI.

The solution was developed by the company Cosylab d.d., Ljubljana, Slovenia, under supervision of the Control System Department at GSI.

SYSTEM DESCRIPTION

The main architecture of the GIS is presented in Figure 2 [3]. PLC Interface Modules (IM) with digital input modules are used to pick up hardware interlock signals. The control system network is used to pick up software interlock signals. After the GIS gathers both hardware and software interlock signals, it calculates summary interlock signals (specifics of this part will be defined and implemented in Phase B).

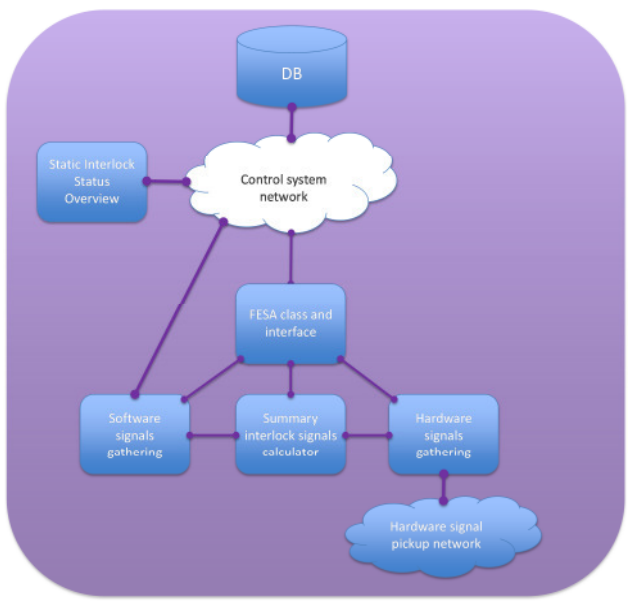


Figure 2: Architectural blocks of the GIS.

All changes of interlock signals are time-stamped. A user can use the GUI of the Static Interlock Status

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Overview application to monitor currently active interlock signals. This application obtains information about interlock statuses by connecting to the FESA class interface of the GIS.

Hardware Signals Pick-up

In Phase A, four remote locations were implemented, each hosting an ET200 SP module with a different number of digital input modules for collecting hardware interlock signals (Figure 3). In total 192 hardware interlock signals can be picked-up. If the collection of interlock inputs in some areas is out of operation, these inputs are considered as if they are in an interlock state and the rest of the GIS system remains fully operational.

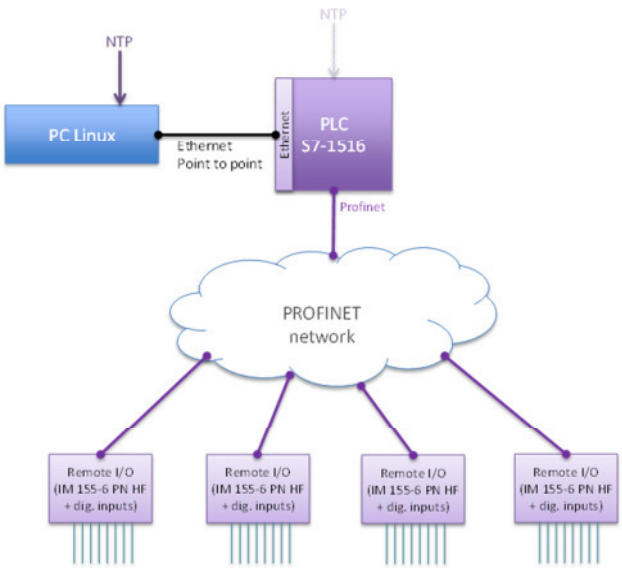


Figure 3: Hardware interlock signals pickup.

Remote stations are connected to a central CPU (S7 1516-3 PN/DP) via a Profinet network. The CPU runs with a fixed cycle time of 10 ms, thus defining the timestamp resolution. In each cycle, the PLC CPU gathers all hardware interlock signals and send them to Interlock Processor (PC) for further processing.

An extensive diagnostics is implemented to recognize different events and errors on the Profinet network and consequently put the relevant interlock signals into an interlock state. With those diagnostics, one can distinguish signalled interlocks from those caused by error, and furthermore the operator is able to see what kind of problem and in which location/module it occurred.

Software Signals Pick-up

Software Interlocks are collected by the Interlock Processor (PC) using UDP/IP as a communication protocol. For higher reliability, clients (systems sending software interlocks) send redundant datagrams whenever the interlock signal changes. The payload is encoded in such a way that critical bit-errors can be detected.

To make this interface and communication more reliable, the central station maintains a periodical communication (“heartbeat”) with all software interlock sources. An exception to this are interlock sources used for testing, for which the heartbeat functionality is not mandatory.

An interlock source can provide the status of its software interlock signal per Beam Production Chain in the same datagram. Also, if an interlock source provides many interlock signals, all of them can be signalled through a single datagram.

Interlock Status Overview Application

Phase A includes the Static Interlock Status Overview (SISO) application that serves for observing currently active interlock signals (Figure 4). The application connects to a GIS FESA class in order to obtain information about active interlocks, their type (hardware or software) and cause (signal or error) as well as their timestamps. Interlock signals are listed in the table and can be sorted or filtered by its name, type, cause or timestamp.

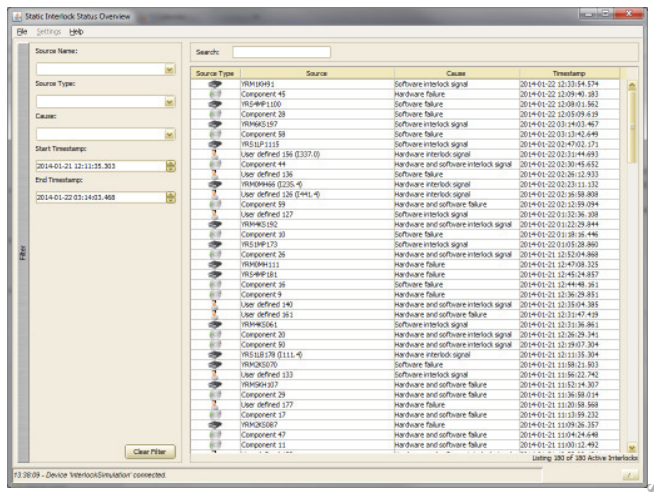


Figure 4: GUI of SISO application.

PHASE A - REALIZATION AND FAT

GIS phase A was developed by Cosylab. The system consists of a PLC CPU 1516-3 PN/DP and 4 remote stations based on IM 155-6 PN-HF, containing a total of 192 digital inputs (Figure 5). The PLC part of the system was developed in TIA portal v13. The rest of the functionality runs on the PC (not shown in the figure); it is implemented in C++ and integrated with the FESA class.

Implementation was followed by extensive factory acceptance testing with over 70 different unit tests covering all features of hardware and software interlock signals pickup, the FESA class and the GUI application.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

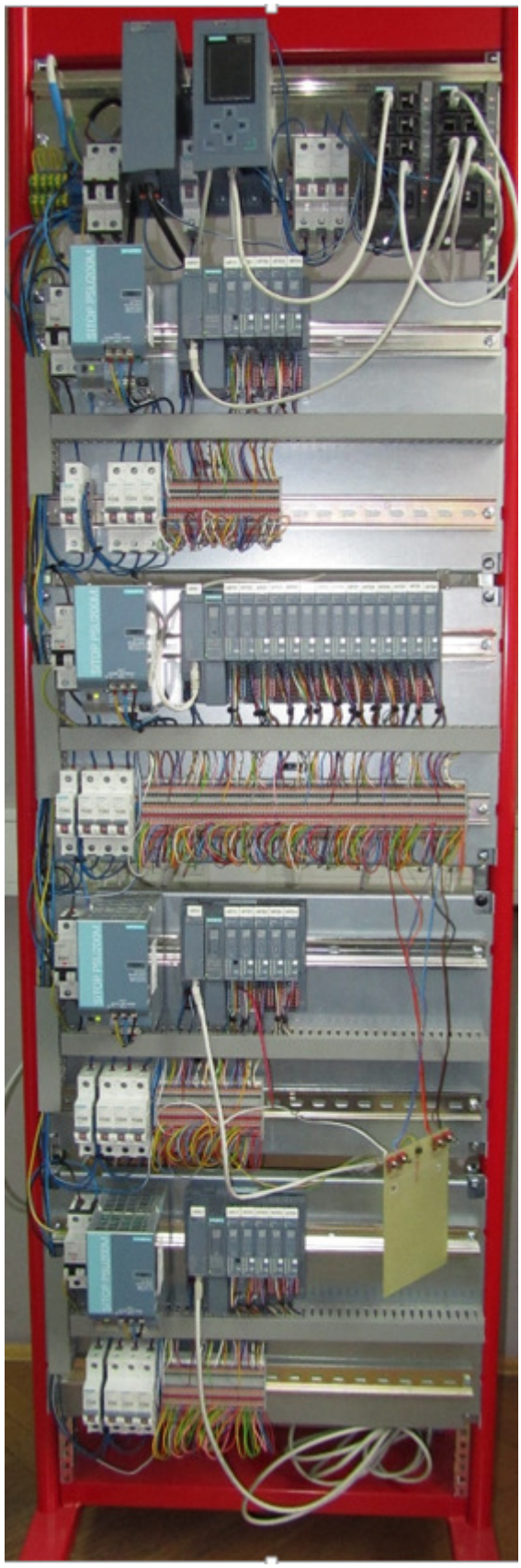


Figure 5: FAT configuration for GIS Phase A.

PERFORMANCE MEASUREMENTS

Two different measurements were done to confirm that the system complies with performance requirements:

- Cycle time measurement, using TIA portal.
- Response time, measured as difference between input and output signal change on two remote stations using an oscilloscope.

Phase A implements only a portion of the whole GIS system (4 out of 60 remote stations), so measuring performance of Phase A system only would not be enough. Therefore, a Profinet network simulator (SIMBA) was used to simulate the missing 56 remote stations with 3808 input signals (i.e. 4000 signals in total with the 4 physical remote stations from phase A).

With the cycle time measurement we want to show that system does not reach saturation even in the most intensive situations like failure of the remote station or even failure of the whole Profinet network. In normal operation, the GIS uses a fixed cycle time of 10 ms. In order to measure the effective cycle time, the fixed cycle time feature was disabled. Figure 6 shows the longest measured cycle time is 4 ms, therefore a fixed cycle of 10 ms has a safe margin.

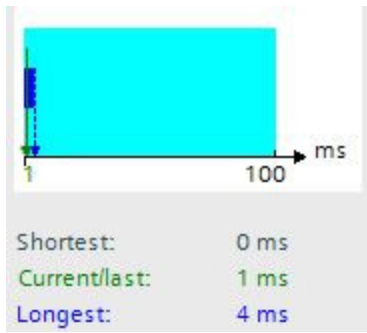


Figure 6: Cycle time measurement.

Real mitigation actions of the GIS will be implemented only in Phase B. To confirm that interlock transportation and processing takes less than 100 ms, we had to implement an additional feature of driving output signals on remote stations. This testing feature puts an output signal on the selected remote station(s) in a high state for a duration of one cycle each time any interlock signal change is detected. Measuring the time between the interlock signal going from high (OK) to low (NOK) state and the output signal going from low to high, gives the interlock system response time (which includes more than the initial requirement for transportation and processing time).

Different response times were measured because the result depends on when the change of input signal (done manually by opening contacts of the switch) occurs with respect to the start of the next PLC cycle [4]. Figure 7 shows the maximal measured response times (minimal response time was approx. 10 ms shorter). The pink line represents the interlock signal, transitioning from the OK to the NOK state. The yellow line is the response of the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

GIS (response was simulated by driving an output signal on each remote station).

These results show that transportation and processing takes far less than 100 ms.

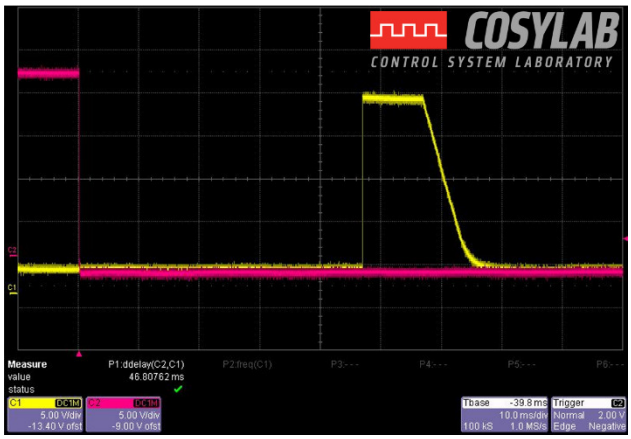


Figure 7: Response time measurement.

CONCLUSION AND OUTLOOK

The GIS for FAIR for Phase A was successfully developed and it fulfils the requirements. A Site Acceptance Test will be performed in the near future at CRYRING@GSI using the system developed for Phase A and operating under real conditions.

Another development is already in progress; its purpose is to include interlock state signals coming from other PLC systems (e.g. Vacuum Control System) using the I-Device functionality. Making use of this functionality avoids physically connecting many signals between two PLC systems; instead of physical signals, "logical" signals are exchanged via Profinet.

The system should be extended further in the next steps of the FAIR accelerator complex construction.

REFERENCES

[1] G. Čuk, FAIR Interlock System – Design Specification and Architecture, CSL-DOC-13-91257, version 2.1 (21.Nov.2013)

[2] J. Fitzek, G. Fröhlich, U. Krause: Detailed Specification of the FAIR Accelerator Control System Component "Interlock System", F-DS-C-08e, version 3.1 (11.Sep.2012)

[3] G. Čuk, FAIR Interlock System – Design document, CSL-DES-13-93467, version 1.4 (9.Apr.2014)

[4] G. Čuk, FAIR Interlock System – FAT Test Report, CSL-TRP-15-93918, version 1.0 (23.Apr.2014)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A MODIFIED FUNCTIONAL SAFETY METHOD FOR PREDICTING FALSE BEAM TRIPS AND BLIND FAILURES IN THE DESIGN OF THE ESS BEAM INTERLOCK SYSTEM

R. Andersson*[1, 2], A. Monera Martinez[1], A. Nordt[1], E. Bargalló[1]
[1] European Spallation Source, Lund, Sweden
[2] University of Oslo, Oslo, Norway

Abstract

As accelerators are becoming increasingly powerful, the requirement of a reliable machine protection system is apparent to avoid beam-induced damage to the equipment. A missed detection of a hazard is undesirable as it could lead to equipment damage on very short time scales. In addition, the number of false beam trips, leading to unnecessary downtime, should be kept at a minimum to achieve user satisfaction. This paper describes a method for predicting and mitigating these faults, based on the architecture of the system. The method is greatly influenced by the IEC61508 standard for functional safety for the industry and implements a Failure Mode, Effects, and Diagnostics Analysis (FMEDA). It is suggested that this method is applied at an early stage in the design phase of a high-power accelerator, so that possible protection and mitigation can be suggested and implemented in the interlock system logic. The method described in this paper is currently applied at the European Spallation Source and the results follow from the analysis on the Beam Interlock System of this facility.

THE BEAM INTERLOCK SYSTEM

The Beam Interlock System (BIS) at the European Spallation Source (ESS) receives around 300 beam permit input signals and needs to stop beam operation in as short as 4-5 µs for some sections after detection of an error in the proton linear accelerator (linac) [1]. The requirement is based on possible damage from the high proton beam power on the surrounding sensitive equipment. The time scale requirement for the BIS is combined with a need to stop the beam in a reliable way, avoiding both false beam trips (stopping the beam without a hazard) and blind failures (missing to stop the beam when there is a hazard). By meeting these requirements, the BIS can help reaching the reliability goals for ESS and at the same time spare the equipment of any unnecessary damage.

There are four types of modules in the BIS at ESS. Together, they resemble a combined tree and star structure for a reliable transfer of the beam permit signal from the inputs to the actuators. Fig. 1 shows a conceptual flowchart of the BIS, disregarding the actual structure for simplicity. The BIS consists of close to 300 Fast Beam Interlock Driver (FBI_D) and Device Interface (FBI_DIF) Modules. For redundancy, each input signal goes to two FBI_DIFs. These modules interface with all input signals and transmit a beam permit signal to the BIS. Each signal into the BIS will be propagated through redundant links over the entire signal path. Table 1 displays nominal reaction times for the different modules in the BIS together with an example of the input signal from a Beam Current Monitor (BCM) and the LEBT chopper actuator. The table considers a worst-case scenario in terms of required reaction times. As is seen, if the modules perform as designed, the BIS achieves a total reaction time of less than 3 µs in the low energy part of the linac.

The rack configuration at ESS contains 24 enclosures with 36 racks each. All the signals from one rack enclosure are grouped into one pair of Master modules (FBI_M), making 48 FBI_M in total. The 48 FBI_Ms connect into two redundant Master of Master modules (MoM) located at the front end of the linac. These two MoMs then connect to three different Actuator Modules (FBI_A) – one for each actuator: Proton Source, LEBT chopper, and MEBT chopper. The FBI_As deliver the final trigger signal to the actuators to stop the beam, by deflecting beam to an absorber by the choppers, and inhibiting the creation of plasma in the proton source [2]. The BIS position in the Machine Protection (MP) conceptual level is seen in red in Fig. 2.

Figure 1: The BIS (red) transmits the input beam permit signals from the monitors and sensors to the actuators, which stop the beam in case of a hazard.

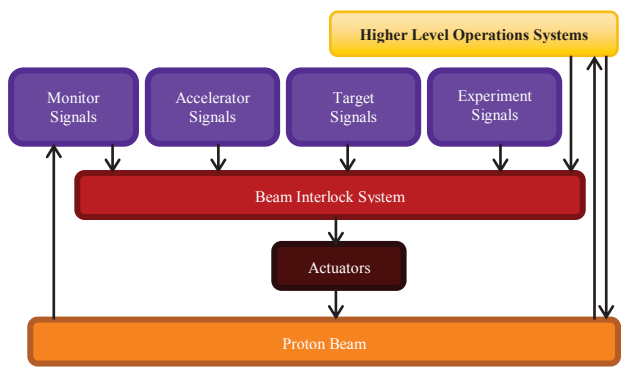


Figure 2: The BIS (red) role in the Machine Protection concept.

* riccard.andersson@esss.se

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

RELIABILITY ANALYSIS OF THE BIS

The reliability analysis of the BIS was made in two distinct parts. The first part consists of a Failure Mode, Effect, and Diagnostics Analysis (FMEDA) for each BIS module, described below. This part follows to a large extent the approach used for the Large Hadron Collider at CERN [3]. The FMEDA was done using a set of spreadsheets for module and component overview and easy backtracking. The second part uses Reliability Block Diagrams (RBD) on the system level using BlockSim from ReliaSoft [4] to obtain the overall BIS reliability figures.

Table 1: An Example of BIS Detection, Processing, and Reaction Times.

Module	Reaction Time	Accumulated Time
BCM	1 µs	1.0 µs
FBI_DIF	500 ns	1.5 µs
FBI_M	250 ns	1.75 µs
FBI_A	300 ns	2.05 µs
MEBT Chopper	300 ns	2.35 µs

FMEDA

The ESS BIS FMEDA involves (a) identifying the components of the four BIS modules, determine their (b) functionality, (c) failure rates, and (d) failure modes, (e) identify the components' impact on the module as a whole, and (f) find the diagnostic ability to detect the failures. The FMEDA process was launched in the mid 90s as an upgrade of the original Failure Mode and Effect Analysis (FMEA), developed in the 60s and 70s for the US Military Standards. It is a method that has grown increasingly popular in reliability engineering since its introduction.

The (a) components and (b) their functionality are extracted from the design of the circuit boards together with the designer. For the (c) failure rates and (d) modes, the US Department of Defense Military Handbooks 217F and 338B [5] were consulted, which provide failure rates per component and their failure mode allocation, respectively. In the cases where the component manufacturers provided failure rates from accelerated tests, these were used instead. The identification of (e) the components' impact on the module together with (f) the diagnostic abilities of the system is an iterative process together with the system designer in order to reach a satisfactory level of protection. An overview of the inputs and outputs of an FMEDA is seen in Fig. 3. It should be clarified that this process typically brings system flaws into light already at the first iterations and the system design itself is improved continuously throughout the FMEDA steps. Where applicable, the diagnostic coverage

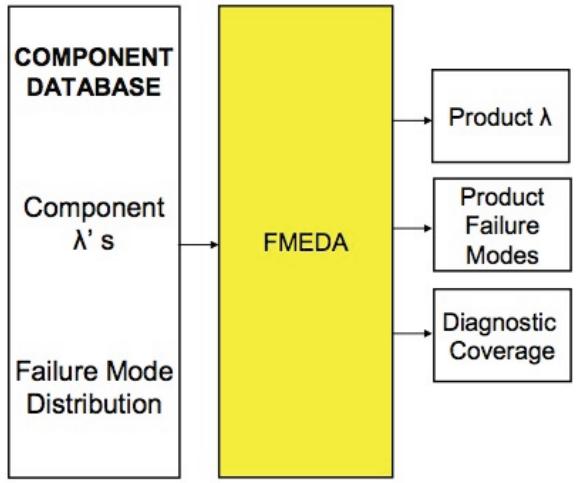


Figure 3: FMEDA inputs and outputs. Source: [6]

of the system can be enhanced in order to foresee and mitigate hazards before they cause the system to fail, as is seen in Fig. 3 where it is displayed as an output of the FMEDA.

From steps (a) through (f) above, the overall failure rates for the four different BIS modules are obtained, and these rates are then used in a system-level analysis using BlockSim. Four failure modes were identified as having impact on the system design and to be dealt with, being (in order of criticality) Blind, Trip, Maintenance, and Negligible. Blind means a blind failure where the system is unable to mitigate a hazard, Trip is a false trip where the beam permit is withdrawn erroneously, Maintenance means that the system can continue to operate under marginal protection but the component needs to be maintained "as quickly as possible", and Negligible means that the failure mode has no apparent effect on the system.

The diagnostic ability (f) was also divided into four different modes, being related to the failure modes and using the same color code. This way, one can easily get an overview of a failure mode's ability to be mitigated by the system's design and find possible flaws by quickly scrolling through the FMEDA spreadsheets. The diagnostic abilities are Test, Diagnostics, Inspection, and Hidden. Here, the Test is an internal test loop that checks the beam permit signal paths for faulty components, Diagnostics is the built-in diagnostic feature in the circuits able to detect anomalies in components, Inspection means that personnel has to go and inspect the component, and Hidden means that the error can not be seen without rigorous demounting and troubleshooting.

Reliability Block Diagrams

BlockSim allows for RBD or Fault Tree Analysis (FTA) approaches to be used for complex systems, in order to analyze reliability measures through both analytical computation and simulation. The BIS was modeled with RBDs to resemble the system behavior. Each module type received its failure rates through the FMEDA described above. As each block in the software

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

can only represent one failure mode, one system setup was created for each of the four failure modes. By altering the connections and number of input signals per module, it was possible to analyze different designs for optimum performance of the system. The system was analyzed for no redundancy at all, redundant links (as is the current design choice) and for a 2-out-of-3 voting design throughout the system (not included in this paper).

THE ESS AVAILABILITY GOALS

ESS aims for unprecedented beam availability for neutron production, which places high demands on all systems involved. The BIS is the central system in avoiding major damage to the beam pipe and surrounding equipment, which causes long downtimes, and needs to avoid false beam trips, which cause short beam trips and unnecessary downtime. This demands an almost fault-free system that always premieres stopping the beam over a hazard of damaging equipment.

Protection Integrity Levels

The IEC61508 standard [7] includes so-called Safety Integrity Levels (SIL) that define the level of which a certain Safety Function (SF) reduces the probability of a safety hazard. These concepts are both applicable and manageable in the ESS MP work, but as MP only uses the standard to a fractural extent and is mission critical rather than safety critical, the concepts are modified to fit the needs and processes of the MP development work. As MP deals with protection, the Protection Integrity Levels (PIL) are introduced together with Protection Functions (PF) that play the same role as their safety counterparts in the analysis [8].

The PF for avoiding Blind failures at ESS needs to reduce the failure rate to 10^{-6} per hour, or the equivalent of once in 114 years. This failure rate has then been allocated to three different major systems: the sensors, the BIS, and the actuators. The sensors are expected to contribute with 35%, the BIS with 15% and the actuators with 50% to the overall PF. That leads to a Blind failure rate requirement for the BIS of $1.5 \cdot 10^{-7}$. The BIS then has to be designed to reach this hardware failure rate [1].

As stopping the proton beam does not introduce any potential for damage, the false Trip rate is related to the overall ESS reliability and availability requirements [9] and its determination lies outside the scope of this paper. In general, however, one can find that the Trip rate is increased as the Blind rate is decreased, as the two are opposites in the realm of MP.

ANALYSIS RESULTS

Analysis Assumptions

The analysis was carried out assuming only hardware failures with 12 input signals per rack enclosure and 24 rack enclosures, giving a total of 288 signals and the same number of FBI_D and FBI_DIF. Each enclosure also has one redundant FBI_M pair and there is one redundant pair of MoMs, adding up to a total of 50 FBI_M. Finally, there are 3 FBI_A, one for each actuator.

Further, it is assumed that all actuators must trigger a beam stop signal and are thus modeled in series. Two prototype designs were considered, named Prototype 0.1 and 0.2. There were two simulations run for each prototype, one without redundant links and one with redundancy.

The component environment is taken to be at 40°C and 45-60% humidity to avoid corrosion, condensation, and static electricity build-up. The components are considered to be mounted in a fixed rack configuration.

Results

Fig. 4 shows the analysis results for the four setups. The bars are explained in the caption of the figure. A non-redundant system does not reach the required failure rate and needs further redesign. In this analysis, a redundant system was considered, which does still not reach the requirement for the initial design, but does so for the second prototype design.

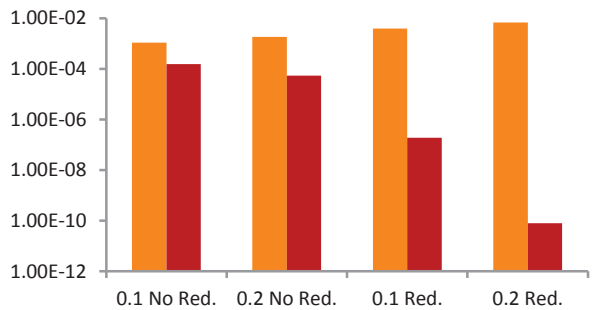


Figure 4: False trip (orange) and blind (red) failure rates for four different BIS designs. From left to right: Prototype 0.1 without redundancy, Prototype 0.2 without redundancy, Prototype 0.1 with redundancy, and Prototype 0.2 with redundancy.

BIS Design Features and Changes

While the current BIS design contains redundant links throughout the system, the option of a 2-out-of-3 voting system was analyzed. However, this also brings a cost issue into the design, as the component costs would increase by 50%. The FBI_Ms also have the ability to mask certain input signals, meaning that specific signals can be completely ignored for a predefined time. While this can increase system reliability in the case of certain components sending spurious signals to remove beam permit, there is also an associated risk with making certain inputs blind that has not yet been analyzed.

The iterative FMEDA process highlighted the weakest links in the BIS layout, being the FBI_A. These modules were a main cause of system-level Blind failures and the redesign of the system lead to intra-module redundancy that removed the Blind failures. A similar approach was carried out for the FBI_M where the Blind failure rate

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

was drastically decreased. The schematics were also separated into different parts separating for example permit signals from test loops and diagnostics. This gave a better overview of the system as a whole and allowed for more efficient troubleshooting.

It should be clarified that the analysis in this paper only considers hardware failures in the BIS. How surrounding equipment, software and firmware, and configuration failures affect the system are not part of the analysis. The same goes for the potential for failures associated with masking. As testing and fault injection are still to be performed for the BIS, the results described here should be viewed as theoretical on the conceptual design level.

EXTENSION TO OTHER SYSTEMS

The analysis performed in this paper is developed in a generic way and is not system-specific. This way, the FMEDA can be extended to other systems at ESS and at other facilities. The intension is to widen the analysis to include all the systems that feed signals into the BIS (purple boxes in Fig. 1 and 2) and the actuators (brown box in Fig. 1 and 2). This way, one would be able to determine the overall reliability from a signal, e.g. a beam loss, to the action of stopping the beam. This plays a central role in fulfilling the ESS reliability and availability goals and pinpoints the weak links that need to be improved further in the signal chain.

CONCLUSIONS

The development and actualization of the FMEDA together with the system-level RBD analysis of the BIS at ESS described in this paper have allowed for important design improvements throughout the iterative process. These improvements are fundamental to reach the high reliability and availability goals of ESS. It is important to note that these studies should be done at an as early stage as possible in the design process of the system, to avoid expensive and cumbersome changes last minute or even during operation.

The results show how a hardware failure rate of $1.5 \cdot 10^{-7}$ for the BIS can be achieved for Blind failures on a system level, at the same time as reliability is kept at a manageable level. By introducing a BIS with redundant links for the entire signal chain, the hardware failure rate requirement is reached without the need to introduce more expensive system designs.

ACKNOWLEDGEMENTS

We would like to thank Christian Hilbes and the Safety Critical Systems Research Team at ZHAW in Zürich, Switzerland for their contribution in the concept design of the Machine Protection at ESS. The analysis in this paper is greatly influenced by the similar study done at CERN, and therefore we want to thank Benjamin Todd, William Viganò, and the Beam Instrumentation Group there. Lastly, the University of Oslo has been of great support academically and financially for the work described in this paper.

REFERENCES

[1] Nordt, Annika, et al., "Development and Realisation of the ESS Machine Protection Concept", TUC3O03, these proceedings, ICALEPCS 2015, Melbourne, Australia (2015).

[2] Monera Martinez, Angel, et al., "Overview and Design Status of the Fast Beam Interlock System at ESS", MOPGF138, these proceedings, ICALEPCS 2015, Melbourne, Australia (2015).

[3] Todd, Benjamin, *A Beam Interlock System for CERN High Energy Accelerators*, PhD thesis, School of Engineering and Design, Brunel University, West London, UK, CERN-THESIS-2007-019 (2007).

[4] ReliaSoft website, "BlockSim: System Reliability and Maintainability Analysis Software Tool"; http://www.reliasoft.com/BlockSim/ (Accessed: October 2015).

[5] U.S Department of Defense, "Guide for Achieving Reliability, Availability, and Maintainability: Systems Engineering for Mission Success" (2005); http://www.weibull.com/mil_std/RAM_Guide_080305.pdf

[6] Picture taken from: http://www.exida.com/Resources/Whitepapers/ FMEDA-Accurate-Product-Failure-Metrics. Accessed: 2015-10-01.

[7] International Electrotechnical Commission, "Functional safety of electrical/electronic/programmable electronic safety-related systems", IEC 61508:2010, CENELEC, Brussels, Belgium (2010).

[8] Kwiatkowski, Maciej, *Methods for the Application of Programmable Logic Devices in Electronic Protection Systems for High Energy Particle Accelerators*, PhD thesis, Warsaw University of Technology, Warsaw, Poland, CERN-THESIS-2013-216 (2013).

[9] Bargalló, Enric, et al., "ESS Reliability and Availability Approach", MOPTY45, IPAC2015, Richmond, Virginia, USA (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

UNDERSTANDING THE FAILURE CHARACTERISTICS OF THE BEAM PERMIT SYSTEM OF RHIC AT BNL*

P. Chitnis#, T.G. Robertazzi, Stony Brook University, NY, USA
K. A. Brown, Brookhaven National Laboratory, NY, USA

Abstract

The RHIC Beam Permit System (BPS) monitors the anomalies occurring in the collider and restores the machine to a safe state upon fault detection. The reliability of the BPS thus directly impacts RHIC availability. An analytical multistate reliability model of the BPS has been developed to understand the failure development and propagation over store length variation. BPS has a modular structure. The individual modules have joint survival distributions defined by competing risks with exponential lifetimes. Modules differ in functionality and input response. The overall complex behavior of the system is analyzed by first principles for different failure/success states of the system. The model structure changes according to the type of scenario. The analytical model yields the marginal survival distribution for each scenario versus different store lengths. Analysis of structural importance and interdependencies of modules is also examined. A former Monte Carlo model [1] is used for the verification of the analytical model for a certain store length. This work is next step towards building knowledge base for eRHIC design by understanding finer failure characteristics of the BPS.

INTRODUCTION

The Beam Permit System (BPS) of Relativistic Heavy Ion Collider (RHIC) plays a key role in safeguarding against the developing faults in the collider. The energy stored in RHIC is about 2MJ in particle beams and 70 MJ in the form of magnet currents. Any abnormality in the machine can cause an undesirable escape of this energy, which can result into machine damage. BPS senses the onset of failures by monitoring the health inputs from RHIC support systems like power supplies, cryogenics, beam loss monitors, access controls, quench detection, vacuum system etc. In case of anomaly, it directs this energy out the machine for safe disposal.

RHIC reliability is directly dependent on the BPS reliability. Thus it is essential for the BPS to be highly reliable, being a safety critical system. Earlier a Monte Carlo (MC) model [1] was developed to quantify the probability of system level catastrophic events. This model simulated the behavior of modules with exponential competing failures. The module level failures were calculated by a quantitative fault tree analysis [2].

The MC model simulated the failure characteristics of the BPS for a certain store length. Also as the simulation takes about 17 hours to run 1E9 iterations to generate reproducible results. It generated probabilities of system level failures for a particular store length. This paper is aimed towards developing a model that generates the system failure probabilities as a function of store length. We develop an analytical model of the BPS behavior using stochastic mathematics for this purpose.

RHIC BEAM PERMIT SYSTEM

The BPS has a modular structure that consists of 33 Permit Modules (PM) and 4 Abort Kicker Modules (AKM). These modules are connected by three fiber-optic links that run 10MHz carrier links whose presence signifies that the system is healthy. These links are called the Permit Link (PL), Blue Link (BL) and Yellow Link (YL). The PMs collect support systems' health inputs from the field, called Permit Inputs (PI) and Quench Inputs (QI). The PM takes decision to drop the carrier, indicating a failure. When there is a magnet system failure, QI goes bad, and all the three links are dropped. In case of any other failure PI goes bad and only the permit link is dropped. The carrier drop propagates to all the modules, resulting respectively in magnet power +beam dump and a beam dump only. The AKMs send the beam dump signal to the beam abort system. They have redundancy incorporated. For more details of BPS refer to [3]. Figure 1 shows the BPS configuration.

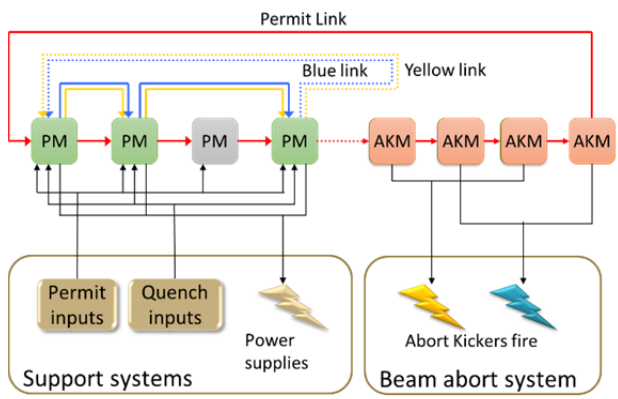


Figure 1: BPS connection diagram.

SURVIVAL DISTRIBUTION

The PMs and AKMs have been analysed for their failure modes in [2]. The top level failure of BPS depends upon the states of the PMs and AKMs. The calculation

*Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy.
#prachi.chitnis@stonybrook.edu

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

showed that the top level failures of the modules have exponential survival distribution, which is characterized by a constant hazard function λ. Modules can have these failure states, False Beam Abort (F), a False Quench (M), Blind (B) and Dirty Dump (D), with failure rates as λ_F, λ_M, λ_B and λ_D resp. The PMs have {F, M, B} or {F, B} failure modes (green PM and grey PM resp. in Fig. 1). The AKMs have {F, M, D}. The inputs PI and QI are modelled as Poisson variable with exponential trigger rate as λ_{PI} and λ_{QI}. The initial state is a Good (G) state, where the module performs its intended function properly. The Markov diagrams for states is shown in Fig. 2.

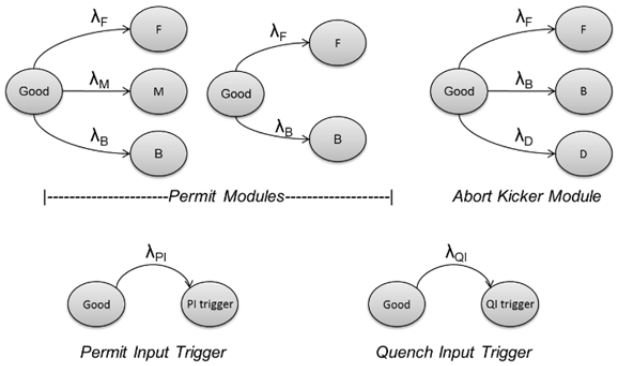


Figure 2: Markov diagrams

A competing risks model with crude lifetimes [1, 4] is implemented here, where multiple risks compete with each other to cause a final failure. Mathematically if a module is subjected to $j = \{1, 2...k\}$ risks, where $j = \{F, M, B\}$ for PMs, and $j = \{F, B, D\}$ for AKMs. The crude probability distribution function of risk j is given by

$$F_j(t) = \frac{\lambda_j}{\sum_{i=1}^{k}\lambda_i}\left(1 - e^{-\left(\sum_{i=1}^{k}\lambda_i\right)t}\right); j = \{1, 2.., k\}$$

The marginal hazard rate for j^{th} risk is λ_j and t is the time of observation or the beam store length. The survival function for the module undergoing competing risks is given by

$$S_T(t) = e^{-(\sum_{i=1}^{k}\lambda_i)t}$$

At any given instant for a module, following expression is always true

$$S_T(t) + \sum_{j=1}^{k} F_j(t) = 1 \quad (1)$$

ANALYTICAL MODEL

The BPS has a ring configuration. A trigger arrival at a module initiates a carrier failure, which returns back to it after traversing all the modules in the system. The functionality of BPS that affects the reliability is to abort the beams and dump the magnet power upon trigger arrival. We thus cut out this ring configuration to a linear structure, with inputs and outputs. The start of this structure is the master module [3] and the end is the AKMs that abort the beam. We approach the problem of developing mathematical equations for system states by looking at the states of each module in this system (Fig. 2). These states are of two kind: trigger state that initiates carrier failure (F, M) and passive state where a module waits in that state for a trigger (B, D, G). PI and QI are also the triggers to the system that initiate a carrier failure. For this reason we always use the failure density function *p(t)* (instantaneous probability) for triggering state and failure distribution function *P(t)* (probability that an event has occurred till now) for passive states to derive the expressions for the system level failures. For m^{th} module, and j^{th} triggering state where *j= {F, M, PI, QI}*

$$p_j^m(t) = \lambda_j e^{-(\sum_{i=1}^{k}\lambda_i)t}\ ;\ P_j^m(t) = \int_0^t p_j^m(t)$$

For jth passive failure state where *j= {B, D}*

$$P_j^m(t) = \frac{\lambda_j}{\sum_{i=1}^{k}\lambda_i}(1 - e^{-(\sum_{i=1}^{k}\lambda_i)t});$$

For passive good state

$$P_G^m(t) = e^{-(\sum_{i=1}^{k}\lambda_i)t}$$

Similar to Eq. 1, for a module *m*, at any given instant

$$P_G(t) + \sum_j \left(P_j^m(t)\right) = 1$$

Using above equations for modules' states, we further develop the analytical equations for the system states. BPS modules are connected by multiple links and some of the modules do not have blue/yellow link connected to them. We thus simplify the structure of BPS (shown in Fig. 1) for different type of system states. Also because AKM exhibit redundancy, we treat them as a separate entity that receive the signals from rest of the BPS. Consider the following configurations named I, II and III shown in Fig 3. Config. I shows the path of permit link triggers i.e. F and PI. Config. II shows the path of quench link triggers i.e. M and QI. Config. III shows the redundant configuration of the AKMs that signals the beam abort system to dump the beams.

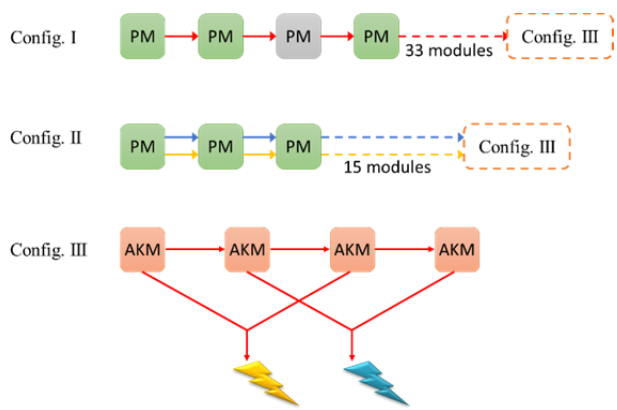


Figure 3: BPS connection diagram

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

To develop equations for system states, we consider the probability of each module and one by one traverse the path of carrier failure to beam abort system output. Because of the complexity and length of the expressions, we avoid writing them in this paper. However we explain the strategy that we chose to develop them. Following are the system states that can result from combinations of module states.

- ND: No Dump - No trigger generated in I, II and III
- GD: Good Dump - Trigger PI arrives in I, or QI arrives in II. No other triggers are generated anywhere else. Signal goes to the output of III, all modules in the forward path are in G state.
- FD: False Beam Abort Dump - Trigger F arrives in I or III. No other triggers are generated anywhere else. Signal goes to the output of III, all modules in the forward path are in G state.
- MD: False Magnet Quench Dump - Trigger M arrives in II. No other triggers are generated anywhere else. Signal goes to the output of III, all modules in the forward path are in G state.
- BD: Blind Dump - Trigger F/PI arrives in I, or M/QI arrives in II, or trigger F arrives in II. Signal does not to the outputs of III, at least one module in the forward path is in B state.
- DGD: Dirty Good Dump - Trigger PI arrives in I, or QI arrives in II. No other triggers are generated anywhere else. Signal goes the output of III, with at least one output passing through the D state of redundant AKMs.
- DFD: Dirty False Beam Abort Dump - Trigger F arrives in I or III. No other triggers are generated anywhere else. Signal goes the output of III, with at least one output passing through the D state of redundant AKMs.
- DMD: Dirty False Magnet Quench Dump - Trigger M arrives in II. No other triggers are generated anywhere else. Signal goes the output of III, with at least one output passing through the D state of redundant AKMs.

RESULTS

The expressions are solved in Mathematica [5] and the time dependent probabilities functions for each scenario are calculated. These probabilities are verified with the help of Monte Carlo simulation reported in [1]. The failure rates of the modules are obtained from [2]. Some of the module failure rates are very small, especially B and D, which reflect at system level failures. Also due to the AKM redundancy, the DGD, DFD and DMD failure probabilities are much smaller. The Monte Carlo simulation will need a large number of iterations to verify these results. Thus we first assign hypothetical failure rates to all the modules, with specific high failure rates for D mode to overcome the redundancy effect, and get substantial number of DGD, DFD and DMD states for the verification. Also we assume a hypothetical store length t = 0.232 hours. The Monte Carlo simulation is run for 2.4E9 iterations and takes about 40 hrs to generate results. The probability is expressed as the number of system scenario generated / total no of system runs. Table 1 compares these hypothetical results from the Monte Carlo and the analytical model (7 digit precision)

Table 1: Verification of the Analytical Model by Monte Carlo Results

Abbr.	Analytical	Monte Carlo
$P_{ND}(t)$	0.0149852	0.0149856
$P_{GD}(t)$	0.0621532	0.0621708
$P_{FD}(t)$	0.3105783	0.3105881
$P_{MD}(t)$	0.2494602	0.2494724
$P_{BD}(t)$	0.2754812	0.2754846
$P_{DGD}(t)$	0.0087564	0.0087507
$P_{DFD}(t)$	0.0406134	0.0406134
$P_{DMD}(t)$	0.0379719	0.0379183
Total (from model)	1.0000000	1.0000000

The individual probabilities of each scenario from the analytical model is very close to the probabilities obtained from the Monte Carlo. The exact sum of all the probabilities from the models is 1 as shown in the table. This verifies that the relative probability expressions for a model considers all possible states in which the BPS can go.

This establishes that the analytical model and the Monte Carlo model both are verified. Next we put the actual failure rates in the analytical model and obtain the actual failure probabilities of the all the system states, with the store length equal to 6 hrs as RHIC average store length.

$$P_{ND}(t) = 0.143573$$
$$P_{GD}(t) = 0.856193$$
$$P_{FD}(t) = 0.000123713$$
$$P_{MD}(t) = 0.000101377$$
$$P_{BD}(t) = 7.74551\,E-6$$
$$P_{DGD}(t) = 1.39145\,E-6$$
$$P_{DFD}(t) = 1.99945\,E-10$$
$$P_{DMD}(t) = 1.64755\,E-10$$

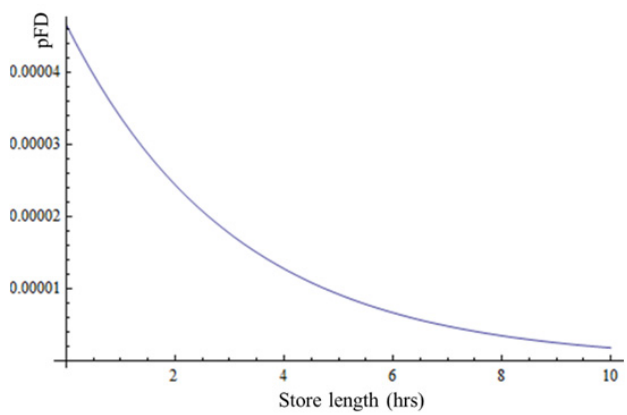


Figure 4: Probability density for FD system failure.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Figures 4, 5 and 6 show the graphs for the probability densities of three important system level failures (FD, MD and BD) plotted as a function of store length. The probability values above are cumulative values and can be calculated as the area under the curve up to a certain store length.

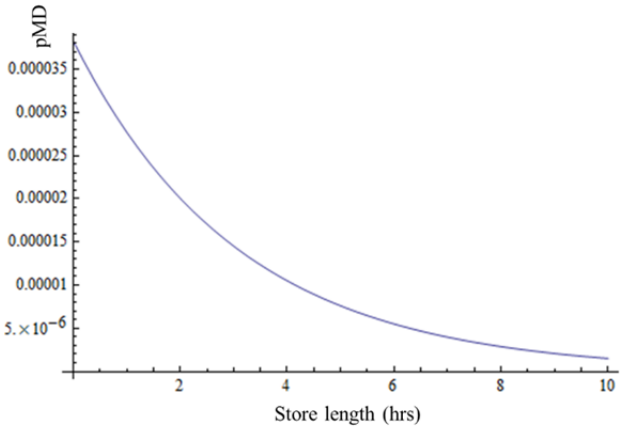


Figure 5: Probability density for MD system failure.

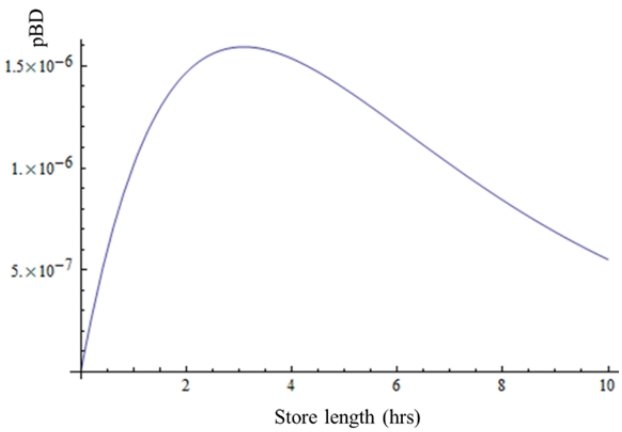


Figure 6: Probability density for BD system failure.

DISCUSSION

There are certain advantages of deducing the mathematical expressions for the probabilities of system level states. It facilitates quicker and easier analysis of the variation in system failure probabilities with change in component failure distributions or PI/QI trigger rates.

We utilize this to analyze the importance and interdependency of individual modules. The importance of the modules is evaluated as a combined effect of their failure rate magnitude and structural position. We analyze the three major system failure states, FD, MD and BD. The procedure is to increase the failure rates of F, M and B modes for each module one by one, and observing the change in the system failure probabilities, which is an indicator of the importance of the module. To summarize this comprehensive analysis, we can say that a module's importance is highly dependent on its failure rate. Alongside structural placement also plays a key role in determining the importance. The modules which are in the path of propagation of multiple failures have higher importance. The nearer a module is to the abort system output, higher is its importance. This is because probability of it being bypassed in failure propagation is small. The AKMs have lower importance due to the redundancy incorporated. Components that are major contributors to higher failure rate are discussed in [2].

To assess the interdependency between the modules, the modules are ordered according to their importance. After this we increase the failure rate for the subsequent module pairs, and observe the increase the system level failure probabilities. For no dependency, the system failure should have the same order for importance of modules for pairs [6]. We do not find any inter-dependency in the modules.

CONCLUSION

We develop a mathematical model for understanding the fine failure characteristics of the BPS. After the verification of this mathematical model through a MC model developed earlier, we are able to probe deeply into the failure probability distributions. This provides us with a faster way to analyze the system failures with change in system configuration. This model will provide a vision for the design of protection systems for the upcoming eRHIC project at BNL [7].

REFERENCES

[1] P. Chitnis et al., "A Monte Carlo simulation approach to the reliability modeling of the beam permit system of RHIC at BNL", ICALEPCS 2013, San Francisco

[2] P. Chitnis et al., "Quantitative fault tree analysis of the beam permit system elements of RHIC at BNL", ICALEPCS 2013, San Francisco

[3] C.R. Conkling, "RHIC Beam Permit and Quench Detection Communication System", Proceedings of the Particle Accelerator Conference, 1997

[4] M.J. Crowder, *Classical Competing Risks*, 2001, Chapman & Hall/CRC

[5] Wolfram Research Inc., Mathematica, Version 9, Champaign, IL (2014).

[6] S.T. Rachev, Professor of Finance, Stony Brook University, *Private Communication,* 2014

[7] V. Ptitsyn et al., "eRHIC, A Future Electron-Ion Collider at BNL", Proceedings of the 9th EPAC, Luceme, 2004

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

INTERLOCK SYSTEM FOR MACHINE PROTECTION AT THOMX ACCELERATOR

N. Elkamchi, P. Gauron, H. Monard, LAL, Orsay, France

Abstract

ThomX is a Compton based photons source. It aims to produce a compact and directional X-rays source, with high performance, high brightness and adjustable energy [1]. The principal application fields are medical sciences, social technology and industry. An interlock system has been implemented for machine protection, especially to protect sensitive and essential equipment (magnets, vacuum system, etc.) during machine operation. It will allow taking appropriate actions to minimize recovery time in case of operation problem. ThomX interlock system is based on Programmable Logic Controller (PLC-Siemens S7-1500) distributed over the machine subsystems, it collects default signals from the different equipment of the machine, up to the central PLC which kills the beam, by stopping the RF or the injection, in case of problem (bad vacuum, magnets overheating, etc.). Actually, the interlock system is under implementation in parallel with the assembly of the machine. It will allow accelerator to work safely, and increase the availability of the machine.

INTRODUCTION

ThomX is a compact light source project where the Compton Effect is used to produce X-rays. A circular accelerator is combined to a high power laser, amplified in an optical cavity, to obtain collisions between an electron beam and the light pulses.

The collision between electrons and photons induces a diffusion of photons in the electrons direction with a high energy gain. Thus, from a few eV photons it is possible to obtain "hard» X-ray (a few tens of keV) using a 50 MeV accelerator. The machine presents the advantage of a compact dimensions (70 m2) and low costs.

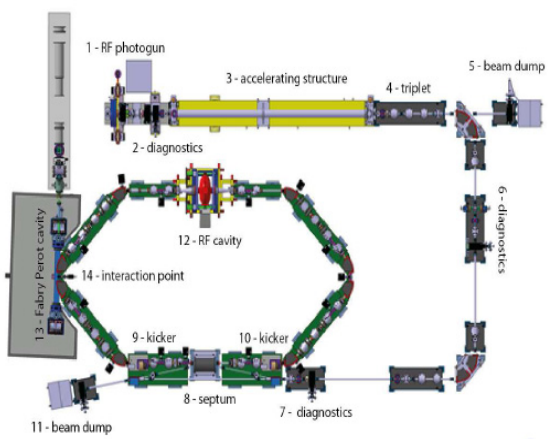


Figure 1: ThomX layout

Figure 1 shows the general layout of the machine. It is composed of four main systems: the injector, the storage ring, the laser and the Fabry-Pérot resonator.

The linac includes an electron RF photogun followed by an acceleration structure. The injection line consists of dipoles to steer the beam, quadruples and two diagnostic stations for energy spread measurements and beam characterization. The injection is performed by a septum and a fast kicker. [2].

The ring includes dipoles, quadruples, and sextuples for chromatic corrections. An RF cavity is used to bunch the electrons and to perform a longitudinal feedback.

The potential applications of this type of machine are numerous, including medical sciences, social technology and industry. Through collaboration between different French laboratories and industrials, it will be possible to share the technological excellence that will allow ThomX to be one of the most efficient machines in the world.

INTERLOCK SYSTEM DESCRIPTION

Because of the high beam power and energies in accelerator, the risk of beam induced damage is significant [3]. An interlock system is then used to protect the very sensitive and essential equipments (BPMs, vacuum chambers and valves, magnets) during machine operation. The ThomX interlock system is based on industrial programmable logic controller (PLC Siemens S7-1500 [4]). It collects default signals from the different equipment of the machine up to the central PLC which shut down the beam, by stopping the RF or the injection, when some parameters drifts outside specifications (bad vacuum, magnets overheating, etc.).

The interlock system consists of two levels. The first one is a local process, whose role is to monitor the variations of different parameters of the machine equipments, and generates a default signal in case of operation problem. The second level is the central PLC, which gathers and processes all the default signals from subsystems, and stops the RF power in a very short time.

To measure the performance of an interlock machine, we are based on two major parameters: reliability which measures the probability of fulfilling the major design function of the system, continuously and without interruptions, for a predefined period of time. The other parameter is the availability which defines the probability to find the machine fulfilling its major design function, when it is claimed to be in operation [5].

These two parameters describe the aptitude of the interlock system to meet the security requirements and to avoid failures.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ARCHITECTURE OF THOMX INTERLOCK SYSTEM

The interlock system is based on PLC technology (Siemens S7-1500). Low level (process level) modules are distributed around accelerators subsystems (figure 2). They are connected together to the master controller (interlock level) that collects the defaults signal and stops the RF by a redundant wired signal. The PLC of the interlock level is connected via Ethernet to the control interface (Panorama [6]) for default monitoring and signals reset. The architecture of ThomX interlock system is shown in figure2.

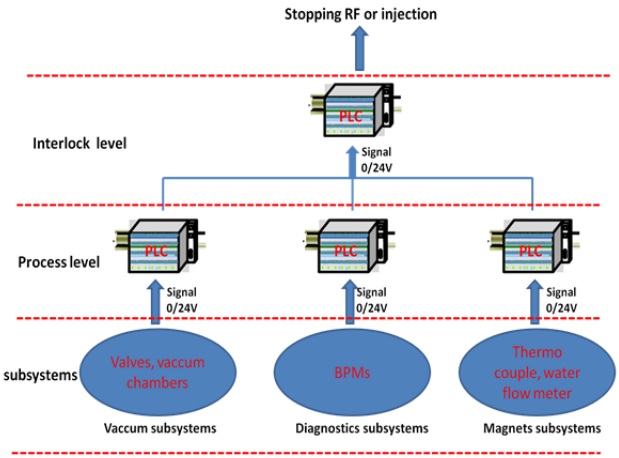


Figure 2: Interlock system architecture

The interlock system is designed basically to protect three sensitive area of the accelerator: magnets, BPMs and vacuum components.

Beam Position Interlock

A beam position monitor (BPM) allows detecting the beam position and assisting to maintain the beam path in beampipe axis. 17 BPM are distributed along the accelerator within 4 cells: storage ring cell 1 (6 BPMs), storage ring cell 2 (6 BPMs), Linac (1 BPM) and the transfer line (4 BPMs). All BPMs are driven by an electronic module (LIBERA [7]) which ensures signal acquisition and processing. Libera modules are connected to the process level PLC, it generate default signal when the parameters of the beam drift from specifications. The interlock signal is set within a millisecond whenever the electron beam position trespasses one of the predefined thresholds. This allows catching up the drift in short duration and avoiding damage to the machine.
The BPM interlock is an essential security component that protects with confidence the machine from orbits shifts.

Magnet Interlock

The magnet interlock system (figure3) is designed to protect magnets from overheating in case of failure of powering or cooling systems. It switches off the power supplies in a minimum time, and sends an error message to the control system to indicate the origin of failure. To ensure short time reaction, the magnet interlock system is a local process, and requires the intervention of a technician to rest the system in case of interlock outbreak.

The magnet security concern 12 dipoles distributed along three subsystems of the accelerator: 8 dipoles in the ring, 4 dipoles in the transfer line and 2 dipoles in the extraction line. To protect the dipoles from overheating, a set of thermocouples and water flow meters are installed in each dipole. when the threshold of the temperature or flow level is reached.

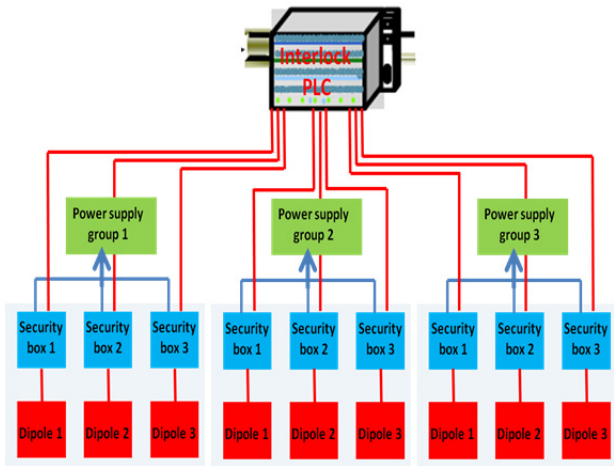


Figure 3: Architecture of magnet interlock

An electronic security box is connected to each dipole; it collects the information of water temperature and outflow and generates a default signal The security boxes of each group are linked in series (figure 3). In case of problem of temperature or water flow on a magnet, one default signal is sent to stop automatically the power supply of the corresponding group with a maximum delay less than 1 second. However all the security boxes are linked to the interlock PLC to indicate the defective magnet and to stop beam operation if needed.

Vacuum Interlock

The main objective of the vacuum interlock is to protect the machine equipments in case of bad or very bad vacuum. This process stops the beam when the vacuum threshold is reached, and allows also checking the positions of valves before authorizing the beam emission.
The accelerator beampipe is divided into several volumes bounded by automatic valves. Multiple ion pumps are installed on each volume. The pumps are connected in series and generate one interlock signal per volume when thresholds are reached for the corresponding volume. The vacuum thresholds are defined as follows: a normal vacuum/bad vacuum threshold around 10^{-7} mbar, and a very bad vacuum threshold corresponding to the disjunction of the ion pump power supply.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The vacuum security process is defined as follow:

- The bad vacuum (BV) threshold stops the beam emission; the operation of the modulator is locked to the threshold (BV) of the ion pump.
- The very bad vacuum (VBV) threshold of all ion pumps (between two valves) upstream and downstream of an automatic valve controls the closing of the valves. The valves can then only be opened after return of good vacuum of the relevant volume, and an open action from the control-command system.

The vacuum interlock system controls 28 default signals from the ion pumps, corresponding to two vacuum thresholds. It controls also the position of 11 automatic valves.

CONCLUSION

In this paper, we presented the operation principle and the architecture of ThomX interlock system, and its importance to avoid equipment damage. The system is actually under implementation in parallel with the assembly of ThomX machine. The future work includes the integration of the interlock system in the supervision interface of ThomX in order to unify the control tool and facilitate the default management. The test phase will start with the commissioning of ThomX, and will include essentially the verification of the response time and the reliability.

REFERENCES

[1] C. Bruni et al.,'ThomX - Conceptual Design Report', 2009, pp.1-136
[2] http://equipex-thomx.fr/
[3] C. Sibley, “Machine Protection Strategies for High Power Accelerators”, Proc. PAC03, pp. 607–611 (2003).
[4] Siemens PLC supplier, www.siemens.com
[5] R. Andersson, « Machine protection system and their impact on beam availibility and accelerator reliability » Proc. IPAC 15
[6] Panorama HMI; uk.codra.net
[7] Libera supplier; www.i-tech.si

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

BUILDING AN INTERLOCK: COMPARISON OF TECHNOLOGIES FOR CONSTRUCTING SAFETY INTERLOCKS

T. Hakulinen, F. Havart, P. Ninin, F. Valentini
CERN, Geneva, Switzerland

Abstract

Interlocks are an important feature of both personnel and machine protection systems for mitigating risks inherent in operation of dangerous equipment. The purpose of an interlock is to secure specific equipment or entire systems under well defined conditions in order to prevent accidents from happening. Depending on specific requirements for the level of reliability, availability, speed, and cost of the interlock, various technologies are available. We discuss different approaches, in particular in the context of personnel safety systems, which have been built or tested at CERN during the last few years. Technologies discussed include examples of programmable devices, PLCs and FPGAs, as well as wired logic based on relays and special logic cards.

INTRODUCTION

There are three basic types of components in safety systems: *sensors* for collecting data on any measurable conditions important for safety, *actuators* for manipulating equipment important for safety when necessary, and *interlocks*, for computing the safety logic between the two. Sensors and actuators are always particular to the application in question, that is, the conditions to be surveyed and actions to be taken are different in each case. However, an interlock is just a unit for processing of logical information, and the technological choices for its implementation have usually more to do with requirements of overall throughput, reliability, cost, or technological diversity as mandated by principles of safety system design.

Safety interlocks are considered critical components that are subject to careful implementation and certification. International standards, such as IEC 61508 and 61511 for process industry, and IEC 61513 [1] for nuclear industry are often used in the design of entire safety systems, including the interlock. IEC 61508 and 61511 use the concept of safety integrity level (SIL) to quantify the required level of reliability of the safety system in its safety function as well as the ability of the system components to satisfy that requirement.

CERN GS/ASE group is responsible for all personnel safety systems, access control systems, and personnel safety alarm systems at CERN. We have a long experience of designing, building, and operating different personnel safety systems for CERN accelerators and experiments. Over the years, different approaches have been used for building interlocks for these systems, and this has given us some insight into the relative merits of the various technological choices. The technologies discussed in this paper are programmable logic controllers (PLCs), relay-based logic, wired logic with dedicated logic cards, and field-programmable gate arrays (FPGAs) with the examples based on actual implemented equipment.

PLC

Programmable logic controllers (PLCs) are the mainstay of modern process control applications. A PLC operates in a cyclic manner: during one cycle all inputs are collected, a new system state is computed in the CPU according to the program logic, and all outputs are set correspondingly. Cycle times of PLCs vary normally from a few to hundreds of milliseconds depending on the number of I/Os and the complexity of the program. I/Os are normally handled by separate modules, which can be placed some distance away from the CPU when connected via copper or fiber-optic cabling using a specific fieldbus protocol such as Profibus or Profinet [2]. Modern PLCs are also able to communicate via standard TCP/IP protocols via Ethernet, facilitating long-distance supervision of these systems. Special Supervisory Control and Data Acquisition (SCADA) systems are often used to integrate PLCs into a larger control framework.

Several manufacturers offer safety-related components certified for use in SIL-rated systems [3,4]. As a PLC-based system cannot be certified higher than SIL 3 (according to the standard, no system containing program code can be at SIL 4), that is normally the highest level of certification for the safety-related components as well. Some PLCs are able to integrate safety and non-safety-related programs and components within a single system, making communication between the two parts seamless.

At CERN, various generations of Siemens PLCs (S5, S7) are used in our personnel safety systems: LHC Access Safety System (LASS) [5], PS Access Safety System (PASS) [6], SPS Personnel Protection System, SPS Primary Ion Interlock [7]. This technology is fully mastered at CERN, and Siemens product life cycles are long allowing for a long utilization of installed hardware. Logic modifications and testing of PLC systems is fairly easy thanks largely to an integrated programming environment. Furthermore, PLC safety signatures provide a safeguard against unauthorized modifications of PLC code.

However, there are also drawbacks to PLC technology: For a large system, powerful CPUs are often needed to manage the logic coupled with sophisticated SCADA

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

systems, which can get quite expensive in hardware and license costs. Also, the entire system environment can get fairly complicated, where considerable expertise is required to manage different generations of hardware, firmware, and software for everything to work correctly together. Upgrading any of the system components, including patching firmware and software even for security reasons can be a complicated and risky affair, which will almost inevitably incur a period of unavailability of the entire system. Furthermore, changing safety system components in any way normally triggers a full suite of validation tests by the responsible safety officer, often requiring days to prepare and carry out, which further limits the usefulness of the otherwise easy programmability of PLC systems in production environments. Figure 1 shows one of PASS PLCs with its I/O modules.

Figure 1: Siemens S7 400 series CPU with ET200 remote I/O modules.

RELAY-BASED LOGIC

Possibly the oldest and still a highly relevant method of building reliable hard-wired logic systems is to use relays. All basic logic gates can be implemented in simple relays with hardwired connections. An example of a relay-based AND-gate is shown in Figure 2.

Implementation of a relay-based interlock is quite straightforward: standard electrical wiring, connectors, and relays are installed in a rack. Special safety relays are often used, which have high MTBFs and known fail-safe states. Generally relay-based systems are quite resistant to external disturbances, with the exception of high magnetic fields in rare cases, which can cause a magnetic relay to misbehave. Also any competent electrician is able to understand, maintain, and modify such a system based on simple circuit diagrams.

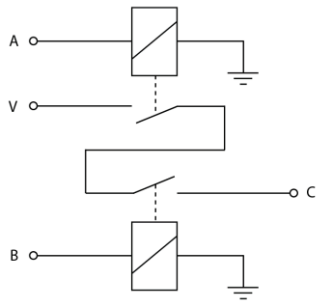


Figure 2: A relay-based AND-gate. A and B are the inputs, C is the output, and V is a constant voltage.

However, relay-based systems have several serious drawbacks compared to more modern approaches: Relay-based systems are bulky requiring plenty of rack space even for a small amount of logic, which limits the practical complexity of the application. Safety relays are expensive and implementation of the system is quite labour intensive. If supervision of interlock functions is required, extra logic needs to be implemented. Modifying the interlock logic can also obviously get quite hard.

As a relay-based system is necessarily built and designed in-house, its certification at a specific SIL-rating, if required, could be difficult requiring a calculation based on the individual SIL-rated components. Relays themselves may exhibit wear and tear over time: contact issues due to oxidation, sulphurisation, or arcing may happen. The upside is that in such a case, changing the defective component is relatively easy.

Relays are also not suitable for applications, where the switching frequency is high, since normal mechanical relays are typically only rated for around 1-2 million switches during their lifetime. Also, switching time of a mechanical relay is somewhere between 0.1-10ms and a certain resolution time is in any case required for the state change due to effects like contact bounce. Using solid-state relays could help in these cases, though.

Figure 3: Hardwired relay-based logic using relays. Green LEDs indicate contact states.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

For these reasons, relay-based systems are realistically only suitable for fairly straightforward and small-scale logic. At CERN, they have mainly been used to build redundant chains of the most important safety functions of the LHC and PS access safety systems. Figure 3 shows part of the relay logic installed in the PASS cabled loop rack.

DEDICATED LOGIC CARDS

Another way of constructing a hard-wired interlock is by using dedicated electronic logic cards such as HIMA Planar4 series [4]. Logic gates are implemented in standard modular sub-rack-mounted cards and interconnections between these gates are realized by wiring the card inputs and outputs on the sub-rack backplane either by soldering or wrapping. A faulty card can easily be exchanged without having to touch the logic at all.

These kinds of systems can be used to build the highest rated safety systems. Most components of the Planar4 series are certified for use in SIL 4 systems, and consequently, they are often used in people transport systems or for critical process safety at chemical plants, oil rigs, and the like. Various I/O modules based on relays or line-monitored connections are available, and all gates in the sub-rack can be supervised by dedicated communication modules offering Profibus, Modbus, or OPC connectivity to an outside PLC or supervision post. Figure 4 shows the modules of the wired chain of CERN SPS North Area primary ion interlock [7] designed using HIMA Planar4.

Figure 4: Planar4 wired logic in a 19-inch sub-rack. From the left: fuse module, two timing modules, logic modules, and far right a Profibus supervision module.

One of the drawbacks of this technology is that due to a safety-related design and the additional diagnostic functions on each card, switching times are relatively long for active gates (AND, NOT), ranging from 2-15ms depending on the case (see Figure 5). Therefore, complicated logic may introduce a considerable delay of up to tens or hundreds of milliseconds. This may not be a problem in most applications, but should it be necessary to optimize this, careful design is required. For example, by application of De Morgan's theorem it is usually possible to construct the logic by using primarily OR-gates, which incur no internal processing delay. The HIMA wired chain of the SPS North Area primary ion interlock was designed this way [7]. The downside of this approach is that the logic becomes harder to understand from the diagrams.

HIMA provides logic cards for the basic logic operations as well as a variety of I/O options. However, any more complicated logic components, such as latches, flip-flops, etc. would have to be constructed from the basic gates. HIMA also does not provide 2-channel complementary (ambivalent) I/O, for which logic would also have to be custom built, if needed. As the logic programming is carried out with a soldering iron or a wrapper, changing this logic is not straightforward. Even small modifications may require unsoldering of a considerable number of connections to gain access, which is obviously both time consuming and error prone.

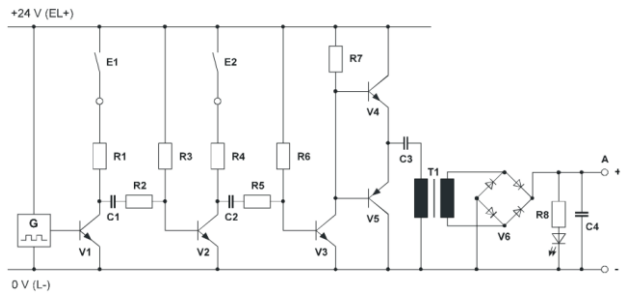


Figure 5: HIMA Planar4 AND-gate. E1 and E1 are the inputs and A is the output. The internal design is based on dynamic signaling driven by signal generator G. A simultaneous failure of up to three separate components leads to the output being de-energized. Compared to Figure 2, the added complexity due to safety-related design is clear.

FPGA

Designs requiring very fast logic processing have since long time used Application Specific Integrated Circuits (ASIC) or Field Programmable Gate Arrays (FPGA). An FPGA is an integrated circuit that can be programmed at the gate level to perform desired operations. In contrast to an ASIC, an FPGA can be reprogrammed in field if necessary. Both ASICs and FPGAs can be designed using a specific hardware description language (HDL). FPGA system vendors often also provide block-based graphical programming environments as well as bindings to conventional programming languages, like C. A schematic program is compiled into an array of gates on the circuit, which will realize the desired logic.

The great advantage of FPGAs over general-purpose computers and PLCs is speed. Response times of implemented functions can be in the nanosecond range, obviously depending on their complexity. Testing and modifying logic is as easy as reprogramming a PLC.

FPGAs are normally run within an integrated framework, such as National Instruments cRIO 903x series [8], where the FPGA is controlled by a real-time RT-Linux system, providing connectivity to network and peripherals similar to a normal PC. The FPGA connects directly to its own I/O modules, of which there are many available for different applications. Supervision of the system is easily implemented via the RT-Linux unit.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Table 1: Comparison of Some of the Most Important Metrics Between the Different Interlock Technologies

	Certification	Time scale	Communication	Supervision	Logic changes	Logic implementation	Space requirements	Scalability
PLC	Up to SIL 3	ms	TCP/IP Profibus	SCADA Custom	Easy	Programming	Medium	Good Large scale
Relay logic	None	ms	Wired	Custom	Hard	Manual Hard-wired	High	Limited Small scale
Logic cards	Up to SIL 4	ms	TCP/IP Profibus	SCADA Custom	Hard	Manual Hard-wired	Medium	Limited Medium scale
FPGA	None yet	ns	TCP/IP Profibus	SCADA Custom	Easy	Programming	Small	Good Large scale

The most important drawbacks when considering FPGA-based solutions for safety systems is that they are usually not safety-related by design in the same way as safety PLC systems or HIMA logic cards. While calculations can be made based on the published MTBFs of individual system components, modules explicitly certified for use in SIL-rated systems have not been available. However, this is likely to change in the future, as vendors will be introducing safety-related modules for their systems.

At CERN, a pilot implementation of a small independent safety interlock using a National Instruments FPGA is being studied [9]. One of the aims of this project is to gain experience and confidence in the technology as well as to test the auxiliary functionality provided by the RT-Linux system in running related access control functions, badge readers, touch panels, etc. The test bench FPGA system of this project is shown in Figure 6.

Figure 6: National Instruments cRIO 9030 FPGA controller test bench. The RT-Linux unit is on the left with Ethernet, serial, USB, and video connections. The FPGA unit is on the right with four I/O modules, of which two connected. Each I/O module of this type hosts 32 digital signals.

CONCLUSIONS

We have compared four different technologies for constructing safety interlocks based on our return of experience over years in designing safety systems. The technologies compared were PLCs, relay-based logic, dedicated logic cards, and FPGAs. All of them have their pros and cons and, indeed, depending on the application, their proper place in the toolbox of an interlock builder: When strictly SIL-certified systems following norms IEC 61508 and 61511 are required, PLCs or dedicated logic cards remain the preferred choices. However, for building diverse and rendundant safety systems following norm IEC 61513, relay-based logic and FPGAs can also be very useful. Table 1 presents a synthesis of some of the most important performance metrics potentially affecting the choice of technology for a new interlock system.

REFERENCES

[1] http://www.iec.ch
[2] http://www.profibus.com
[3] http://www.siemens.com
[4] http://www.hima.com
[5] T. Ladzinski et al., "The LHC Access System," ICALEPCS09, Kobe, Japan, WEP102, p. 600 (2009); http://www.JACoW.org
[6] P. Ninin et al., "Refurbishing of the CERN PS Complex Personnel Protection System," ICALEPCS13, San Francisco, USA, MOPPC059, p. 234 (2013); http://www.JACoW.org
[7] T. Hakulinen et al., "Personnel Protection of the CERN SPS North Hall in Fixed Target Primary Ion Mode," ICALEPCS13, San Francisco, USA, MOPPC067, p. 66 (2013); http://www.JACoW.org
[8] http://www.ni.com
[9] F. Valentini et al., "Integration of Heterogeneous Access Control Functionalities Using the New Generation of NI cRIO 903x Controllers," MOPGF143, ICALEPCS15, Melbourne, Australia (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DESIGN OF FAST MACHINE PROTECTION SYSTEM FOR THE C-ADS INJECTION I

Fang Liu, Jun Hu, XiaoShan Jiang, Qiang Ye, IHEP, Beijing, China
Guanghua Gong, Tsinghua University, Beijing, China

Abstract

In this paper a new fast machine protection system is proposed. This system is designed for the injection I of C-ADS which fault reaction time requires less than 20 us, and the one minute down time requires less than 7 times in a whole year. The system consist of one highly reliable control network based on a control board and some front IO sub-boards, and one nanosecond precision timing system using white rabbit protocol. The control board and front IO sub-board are redundant separately. The structure of the communication network is a combination structure of star and tree types which using the 2.5 GHz optical fiber links the all nodes. This paper pioneered the use of nanosecond timing system based on the white rabbit protocol to determine the time and sequence of each system failure. Another advantage of the design is that it uses standard FMC and an easy extension structure which made the design is easy to use in a large accelerator.

INTRODUCTION

The injection I of ADS (Accelerator Driven Sub-critical System) is being constructed in institute of high energy physics, China [1]. How to ensure the safe and stable operation of the injection, the fast protection system has played a crucial role. The protection system needs to make a quick decision on the current failure. When a general failure, the system needs to quickly adjust the system without stopping and set the working state to the normal mode. When a serious fault, it is necessary to quickly send out a signal and shutdown the whole system. The system also cans storage the statement before and after the fault. Through the storage of information the staff can quickly rule out the reason of occurrence failure [2].

FAST MACHINE PROTECTION SYSTEM FRAMEWORK

ADS fast protection system is composed of main control board, some front IO sub-boards, supervisory computer and one nanosecond precision timing system. The system framework is shown in Fig. 1.

Front IO sub-board monitors the 24V or 5V switch signal from the LLRF, PS, Vacuum, RFQ and other critical points. When the monitored points occur error, it sends the error signal to the main control board through optical fiber. The signal contains the time information that is labelled by the White Rabbit system. The main control board analyses and judgments the error through the stored fault tree. Then, it issued the processing results to the front IO sub-boards. Meanwhile, the main control board

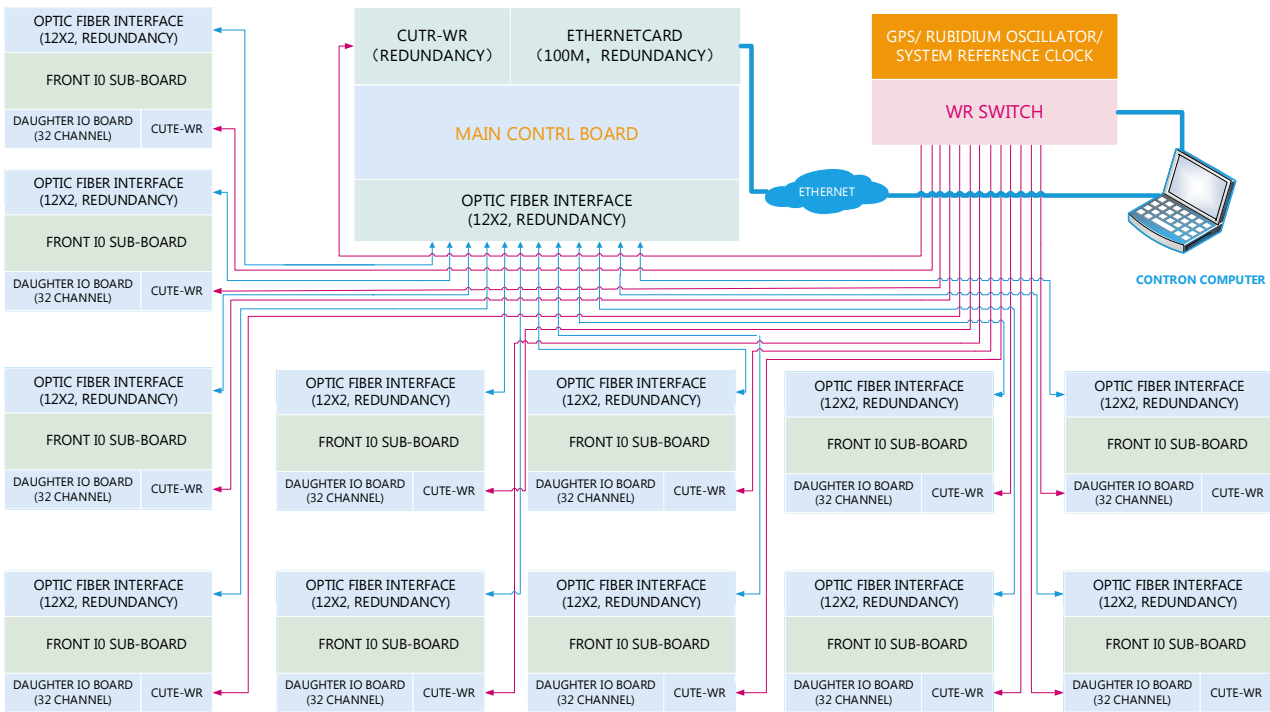


Figure 1: System Framework.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

record the error, process the result and send it to the supervisory computer.

The whole fault reaction time requires less than 20us, including the monitor reaction time, transmission time and processing time.

The framework of the network is a star and tree combination structure. A main control board can control 12 block IO sub-board, and an IO sub-board can manage 32 inputs and 32 outputs separately. It will form a 384 nodes network. The framework shows in Fig. 2. When we need a greater network, we can link tow showed network (Fig.2 showed). It can reach 4068 nodes. If we want to a much bigger one, we can cascade again. So it is very suitable for Multi-node large accelerator. Another advantage of the framework is it save the signal transition time cost in each node. It just need through three nodes from one node to another node.

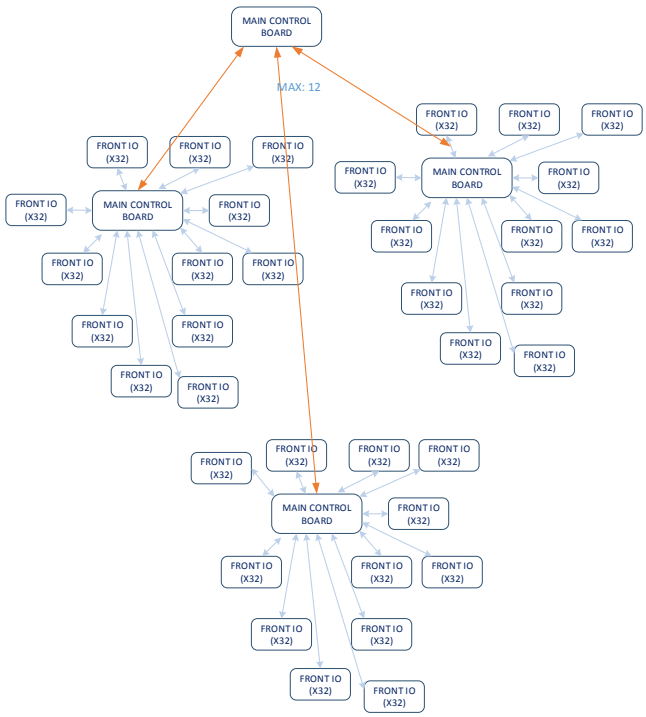


Figure 2: Network framework.

FRONT IO SUB-BOARD

The front IO sub-board use a 2.5 U box, each board can be placed independently. The input voltage is 220V alternating current. The Fig. 3 is a schematic diagram of the front IO sub-board. The front IO sub-board uses the motherboard and daughterboard design. The motherboard is responsible for power processing, logical processing, cache and data transmission. The daughter board is responsible for the acquisition and transmission of the front-end switch quantity.

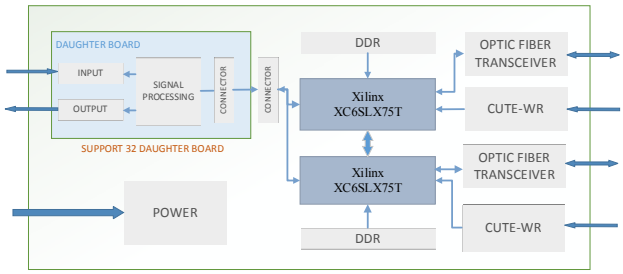


Figure 3: Schematic diagram of the front IO sub-board.

Mother Board

The mother board uses XC6SLX75T-3FGG676I FPGA as the main processor, and communicates with 32 daughterboard through 32 cables. We use the embedded Rocket IO module in FPGA to implement the optic fiber transceiver [3]. The transmission protocol is Aurora 8B/10B. The line rate is 2.5 Gbps. FPGA logic is responsible for the packing and unpacking of parallel data. At the same time, the FMC interface is used to connect the White Rabbit's timing system. And the DDR2 1 GB is used to cache the data.

In order to improve the reliability, the motherboard uses the method of watchdog, redundancy FPGA, DDR and optic fiber transceivers. The reliability of transmission between motherboard and daughterboard is guaranteed by the link showed in Fig. 4. Normal work, the master FPGA on motherboard receive data from 4 and send data to daughterboard from 3, the slave FPGA will also receive the same data from the daughterboard through 7 as a copy one and receive the feedback data from daughterboard through 8 as the verification of the master FPGA send data success. At the same time，the master and slave FPGA will check the work status for each other, and timely send control state to the back-end processing module.

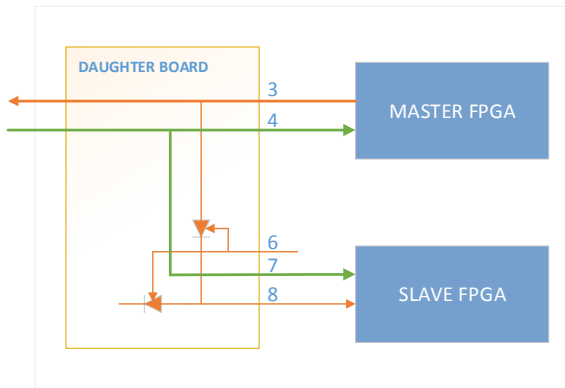


Figure 4: The link between motherboard and daughterboard.

Daughter Board

In order to ensure the link between the fast protection system and monitored system, we design three type boards: 24V optocoupler isolation board, 5V optocoupler isolation board and optical transmission board. The line

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

rate of each board requires not less than 3Mbps. The board logic is defined as follows:

- ***24V optocoupler isolation board:*** level greater than 12V as normal, as the error level is less than 5V.
- ***5V optocoupler isolation board:*** level greater than 3.5V as normal, as the error level is less than 0.8V.
- ***Optical transmission board:*** according to the optical fiber signal definition "1" as normal, "0" as an error.

The 24V optocoupler isolation board uses TND316S chip as a driver, the 5V optocoupler isolation board uses SN54AC245. The optocoupler chip is FOD8001. The fiber optical transmitter uses HFBR-1414TZ and HFBR-2412TZ. When the signal is transmitted over a long distance or high voltage, the signal will be transmitted through the optical fiber.

MAIN CONTROL BOARD

The main board also use a 2.5U box. The function is receiving data from front IO sub-board, judging and processing the data, sending control command and storing system useful information. The Fig. 5 is a schematic diagram of the main control board.

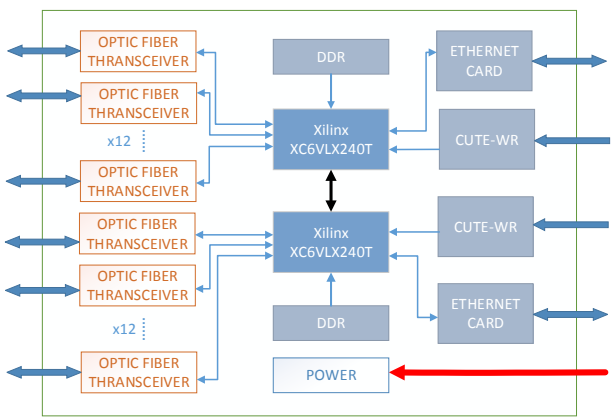


Figure 5: Schematic diagram of the main control board.

The main control board uses Xilinx XC6VLX240T FPGA. The board can receives12 channel signals from the front IO sub-board. It will complete the monitor, analysis, packing and feedback in the FPGA. The status log information will regularly sent to upper computer by a 100Mbps Ethernet sub-card using a FMC interface.

The types of optic fiber transceiver, the transmission protocol, DDR, White Rabbit module are all as same as the front IO sub-board.

In order to improve the reliability, the main control board also uses redundancy design including FPGA, optic fiber transceiver, Ethernet and White Rabbit. The double FPGA will timely check the status of each other. The status includes voltage and temperature information, the current FPGA state and the logic confirm. When we detect the master FPGA in an abnormal mode, the slave FPGA will automatically take over the function of the master FPGA and report the error.

WHITE RABBIT SYSTEM

The White Rabbit timing system is a less than 1ns accuracy, 10ps precision and suitable for long-distance transmission timing system. It can receive GPS, rubidium oscillator and reference clock signal [4].The framework of ADS injection I White Rabbit timing system is shown in Fig. 6.

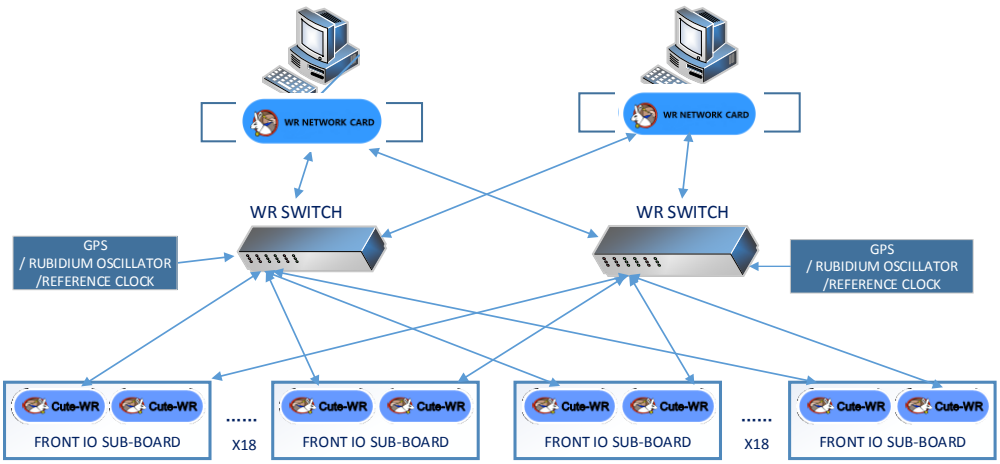


Figure 6: Framework of White Rabbit timing system.

The network is mainly composed of a white rabbit switch, some front end Cute-WR modules and a white rabbit protocol network card that responsible for the linking to computer. According to the final construction requirement, the switch can choice GPS, rubidium oscillator or the accelerator timing reference clock. The Cute-WR module is the terminal module of the system. It can receive clock and time signal from the switch. A switch can link 18 Cute-WR modules. For this system, we use 13 Cute-WR modules, 12 of which are used in the link of 12 front IO sub-boards, and the other for main control board. The entire WR network uses backup design. It guaranteed the system reliability.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

CONCLUSION

According to the reliability and safety requirements, this paper presents a new scheme of accelerator machine protection system. It can make the protection action time less than 20 us and it use sub-nanosecond timing system as the clock and time module, which make it possible to determine the errors occur time and the error sequence. And each node of the system is an independent and easy organized one. It made the system with good integration and scalability.

REFERENCES

[1] Jingyu Tang and Zhihui Li (ed.), Conceptual physics design of the C-ADS accelerators, IHEP-CADS Report/2012-01E.

[2] C.Sibley, "Machine Protection Strategies for high power accelerators", IPAC2003, Portland, USA, p.607 (2003).

[3] Xu Wenbo and Tian Yun， Xilinx FPGA：Development and Application (second edition).

[4] Open hardware repository, http://www.ohwr.org/projects/white-rabbit

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

UPGRADE OF THE TRIGGER SYNCHRONISATION AND DISTRIBUTION SYSTEM OF THE BEAM DUMPING SYSTEM OF THE LARGE HADRON COLLIDER

N. Magnin, A. Antoine, E. Carlier, V. Chareyre, S. Gabourin, A. Patsouli, N. Voumard,
CERN, Geneva, Switzerland

Abstract

Various upgrades were performed on the Large Hadron Collider (LHC) Beam Dumping System (LBDS) during Long Shutdown 1 (LS1) at CERN, in particular to the Trigger Synchronisation and Distribution System (TSDS): A redundant direct connection from the LHC Beam Interlock System to the re-trigger lines of the LBDS was implemented, a fully redundant powering architecture was set up, and new Trigger Synchronisation Unit cards were deployed over two separate crates instead of one. These hardware changes implied the adaptation of the State Control and Surveillance System and an improvement of the monitoring and diagnosis systems, like the various Internal Post Operation Check (IPOC) systems that ensure that, after every beam dump event, the LBDS worked as expected and is 'as good as new' for the next LHC beam. This paper summarises the changes performed on the TSDS during LS1, highlights the upgrade of the IPOC systems and presents the problems encountered during the commissioning of TSDS before the LHC Run II.

INTRODUCTION

LHC Beam Dumping System

The Large Hadron Collider (LHC) Beam Dumping System (LBDS) is responsible for the single turn extraction of the beam from LHC rings, after reception of a Dump Request (DR). It is composed, for each beam, of 15 extraction kicker magnets (MKD) followed by 15 extraction septum magnets (MSD) to extract the beam from the ring, followed by 10 dilution kicker magnets (MKB) in the extraction channel to spread the beam over the dump target (TDE) surface [1].

Each MKD and MKB is powered by a High-Voltage Pulsed Generator (HVPG), to produce the fast high current pulse needed to extract and dilute the beam, upon reception of a trigger.

Trigger Synchronisation and Distribution System

Within the LBDS control architecture, the Trigger Synchronisation and Distribution System (TSDS) is responsible for the detection of a DR from various sources, the subsequent generation of a dump trigger synchronised with the beam-free abort gap of the LHC, and the distribution of this Synchronous Beam Dump Trigger (SBDT) to the 25 HVPG of the 15 MKD and 10 MKB [2]. The TSDS is built around a major component, the Trigger Synchronisation Unit (TSU), which detects the DR from sources like the Beam Interlock System (BIS) [3], the State Control and Surveillance System (SCSS) [4], the Beam Loss Monitor Direct Dump (BLMDD), and issues a dump trigger synchronised with the Beam Revolution Frequency (BRF) signal [5].

The TSDS also includes an asynchronous Re-Triggering System (RTS) to cover the cases were the SBDT is not distributed properly, or when a MKD HVPG self-triggers [2].

During the LHC Run I operation, all beam dump requests were properly detected and distributed. Nevertheless some weaknesses of the TSDS were pointed out during various reviews and discussions, and a consolidation project was planned for Long Shutdown 1 (LS1).

Re-Trigger System Principle

The RTS is entirely passive and is composed of two redundant floating Re-Trigger Lines (RTL). A simplified view of one RTL is shown in Fig. 1. The 25 HVPG are interconnected by a chain of Re-Trigger Boxes (RTB), connected serially using a twisted pair cable. These RTB are composed of simple insulation transformers and decoupling diodes. When an HVPG triggers, five internal pickups in the HVPG will inject pulses in the RTL through five input transformers of the RTB. These pulses are then transmitted to all the other HVPG through two output transformers in each RTB. Thanks to this mechanism, in case of self-trigger of one MKD HVPG, all the other HVPG of LBDS will be triggered automatically as quickly as possible. Only the 15 MKD HVPG will inject pulses in the two RTL, as MKD see the circulating beam, and a self-trigger of one MKD HVPG requires retriggering all LBDS HVPG. The MKB HVPG do not inject pulses in the two RTL, because MKB do not impact the circulating beam, but a self-trigger of MKB HVPG will be detected by the various surveillance systems and a normal SBDT will be issued.

To cover a possible failure of the distribution of the SBDT issued by the TSU, an asynchronous dump trigger is also systematically generated by the TSU at the time it detects a DR. As up to 90 µs (one beam revolution) may be necessary for the TSU to issue a synchronous dump, the asynchronous dump trigger is delayed by 250 µs and injected in the RTL using a Trigger Delay Unit of 250 µs (TDU-250), as can be seen in Fig. 1.

The upgrades performed on the various TSDS subsystems, along with their motivations, are detailed in the following sections.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

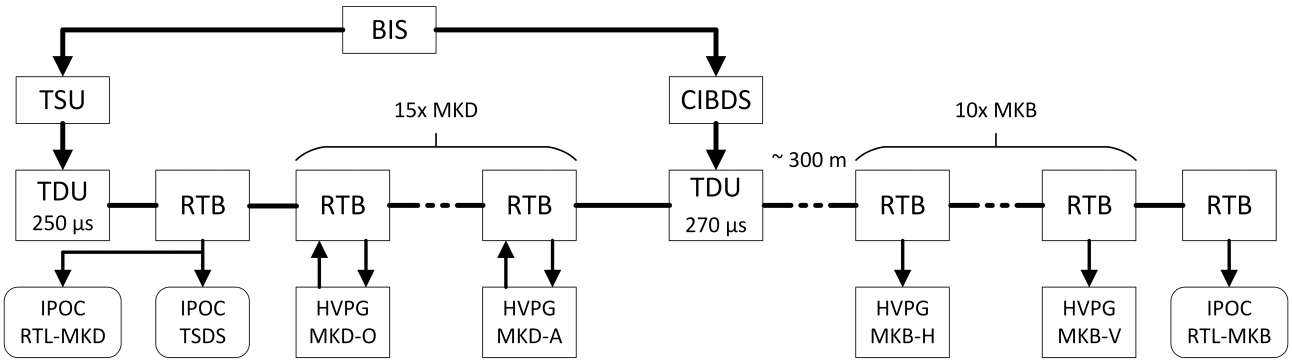


Figure 1: Simplified view of one Re-Trigger Line.

UPGRADES TO TSDS

Improvement of Power Distribution Architecture

During LHC operation in 2012, we faced an unexpected event: One Power Supply Unit (PSU) of one of the Front-End Computers (FEC) located in the control electronics racks of the LBDS failed, causing a short-circuit on its main input. The circuit breaker located in the distribution bar for the electronics rack did not react quickly enough, and the short circuit propagated up to the general circuit breaker in the UPS distribution powering the LBDS. This resulted in eight racks of electronics losing their UPS, which was totally unexpected for such a simple FEC PSU failure. The LBDS nonetheless properly executed a synchronised beam dump, but some of the diagnosis systems were not available for post-operation analysis. An internal review of LBDS powering was conducted later in 2012, when recommendations for a consolidation of power distribution during LS1 were given.

To avoid such a situation in the future, we provided a distribution box to power each crate PSU through an individual circuit breaker, as depicted in Fig. 2.

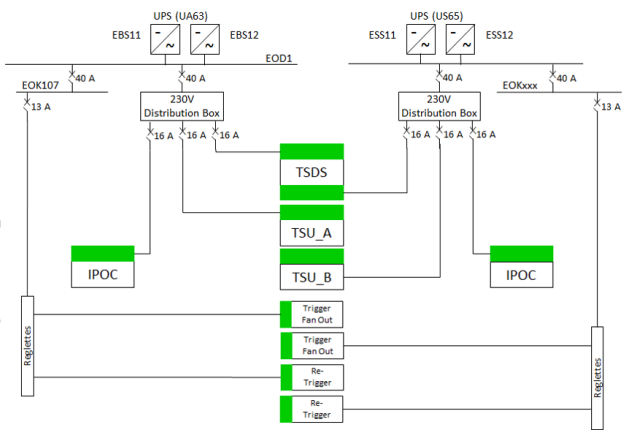


Figure 2: Full redundant electrical distribution.

During LS1, a new dedicated direct connection to a second UPS was also provided for LBDS, so now two independent UPS, one located in UA63, the other in US65, power the various LBDS redundant PSU, see in Fig. 2.

The UPS system was also upgraded during LS1, and a new fully redundant architecture was implemented. As illustrated by Fig. 3, a third 'backup' UPS unit was installed to feed the bypass line of the first two UPS units. Hence, in case of failure of one of the first UPS units, the backup unit guarantees the redundancy of the UPS distribution paths.

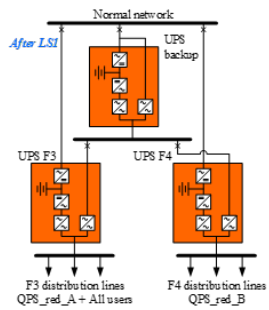


Figure 3: Full redundant UPS configuration.

All the FEC housing critical post-operation diagnosis systems were equipped with two redundant PSU, each connected to a different UPS. The monitoring of the state of all the redundant PSU of the LBDS crates is now performed remotely through Ethernet, so a software interlock system can request a beam dump in case a failure is detected in a PSU.

Before the startup of LHC Run I, a simulation of UPS failure was performed for LBDS. Each of the two UPS was disconnected, in two successive tests. Some non-conformities were detected, such as two redundant PSU of a FEC connected to a same UPS. These problems were fixed, but the UPS test proved to be useful and subsequently shall be planned after every intervention on the LBDS power distribution.

New Deployment of TSU Cards

During the LHC Run I operation, a new failure mode of TSDS was identified: Due to the deployment of both TSU cards in the same TSDS VME crate, along with the BRF optical receiver card and the BLMDD interface card, if only the +12V VME power is lost, neither synchronous nor asynchronous triggers are issued. This power failure was not detectable and would have caused the LBDS to not

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

function, which is totally unacceptable. This failure never occurred, but at the time the failure mode was identified, we immediately requested a beam dump and stopped the LHC operation until a solution was found. The same day, we implemented a surveillance of the +12V power supply of the TSDS VME crate, to inject an asynchronous trigger on the RTL in case a failure is detected.

In order to definitively remove this +12V common failure mode, a new deployment of the two TSU cards was performed during LS1. Each TSU is now installed in its own VME crate with a redundant PSU, and a third TSDS VME crate contains the RF optical receiver card and the BLMDD interface card, as can be seen in Fig. 2.

Design of a New TSU Card

Following an external review of the TSU card design performed in 2010 during which recommendations for minor hardware improvements were made, and due to the new deployment of TSU cards over two separate crates, a new TSU card was developed.

A new cabled interconnection between the two boards was implemented to replace the common VME backplane. A surveillance of all internal voltages was added to the TSU card itself, and a DR is issued to the redundant card in case any power supply failure is detected. Additionally, an internal continuous surveillance of the CRC of the TSU programmable logic circuits (FPGA) was implemented, so in case of a Single Event Upset corruption of one of the programmable circuits, an incorrect CRC is detected and a DR is issued to the redundant TSU.

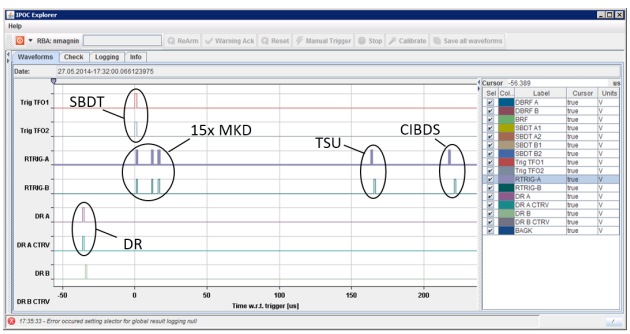


Figure 4: View of some signals captured by IPOC-TSDS.

Numerous on-board post-operation diagnosis functionalities have been added, such as the surveillance of the output current of the SBDT signals. Additionally, 32 TSU internal signals are now acquired and analysed by an Internal Post Operational Check (IPOC) system [6], such as all the redundant dump requests from the various clients, which are checked for presence and correct synchronisation after every dump. A view of some signals captured by this IPOC-TSDS is shown in Fig. 4. The eXternal Post Operation Check (XPOC) system [7] receives the results of IPOC-TSDS analysis after every dump, and will inhibit the injection of beam in LHC in case any problem is detected after the last dump execution.

Direct Connection from the BIS to the RTS

It was noted that the LBDS is very sensitive to any unidentified failure mode of the TSDS. Despite the large redundancy within them, in case of simultaneous failure of both TSU cards, no beam DR would be executed. To reduce this sensitivity, a direct link has been established between the BIS and the RTS of the LBDS [3]. This new link consists of a new electronic board (CIBDS) that follows the same principle as the TSU: It is included in the BIS optical loops, and generates an asynchronous dump trigger when it detects an opening of the beam permit loop. This additional asynchronous trigger is delayed by 270 µs and injected in the RTL, between the MKD and MKB RTB, using a TDU of 270 µs (TDU-270), as displayed in Fig. 1.

To assess whether the addition of this new hardware has a significant negative impact on the overall rate of false beam dumps of the LBDS, a reliability study of the CIBDS card and the TDU was carried out, and revealed that the increase in failure rate is almost negligible [8].

Adaptation of the SCSS

First, the SCSS hardware and software were updated to add the functionalities for the surveillance of all the power supplies of TSDS, as explained above. In case a power supply failure is detected, a beam dump is requested before the situation further deteriorates. Secondly, the CIBDS card got a direct redundant connection to the SCSS to inhibit the CIBDS pulses on the re-trigger lines when the LBDS is in local test mode, to make sure that any activity on the BIS does not interfere with LBDS tests. A read-back of the inhibit status of the CIBDS was also implemented, and a beam dump is requested if the CIBDS is not detected as operational [3] Finally, an improvement to the SCSS was the implementation of a 'spontaneous triggering' test, to make sure that the re-trigger line is working as expected, and that all HVPG are triggered in case one of them triggers spontanously. The SCSS will send a trigger to all MKD HVPG one after the other on a dedicated line, to simulate a MKD HVPG spontaneous trigger and check that all other HVPG are triggered through the re-trigger line as expected. This system is only active when the LBDS is in local test mode.

Check of Re-Trigger Lines Continuity

The RTL is actually composed of two twisted pairs, one to transmit the re-trigger pulses and the other to allow the SCSS to check that the line is closed using a DC current. During the first tests with CIBDS, we faced an unforeseen problem: The RTL was physically closed and the DC current was normal so the SCSS did not detect any error, but the CIBDS re-trigger pulse was not visible on the IPOC-TSDS for one RTL (See Fig. 4). Further investigations showed that a pin was not connected within one of the 110 connectors included in one RTL. We then realised that the DC current loop continuity did not guarantee the continuity of the re-trigger pulse line. Such a fault could not be detected before

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

LS1, because the IPOC-TSDS picks up the signal on the RTL using a RTB placed at the output of the TDU of the TSU, at the MKD extremity of the line, as shown in Fig. 1. This non-conformity was detected thanks to the addition of the CIBDS, and the check by IPOC-TSDS of the presence of CIBDS re-trigger pulse travelling through the RTB of the 15 MKD. However, with only one IPOC-TSDS system at the MKD extremity of the re-trigger line, we cannot ensure that the re-trigger pulses reach the MKB extremity of the line, so a new IPOC system was added at the MKB extremity of the RTL, to capture and check the presence of the two re-trigger pulses from the TSU and CIBDS cards.

Problem of Re-Trigger Lines Absorption

During the re-commissioning of the RTL, we were very surprised to see that the CIBDS re-trigger pulse was not detected by the IPOC-TSDS system when pulsing at energies above 3 TeV. After further investigations, we discovered that the CIBDS pulse underwent an attenuation that increased with both the LBDS energy and the delay between the SBDT and CIBDS pulses. The CIBDS pulse is sometimes so low at the input of the IPOC-TSDS that it is not detectable by its binary digitiser (TTL). We added a new IPOC system at the MKD extremity of the RTL, in parallel with the existing IPOC-TSDS, but with high resolution digitisers, as displayed in Fig. 1.

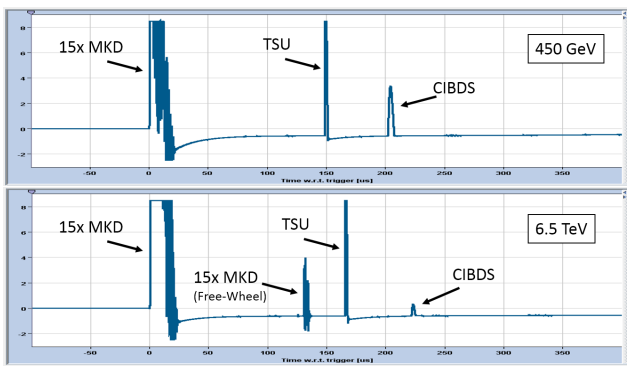


Figure 5: IPOC-RTL-MKD at 450 GeV and 6.5 TeV.

Figure 5 shows an acquisition of this new IPOC-RTL-MKD for a pulse at 450 GeV, and another at 6.5 TeV. We first see the pulse train injected by the 15 MKD HVPG when they receive the SBDT, then later the TSU re-trigger pulse, and finally, the CIBDS re-trigger pulse, which has an acceptable level at 450 GeV, but a very low level at 6.5 TeV, making its detection very difficult.

This problem of re-trigger pulse attenuation is not yet fully understood. A project was started to simulate the behaviour of the whole RTL in order to understand the phenomenon, and find a way to reduce this attenuation.

Change of TDU Delay

A pulse due to a free-wheel pickup inside the 15 MKD HVPG appears on the RTL when operating above 5 TeV, see Fig. 5. In some cases, this pulse could occlude the TSU re-trigger pulse, and thus hinder IPOC from checking its presence. This was not an issue during LHC Run I, as the LHC energy was limited to 4 TeV. To avoid this problem, we changed the TDU delay of TSU and CIBDS from respectively 200 and 250 µs to 250 and 270 µs.

CONCLUSION

Although the TSDS performed as expected during LHC Run I, some major problems were discovered, and an important number of improvements were implemented during LS1. Many diagnosis systems were added, and already proved to be useful, as they highlighted various issues with the re-trigger lines during the re-commissioning phase. A problem of re-trigger pulses attenuation is still not fully understood, and a full simulation of the re-trigger lines behaviour has been launched in order to address it. The TSDS was successfully commissioned and has been operating flawlessly since the start of LHC Run II.

REFERENCES

[1] The LHC Design Report: “The LHC Main Ring”, Vol. 1, CERN-2004-003-V-1, pp. 441-464, Geneva, CERN, June 2004.

[2] A. Antoine et al., “The LHC Beam Dumping System Trigger Synchronisation and Distribution System”, ICALEPCS’05, Geneva, October 2005. https://accelconf.web.cern.ch/accelconf/ica05/proceedings/pdf/P3_020.pdf

[3] S. Gabourin, "Implementation of a Direct Link Between the LHC Beam Interlock System and the LHC Beam Dumping System Re-Triggering Lines" IPAC’14, Dresden, Germany, June 2014. http://accelconf.web.cern.ch/AccelConf/IPAC2014/papers/thpri021.pdf

[4] E. Carlier, et. al, “Design Aspects Related to the Reliability of the Control Architecture of the LHC Beam Dump Kicker Systems”, ICALEPCS’03, Gyeongju, Korea, 2003. https://accelconf.web.cern.ch/accelconf/ica03/PAPERS/TH213.PDF

[5] A. Antoine et al, "First Operational Experience with the LHC Beam Dump Trigger Synchronisation Unit", ICALEPCS’11, Grenoble, October 2011. https://accelconf.web.cern.ch/accelconf/icalepcs2011/papers/wepmu019.pdf

[6] N. Magnin et al., "Internal Post Operation Check System for Kicker Magnet Current Waveforms Surveillance", ICALEPCS’13, San-Francisco, October 2013. http://accelconf.web.cern.ch/AccelConf/ICALEPCS2013/papers/moppc029.pdf

[7] N. Magnin et al., "External Post-Operational Checks for the LHC Beam Dumping System", ICALEPCS’11, Grenoble, October 2011. https://accelconf.web.cern.ch/accelconf/icalepcs2011/papers/wepmu023.pdf

[8] Vatansever. Vatansever, "Title Reliability Analysis of the new Link between the Beam Interlock System and the LHC Beam Dumping System : Zuverlässigkeitsanalyse der neuen Verbindung zwischen dem Beam Interlock System und dem LHC Beam Dumping System" CERN-THESIS-2014-042 https://edms.cern.ch/document/1372314/1

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ADaMS 3: AN ENHANCED ACCESS CONTROL SYSTEM FOR CERN

P. Martel, C. Delamare, G. Godineau, R. Nunes
CERN, Geneva, Switzerland

Abstract

ADaMS is CERN's Access Distribution and Management System. It evaluates access authorisations to more than 400 zones and for more than 35000 persons. Although accesses are granted based on a combination of training courses, administrative authorisations and the radio-protection situation of an individual, the policies and technicalities are constantly evolving along with the laboratory's activities; the current version of ADaMS is based on a 7 year old design, and is starting to show its limits. A version 3 of ADaMS will allow improved synchronization with CERN's scheduling and planning tools (used heavily during technical shutdowns, for instance), will allow CERN's training catalogue to change without impacting access management and will simplify and reduce the administrative workload of granting access. The new version will provide enhanced self-services to end users by focusing on access points (the physical barriers) instead of safety zones. ADaMS 3 will be able to cope better with changing and new requirements, as well as the multiplication of access points. The project requires the cooperation of a dozen services at CERN, and should take 18 months to develop.

INTRODUCTION

ADaMS is a system that centralises and standardises access policies throughout CERN; it is a central management system generating access authorisations for a multitude of different physical access systems. ADaMS has existed since 2007, and was designed for the 30 access zones existing at the time and that were exclusively related to safety, each with a relatively small area and with their access points geographically close to each other.

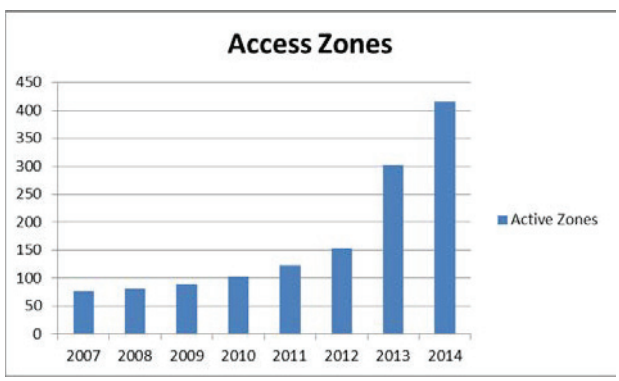


Figure 1: Evolution of the number of zones managed by ADaMS.

Today, the system is managing more than 400 zones (Fig. 1), not all related to safety, and with some of them concerning non-contiguous areas with access points very far from each other. At the same time, the business logic has changed dramatically both in quantity and quality; where originally only 5 criteria existed to grant or deny access (CERN access card, contractual situation of the person, dosimeter, followed courses and access request), new and more complex criteria have been added following new business requirements; IMPACT – a tool to coordinate the interventions on the accelerator premises during technical stops – now determines if access is granted or not, on top of the original criteria, for dozens of zones, and for around 200000 access authorizations (Fig. 2).

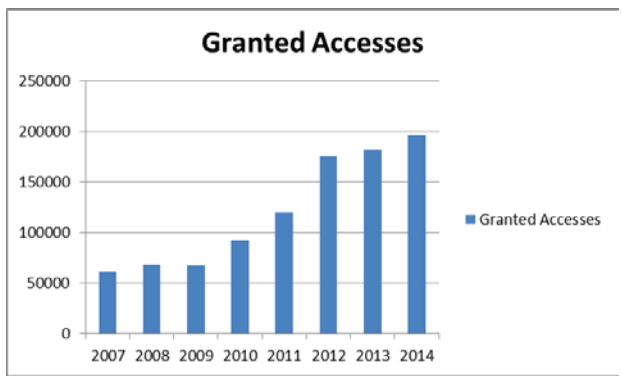


Figure 2: Evolution of the number of granted accesses by ADaMS per cycle.

With time, the changes and improvements to meet new requirements have been implemented on top of the original architecture; this means that even if for the past 8 years all new requirements were met with more or less complex adaptations of the original system, in some circumstances the system is no longer able to automatically justify the granting or revoking of an access right, as by design it does not collect and hold enough data for it. For these reasons, we have decided to build a new version from scratch.

MAJOR CHANGES FROM VERSION 2

Zones and Resources

One of the requirements that ADaMS 3 will have to fulfil is to be able to control access to entities other than "zones". Access to resources such as key cabinets, rooms

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

and cars are to be managed by ADaMS, using similar business logic to that of zones – namely criteria required to have access, notifications on access revoking, etc.

Access Points vs. Access Zones

Where in the past (with a small number of safety zones) it was easy to remember or deduce the equivalence between "safety zone" and "access point", currently (with more than 400 zones) this equivalence is difficult to remember or deduce. ADaMS 3 will focus on "access points", which is a concept closer to the end user, with a known identifier, and keep the "safety zones" concept as hidden from the end user as possible.

Access Profiles

ADaMS 3 will allow the definition of access profiles, a mechanism allowing to request a pre-defined set of access rights for a person, with a single operation. For those cases where each person arriving in a new service (e.g. CERN's Fire Brigade) is granted access to the same set of zones, the concept of an access profile will greatly simplify the acquisition of all required administrative authorisations as well as training requirements.

Identifiers Associated to Access Points

ADaMS 3 will make a clear distinction between the right to pass by an access point and the ability to do so with the needed identifier. This means that ADaMS will be able to inform the user that even if the authorisation to enter a certain area or take hold of a certain resource is granted, in practical terms, access cannot be done as the person concerned lacks the required device and/or biometric identifier.

Delegation /Simplification of Administrative Authorisations

ADaMS 3 will allow the administrative authorisation criteria for a zone or resource to be fulfilled without the circulation of an electronic document. The usage of an "egroups/roles" tool will allow to administratively allow the access to a resource without any administrative overhead. For instance, automatically authorising the shared access to a key or a room to those persons in a certain administrative unit at CERN, etc. This delegation (on the management of an egroup or role) of the administrative authorisations, even though not applicable to safety zones (where an explicit, personal signature will allow each person to gain access to it) is expected to greatly reduce the circulation of electronic documents concerning access authorisations at CERN.

Training "Ranks" (or levels)

In the beginning, ADaMS only took 3 safety courses into account; when those courses were replaced by other ones, some tweaking was done in the business logic in order to handle it (e.g. the waiving of a course A if course B had been followed, etc.). With the dozens of courses currently taken into account, and the constant changes to CERN's training catalogue, this way of working (hard-coding the business logic) can no longer be used, if an explanation on an access refusal is to be automatically deduced and a trace of changes in ranks/courses requirements is to exist. ADaMS 3 will interface with CERN's new LMS (Learning Management System), and delegate to it the management of "ranks" (or "levels") and courses. The system works with "ranks", independently of the course or certificate that grants it.

Better Anticipation of Changes

ADaMS will allow the definition of start and end dates for the criteria required (e.g. a new required rank) for a certain zone or resource, and will allow the evaluation of the impact of that change beforehand, namely in accesses granted or lost. It will also allow notification to those persons affected by the change before the change occurs.

More Systems Managed and Interfaced

ADaMS 3 will have more systems managed/interfaced and will work with more data about each of those systems; ADaMS was created to handle 1 access system for CERN's infrastructure and one accelerator access system. It was later enlarged and it now interfaces with multiple accelerator access systems (LHC/SPS/PS), 2 generic non-accelerator access systems, as well as SALTO for door locks, TRAKA for key cabinets, Altaïr for car barriers, and RFID reading systems for "kiosk IT". The interface between ADaMS and these systems is for the time being unidirectional, as all of them fetch data from ADaMS, but supply none, as ADaMS 2 does not work with any data from any physical system. ADaMS 3 will fetch data about the access systems, and will hold data about which systems are used, where, when and with which identifier. By scaling up the new architecture, it will be able to accommodate any new access control systems that may be added (e.g. Indico for room reservation and the SPS Access renovation project).

NEW INTERFACES

ADaMS 3 new graphical user interface will answer to the growing usage of the system by end users (Fig. 3); it will be leaner and more intuitive for the users, proposing more ways to find out if access to a certain location is granted or not; access checks will be proposed by access point name, location code or geographical location (in a map). It will be focused on access points, and not on access zones, and will be able to explain with a simple text, why access has been granted or revoked. It will also allow to easily identify the missing criteria in order to obtain access to a zone or resource.

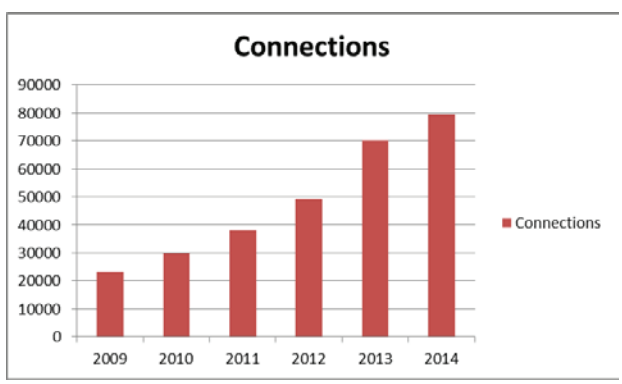


Figure 3: Evolution of the number logins to ADaMS GUI.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The new system will have functionality providing an overview of all access conditions and situations for the persons in an egroup or organisational unit; this will allow managing the accesses of a series of related persons, in an easy way.

ADaMS 3 will make data available to other systems via the conventional mechanism of database table or view; it will also expose data via RESTful Web Services.

ADaMS will propose simple, visually clear pages to be displayed at access points, with the reasons for an access refusal, at real time.

PERFORMANCE

The new system will perform a full evaluation of all accesses in less than 5 minutes (Fig. 4). This way, we can ensure a high degree of reactivity to any change in the source data used by ADaMS, This period of a refresh cycle will allow any modification to accesses to be propagated to any access point in less than 30 minutes, if desired.

ADaMS 3 will keep a full log of all changes in the source data; this new feature will allow to justify why an access has been lost or granted due to a change in the original system where data is fetched from, without the need to access that system's log mechanism (for those cases where there is one).

Unlike in the past, where for most criteria only the "date" was considered, ADaMS 3 will use the "hours and minutes" for start and end dates; this will allow to have finer access control, and better adapt to other systems such as IMPACT, that use the hours and minutes as well.

ADaMS will have a mechanism that will be able to asynchronously evaluate and make available one person's situation; this feature will be required if ADaMS is to be used as an individual protection equipment validator for access control in the future.

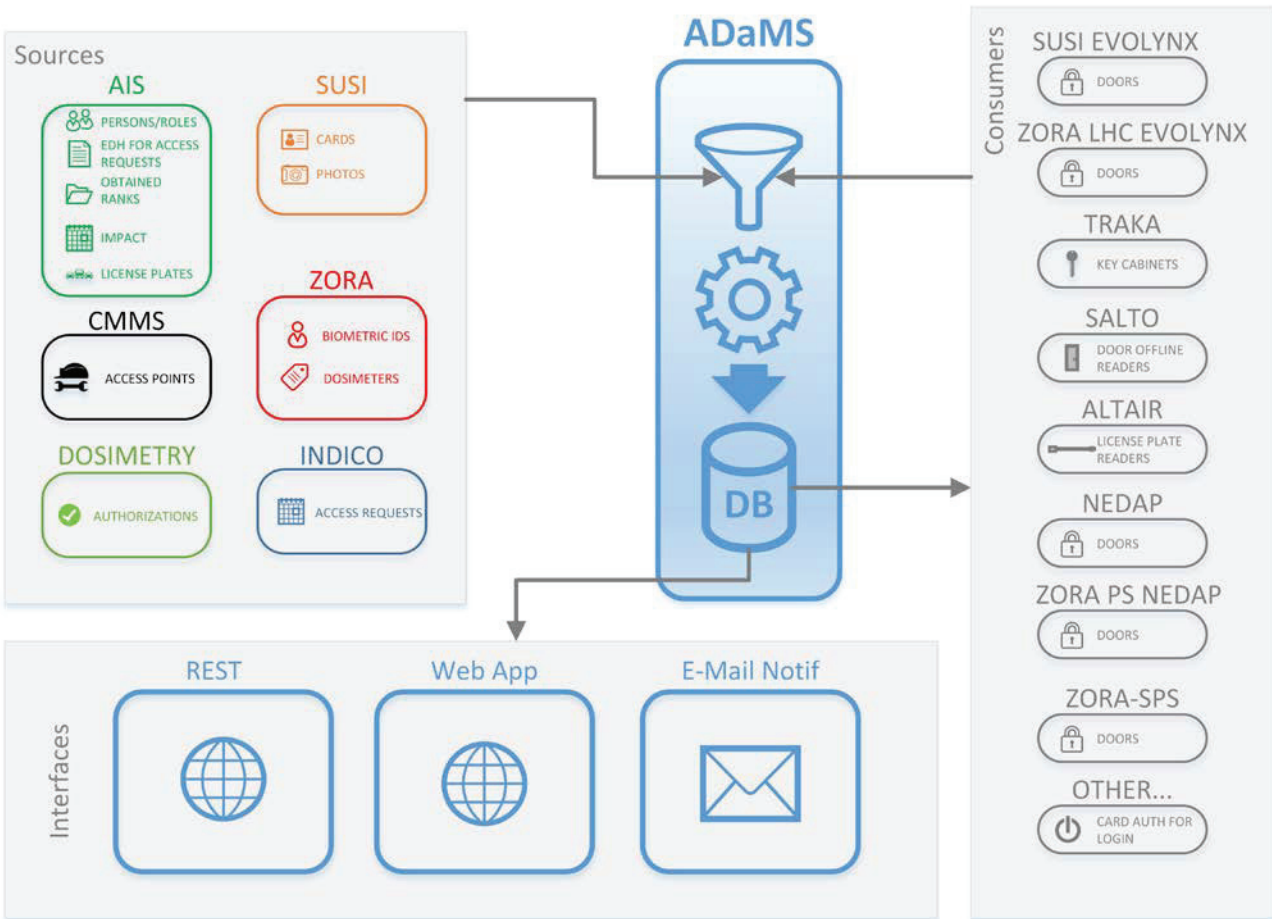


Figure 4: ADaMS 3 refresh mechanism, with its data sources and consumers.

ROLE BASED ACCESS CONTROL

ADaMS 3's interfaces (both GUI and programmable) will have a Role Based Access Control (RBAC) mechanism that will allow (or deny) a certain functionality to be applied to a particular data set; for instance, access to all access data concerning a contract's personnel will be granted to the person defined as the contract's manager.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

RBAC will be implemented using CERN's AIS (Advanced Information Services) roles/egroups application, with role types and targets; the logged user is able to perform a certain function on data belonging to a certain person, if he/she is assigned to a role type that may perform the function on a target containing that person.

CONCLUSIONS

Physical access control for an organisation such as CERN is an evolving domain, with new and more complex constraints and devices being introduced constantly. ADaMS has been delivering access data to these systems, in accordance with changing policies since 2007; its original architecture has been stretched to the limit and it is now time to refresh its architecture.

ADaMS will help analyse and allow the modification of our access policies following the observations of the Long Shutdown 1 (LS1) access patterns and problems. It will also facilitate the access process and user experience for accelerator zones, providing new functionalities and interfaces.

ADaMS is also a key element in any access renovation project, as it needs to adapt to new constraints and requirements (e.g. the SPS access renovation project).

ADaMS 3 will allow CERN to continue having a single, centralised system for access management and distribution, adaptable to new requirements and scalable to the new domains where access control is required, while being at the same time, an important link in the safety chain of CERN's accelerators.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INTERLOCK OF BEAM LOSS AT LOW-ENERGY PART OF J-PARC LINAC

A. Miura#, Y. Kawane, N. Kikuzawa,
J-PARC Center, Japan Atomic Energy Agency, Tokai, Ibaraki, 319-1195, JAPAN
T. Maruta, T. Miyao
J-PARC Center, High Energy Accelerator Research Organization, Oho, Tsukuba, 305-0801, JAPAN

Abstract

The output beam power of the J-PARC linac has been improved by increasing the acceleration energy and peak beam current. The beam loss is getting serious along with increasing output beam power; however, the beam loss in the front-end region is difficult to detect because of the low energy of the radioactive emission. An interlock system using the beam current monitors has been developed to prevent significant material activation. In this system, an electrical circuit measures the beam transmission between the two beam current monitors. This study describes the design and performance of this electrical circuit and introduces the system configuration.

INTRODUCTION

The Japan Proton Accelerator Research Complex (J-PARC) linac can provide high intensity beams of peak current 50 mA, beam energy 400 MeV, pulse width 0.5 ms, and repetition rate 25 Hz using a radio frequency quadrupole linac (RFQ), three drift tube linac (DTL) cavities, 16 separation-type DTL (SDTL) cavities, 21 modules of annular coupled structure (ACS) cavities, and two beam transports, which have two ACS-type buncher cavities and debuncher cavities as shown in Fig. 1 [1, 2].

The J-PARC linac has a matching section between the RFQ and the DTLs, where 3-MeV beams are transported and measurements and beam profile matching are performed. An ion chamber-type beam loss detector has been employed for beam loss detection. The beam loss detector has a fast signal response; therefore, we have used this to develop an interlock system to stop beam operation when abnormal beam shots are detected. However, there is no beam loss detector in this section because the energy of radiation generated by 3-MeV beam is not sufficiently high to be detected.

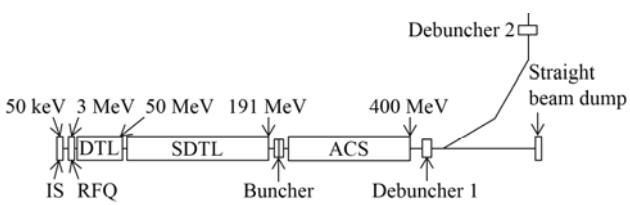


Figure 1: Schematic Layout of Present J-PARC Linac.

An interlock system that uses a new electrical circuit to measure the beam transmission between the two beam current monitors is proposed because of the large number of beam current monitors in the beam line.

In this study, a procedure for signal processing in the system and for processing waveforms in the processing unit is introduced. The waveforms obtained are also analyzed.

CIRCUIT DESIGN OF BEAM TRANSMISSION MEASUREMENT

The new interlock system, which uses beam current monitors, is named the beam transmission monitor (BTM) because the system compares beam currents detected by two independent beam current monitors. The signal design process is described below.

System Configuration

The schematic configuration of the system that uses the BTM is shown in Fig. 2. The system comprises two beam current monitors, and the signal processing circuit (BTM) shown in gray. Pre-amplifiers are usually used for continual beam operation and they are annually tuned to 0.10 V/mA of the beam current. The buffer amplifier has four channels per module, with two channels connected to the digitizer (one for both waveform outputs, 1 and 2) during operation, and the other two channels used by the BTM. The BTM circuit has six output terminals, as shown in Fig. 2.

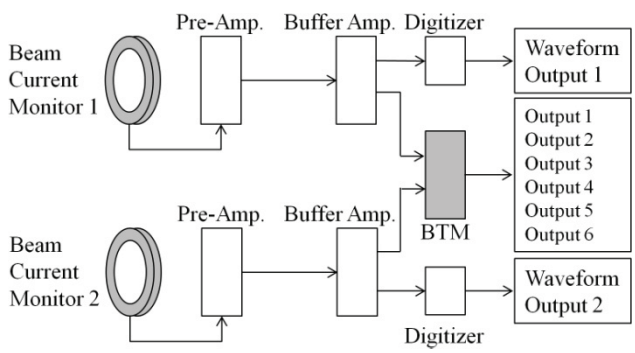


Figure 2: System Configuration of the Beam Transmission Measurement.

Process Flow

The signal processing flow in the BTM is shown in Fig. 3. It has two input terminals for the two signals coming from the two beam current monitors. Just after the signal input terminals, gain adjustment knobs amplify the input

#akihiko.miura@j-parc.jp

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

signals independently. These can be tuned by observing the low beam current signals (outputs 1 and 2). To detect small beam losses, the beam current signals are integrated with a gate signal that covers an entire beam pulse. The rise of the gate signal pulse plays a significant role in the trigger of the integration, whereas the flank plays a significant role in the reset of the integration. After integration, which can be observed in outputs 3 and 4, the difference, which should be small, is usually taken by the two integrated inputs. The difference can be observed at the output 5 terminal and it is integrated again into the circuit. The final integrated differential signal can be observed in output 6.

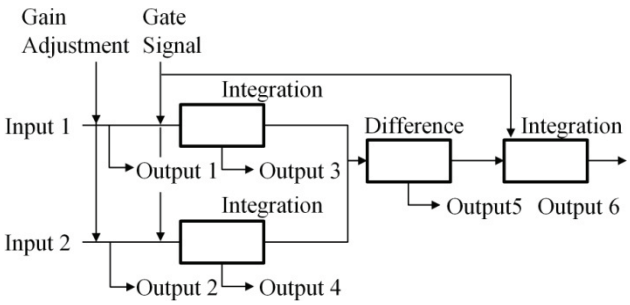


Figure 3: Block Diagram of the signal processing flow in the beam transmission monitor circuit.

Estimated Waveform

Figure 4 shows the estimated waveform at each output terminal. Two rectangular simulated beam input pulses are assumed with different wave heights at inputs 1 and 2. When input 1 is upstream from input 2 and the input gain is adjusted, the wave height of input 2 is smaller than that of input 1. After inputs 1 and 2 are integrated during the gate period, outputs 3 and 4 are obtained. The waveforms of outputs 3 and 4 are proportionally increased between beam-on and beam-off, and the highest peak is held ideally until the reset signal. The integrated signals are subtracted and the result is output 5. Furthermore, this signal proportionally increases in a beam period. The proportional difference is added over time, with the integrated difference being quadratically increased. After the beam has passed, the difference is proportionally increased until the reset signal.

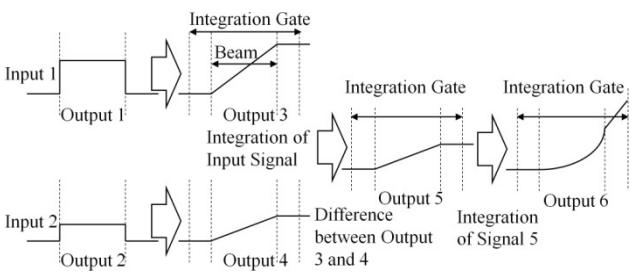


Figure 4: Waveforms during the signal processing.

SYSTEM TEST

Installation

There is a beam transport line between the RFQ and DTL, where the beam energy is 3 MeV, and there are five beam current monitors in this section. We assume that all the beams are lost in the DTL cavity, and we focus on the beam current monitors upstream and downstream of the DTL. Therefore, input 1 is occupied by the beam current signal from the last MEBT1 and input 2 is for the last DTL. In the test operation, we use a 30-mA peak beam current and a 100-μs pulse length. The integration gate covers the beam pulse preceding 50 μs from the beam pulse and having a width of 2 ms.

Test Results–Normal Operation

When the beam is operated with tuned beam lines, the beam loss in the DTL is assumed to be small. When the input currents are the same, the trends of inputs 1 and 2 as well as those of outputs 3 and 4are the same. Therefore, the subtraction of outputs 3 and 4 gives a value of zero for output 5, which will result in no signal in output 6.

For the beam test to use the ideal tuned beam line, inputs 1 and 2 were maintained with the same pulse height by using the gain adjustment knobs for the input signals. The waveforms of output 1 (raw output of input 1), 2 (raw output of input 2), 5 (difference of integration between 3 and 4), and 6 (final integration of the difference 5) are shown in Fig. 5. We used a pulse of 100 μs with a horizontal axis scale of 20 μs/div.

Figure 6 shows the waveforms of outputs 3 (integration of input 1), 4 (integration of input 2), 5, and 6 at time ranges of 40 μs/div along the horizontal axis. Integration of the input pulses, which started from the gate state, shows the same trends for outputs 3 and 4, with the difference between (output 5) of outputs 3 and 4 then being close to zero. Finally, the integration (output 6) of output 5 was shown to be almost zero.

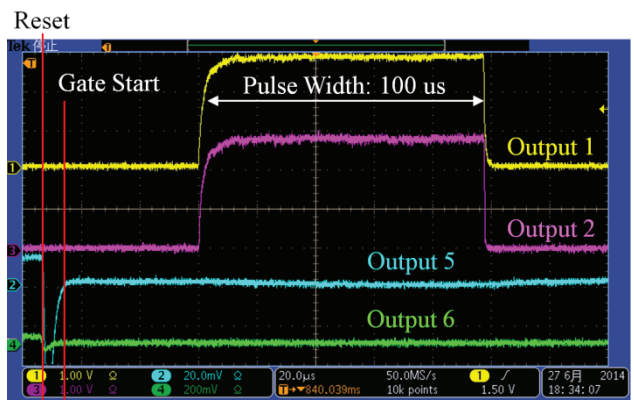


Figure 5: Waveforms of outputs 1, 2, 5, and 6 when pulses with same heights are input. The horizontal axis scale is 20 μs/division.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

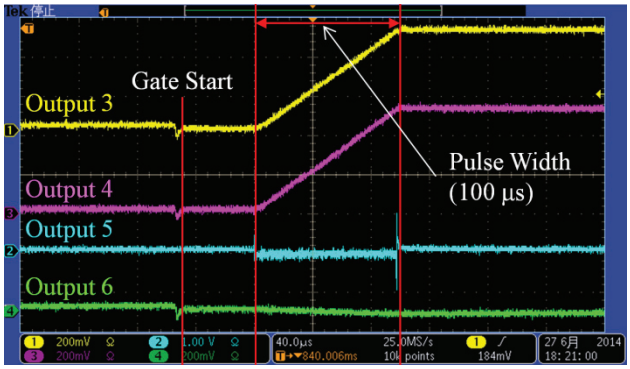


Figure 6: Waveforms of outputs 3, 4, 5, and 6. The horizontal axis scale is 40 μs/div.

Test Results–Beam Loss in the DTL Section

We assumed the entire beam to be lost in the DTL section. Input 1 remained; however, input 2 was suspended from producing any signals. Figure 7 show the waveforms of the outputs 1, 3, 5, and 6 at 400 μs/div along the horizontal axis. Output 3 is proportionally integrated and the difference of output 3 and input 2 (actually no signal input) was also proportionally integrated and the peak value was maintained until the reset signal. While the beam was passing, the proportional difference was integrated quadratically.

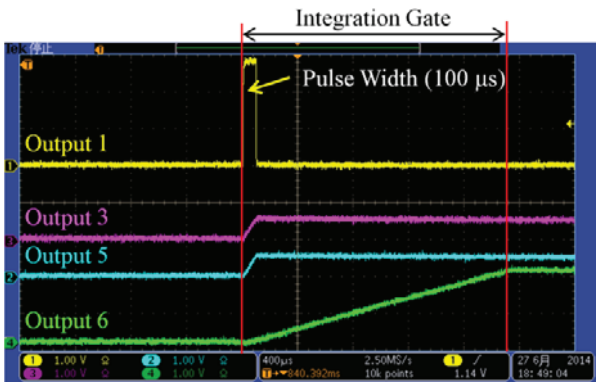


Figure 7: Waveforms of outputs 1, 3, 5, and 6 for beam loss in the DTL section. The horizontal axis scale is 400 μs/div.

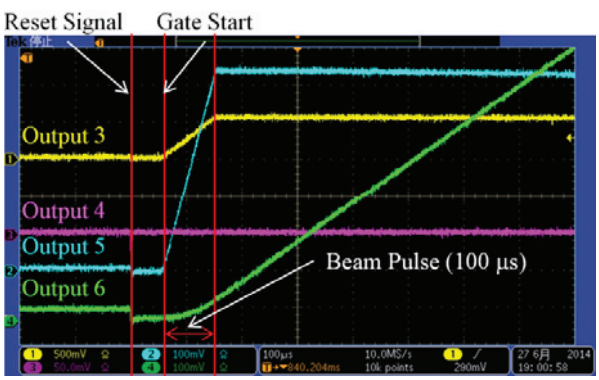


Figure 8: Waveforms of outputs 3, 4, 5, and 6 with beam loss in the DTL section. The horizontal axis scale is 100 μs/div.

After the beam had passed, the integration of the differential signal was proportionally increased due to the remainder of the peak value. Figure 8 shows the extensions of the waveforms of outputs 3, 4, 5, and 6 around the pulse duration. A quadratic increase of output 6 was observed.

We tested the beam loss detection system in the case of normal operation with beam loss occurring in the front-end section (DTL section). We confirmed that the signal outputs of all the terminals worked as estimated.

Thresholds for the Interlock

We confirmed the performance of the BTM by obtaining the maximum output of the integrated difference. In the interlock level, the worst case is that provides the maximum output from output 6. The reason for using output 6 is that it is sensitive to the small difference between outputs 1 and 2. We decided the threshold value as that which suspends the beam operation alarm. We are now using half of the maximum value, which implies that the alarm is triggered later than halfway through the pulse. The signal rise of output 6 is slow because it increases quadratically in the case of ideal beam loss. We can obtain a quicker response from output 5 than from output 6. Nevertheless, we plan to use both outputs 5 and 6.

This system can also be used for comparing two different signals. In the front-end of the beam line, an RF chopper system is used. As long as the RF chopping error is not sufficiently high to shift the beam away to the beam scraper, the beam remains in a beam line. Because this beam should not be injected into the downstream synchrotron, it is necessary to stop the beam after the chopper error occurs. We have attempted to use the system for detecting chopping errors using a beam current monitor and a beam scraper where the beam was irradiated by the chopper [3].

Operational Use

We continue to use the system for its usual operations. We experienced a triggering of alarm during beam operation when all the magnet setting parameters were lost due to a human error. The system was confirmed to be successful when used in actual beam operations.

SUMMARY

The J-PARC linac has a matching section between the RFQ and DTL, where 3-MeV beams are transported. We usually employ an ion chamber-type beam loss detector for beam loss detection in the linac beam line; however, there is no beam loss detector in the front-end section of the linac because the energy of radiation generated by the 3-MeV beam is not sufficiently high to be detected.

A new interlock system using beam current monitors is proposed. We developed an electrical circuit to measure

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

the beam transmission using two beam current monitors for the system. We established the new system and tested it using an actual beam in two typical cases. Based on the results obtained in all the cases of beam loss, we obtained the worst signal response, which will be used as a threshold for the interlock. We confirmed the performance of the BTM and its system as designed and decided on the operational threshold. We have been using this system continuously; however, the strategy of the threshold should be more serious if used for alarm timing.

REFERENCES

[1] Y. Yamazaki, eds. *Accelerator Technical Design Report for J-PARC*, KEK Report 2002-13.

[2] H. Oguri, Proceedings of IPAC2013, Shanghai, China, WEYB101, 2013.

[3] Y. Kawane, et. al., Proceedings of the 11th Annual Meeting of Particle Accelerator Society of Japan, Aomori, SUP099, 2014 (in Japanese).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

OVERVIEW AND DESIGN STATUS OF THE FAST BEAM INTERLOCK SYSTEM AT ESS

A. Monera Martinez, R. Andersson, A. Nordt, M. Zaera-Sanz, ESS, Lund, Sweden
C Hilbes, ZHAW, Winterthur, Switzerland.

Abstract

The ESS, consisting of a pulsed proton linear accelerator, a rotating spallation target designed for an average beam power of up to 5 MW, and a suite of neutron instruments, requires a large variety of instrumentation, both for controlling as well as protecting the different hardware systems and the beam. The ESS beam power is unprecedented and an uncontrolled release could lead to serious damage of equipment installed along the tunnel and target station within only a few microseconds. Major failures of certain equipment will result in long repair times, because it is delicate and difficult to access and sometimes located in high radiation areas. To optimize the operational efficiency of the facility, accidents should be avoided and interruptions should be rare and limited to a short time. Hence, a sophisticated machine protection system is required. In order to stop efficiently the proton beam production in case of failures, a Fast Beam Interlock (FBI) system with a targeted reaction time of less than 5 microseconds and very high dependability is being designed. The design approach for this FPGA-based interlock system will be presented as well as the status on prototyping.

INTRODUCTION

The team in charge of ESS Machine Protection is facing a great challenge: design, deploy and operate a Beam Interlock System (BIS) needed to protect the machine from beam-induced damage. The unprecedented beam power of 5MW at ESS can lead to severe damage within 10-20 microseconds only and thus beam operation has to be stopped reliably in an even shorter time upon the detection of non-nominal beam conditions. Such non-nominal beam conditions will be detected by several beam instrumentation systems such as the Beam Current Monitoring System, the Beam Loss Monitoring System and the Beam Position Monitoring System. The BIS shall fulfil demanding requirements regarding the hardware failure rate and beam availability as well as the requirements for the very short response time of a few microseconds. There are only four years to develop, test, produce and install the whole system. First commissioning of the ESS linac is planned for the beginning of 2018, and a first version of the BIS has to be ready by then. First protons on target are expected in mid 2019.

In a first step, the possible failure modes of the accelerator systems have been analysed. This analysis has been used to define around 170 different protection functions needed to protect the linac from beam induced damage, directly or indirectly. For each of these protection functions, an execution time has been defined as well as the required rate of dangerous failures. The highest rate for a function found is of 10^{-6} to 10^{-7} failures per hour with a response time of 5 microseconds (including the detection, processing and execution time) [1].

SYSTEM SPECIFICATIONS

In order to design a Beam Interlock System it is important to understand the system requirements and specifications. The design of the BIS is following the IEC61508 [2] and the IEC61511 [3] standards where applicable. However, since the BIS is a mission critical system and not a safety critical system, it is not needed to be compliant with those standards. Still, the team felt, that it is good to follow the guidelines provided by these standards in order to achieve a good level of protection thanks to the solid framework that the standards offer.

The following selected specifications allow to have a clear picture of what is expected:

Remote Monitoring: The BIS should be able to inform about its status and in case problems occur, operators shall be notified or an alarm shall be raised. For that it is required that the BIS will be connected to the main control system (EPICS) for monitoring and parameterization purposes.

Fast response time: the fast reaction time of the BIS is resulting from a time allocation which is done for a full protection function. A protection function includes time to detect a non-nominal condition of the machine, to process this information, to propagate and inform the BIS, process the information on BIS level and initiate a stop of beam operation. The fastest reaction time found for a protection function is 4-5 microseconds implying a reaction time of the BIS below 2-3 microseconds. This time differs for different beam energies along the linac and is only valid for the low energy part of the linac and can be relaxed for higher beam energies. Hence the Beam Interlock System will be called the Fast Beam Interlock (FBI) System. In order to achieve this reaction time the development of an FPGA-based system is required.

High availability: The FBI System should not interrupt the beam operation unnecessarily. One of the main requirements for the ESS facility is to deliver beam with very high availability. Thus, it is needed to keep downtime and false beam stops due to failures in the FBI System at a minimum. Hence the FBI System should be highly reliable as well as highly available.

Protection: The FBI System shall fulfil the requirement of a dangerous failure rate of 1.5 x 10^{-7} to 1.5

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

x 10^{-8} per hour. This requirement is based on a first risk analysis [1].

Number of Inputs: The number of inputs towards the FBI System will be in the order of a few hundred signals (~300), randomly distributed in a 600m long gallery (where the electronics is located). The final number of inputs is currently under investigation. However, it is not expected that this number will significantly increase.

The Environment

Initially underestimated by many, the environment in which the system is located plays an important role when defining the system's architecture and its distribution. All the electronics needed to operate the accelerator will be installed in the so called Klystron Gallery at ESS. This gallery is very close to the accelerator to keep the cable length at a minimum. Racks will grouped in two rows of 18 racks each and a group of 36 racks will be housed in an enclosure which is cooled, fulfilling the high temperature stability requirements for the RF system. There will be approximately 24 enclosures for ~800 racks in total. Due to the confinement of these enclosures and requirements for each of them, it could be difficult to distribute cables in and out. It is important to have this factor in mind when defining the location for the different electronics.

Additional non-physical factors like electromagnetic noise impact on the performance level of a system. The FBI System should be designed in a way that this noise doesn't damage the electronics and doesn't impact on the beam availability of the accelerator. Special attention will be given to the selection of cables and connectors, focusing on shielding and good noise immunity but not compromising price and installation cost significantly.

TRADITION VS TECHNOLOGY

Collaboration with many institutes has allowed us to get hold on already developed interlock systems, designed by experienced teams during many years and with proven functionality. The idea of developing a BIS in a short time reinforces the strength of such collaboration and offers the possibility of doing a fast development adapted to ESS needs. On the other hand, new technologies will allow tailor-made solutions for ESS at the cost of lack of reputation and longer time to develop and test.

PROTOTYPES UNDER DEVELOPMENT

Two main ideas are currently being developed, both based on the same principle: Interface modules will allow a wide variety of electronic devices to connect to the FBI System, concentrators will aggregate the signals to simplify and minimize the number of traces and a main master module that will decide if the beam operation has to be interrupted or not.

The FBI System will receive a boolean signal from the Local Protection Systems (LPS) and from Beam Instrumentation (BI) systems indicating if beam operation should be permitted or not. This boolean signal will be the "Beam Permit" signal. Upon removing this signal, beam operation will be stopped immediately by the FBI System.

Copper-based System

Following the approach that CERN took for their Beam Interlock Systems deployed at LHC [4], LINAC4 and other machines at CERN, ESS will build a similar system following the main design concepts of the CERN Beam Interlock System. This system is based on DC signals, a modular design and very reliable electronics.

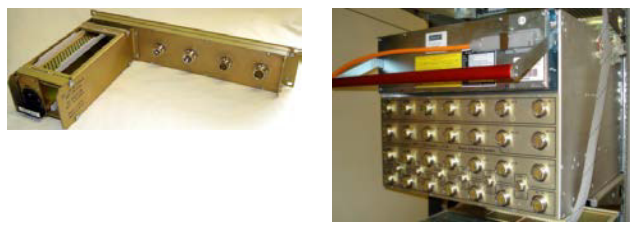


Figure 1: CERN Interface Module (CIBU) (left), Beam Interlock Controller (right).

Designed for VME form factor, this system was designed not only to be very robust mechanically and electronically but also to be operated safely in radiation areas [5].

The ESS version of this system will have slight changes in the architecture in order to be accommodated to the ESS Klystron Gallery distribution and the size has been reduced to fit the restricted dimensions without compromising the mechanical strength.

The FBI System will be composed of the following:

The FBI Device Interface (FBI_DIF) will receive the first size reduction and higher integration count, aggregating into a 1U unit up to 8 interfaces without losing availability or protection integrity level. The electronic used will be very similar to the original CERN electronics keeping a wide input range (from 3.3V to 36V) allowing an easy connection, for PLC and FPGA based systems.

The FBI Master (FIB_M) and FBI Master of Masters (FBI_MoM) will undergo major re-design. The form factor will be changed from VME to MicroTCA (or Pizza box) and aggregation from 16 to 32 inputs into a single crate depending of the master location, is foreseen.

The FBI Actuator module (FBI_A) has been designed following ESS requirements. A redundant input and redundant signal path within the actuator are used to avoid dangerous failures. In addition, diagnostics have been implemented to supervise the behaviour together with redundant power supplies and a discrete critical path (no programmable devices on the permit path) ensuring high availability.

An example of the connector chosen and the mechanical redesign is shown in Fig. 2.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

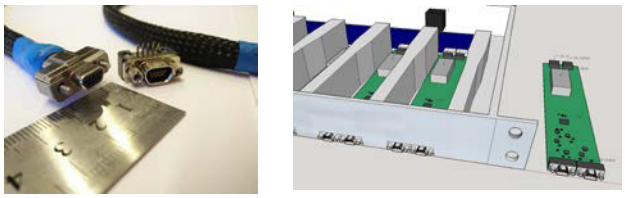


Figure 2: Micro D connector cable for the FBI System (left), 3D model for the FBI Interface module (right).

In order to fulfil the required dangerous failure rate, the FBI System will be installed in a redundant configuration. However, due to restrictions on budget and space limitation in the accelerator, the Input devices will not be redundant, only the FBI System will be, offering redundancy from the FBI_DIF up to the Master of Masters. The FBI_A will act as a final, simple but very robust concentrator that will use both permits to turn the actuators on and off when needed.

Furthermore, to avoid single points of failures at the inputs of the FBI System, the Local Protection Systems and Beam Instrumentation systems shall offer a redundant output independently connected from the CPU or FPGA that processes the main protection function.

A small diagram of the system's architecture can be seen in Fig. 3.

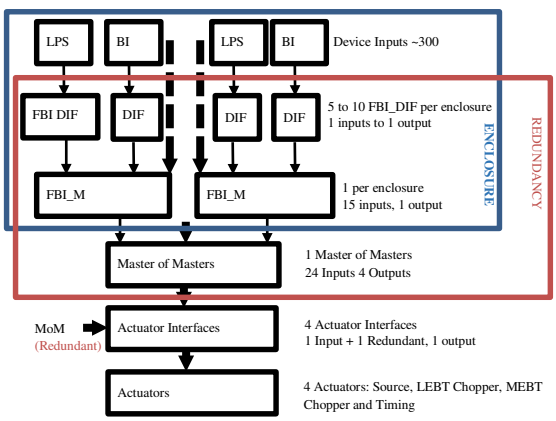


Figure 3: Schematic of the Fast Beam Interlock System based on copper connections.

A recent Failure Mode, Event and Diagnostics Analysis (FMEDA) [6] has been performed for the first engineering model of the FBI System. The results of this analysis show that the system with minimum firmware and redundant channels would offer 1 dangerous failure within 450 years while keeping the false beam trips to less than 3 per month and a reaction time of less than 2 microseconds excluding delays induced by cables.

In addition to the hardware and mechanical modifications made to the CERN System, an extra level of protection has been added to ensure full system functionality and reduce likelihood of blind failures even more.

Self-Testing Functionality

In order to ensure that the FBI System is ready to allow for beam operation, a self-test will be implemented and executed before each beam injection. This test will force the FBI System and input devices to prove their ability to transmit a non-permit command from the protection function to the FBI Master of Masters. Using this self-testing sequence, it would be possible to find dangerous failures on the FBI System itself but also at the input devices.

Due to time restrictions and to minimize the system blindness, this test should be only applied to the FBI System itself and to FPGA-based systems but not to the PLC based systems, which have a slower reaction time.

In addition, this test sequence will be realized on different time scales for each channel. That means, that at all times, only one out of two channels is in test mode.

Additional System Tests

Two additional tests will be used in order to ensure the full functionality of the whole FBI System.

Slow Test: This test will be initiated by the timing system when there is no beam in the machine. The test will virtually generate all the events that can trigger a protection function ensuring the correct behaviour of all elements that form part of every protection function. This test would be scheduled after each technical stop to warranty the protection integrity.

Actuator Test: This test will be executed after the Slow Test but with minimal beam power. During this test the Master of Masters will generate independent triggers that will test all the beam stopping mechanisms (one at the time).

Adding testing and diagnostics functions reduce the dangerous failure rate to below one failure in 1000 years [6].

Optical Fibre Gigabit System

During the analysis of the copper based FBI System it has been found that DC data paths can be easily corrupted due to potential electromagnetic interferences. Using DC signals for transmitting critical information may hide dangerous system failures. An alternative system design that will use gigabit optical links transmitting data packets instead of RS485 links with DC signals, is being evaluated currently.

The FBI System based on optical fibres will use high speed optical links between FPGAs through SFP optical transceivers, allowing for a noise free communication at very high speed. If compared to the copper based system, the gigabit links will always fail into a safe state due to the interruption of the data packs. 10 km transceivers will be chosen for the 600m long Klystron Gallery, allowing a long lifespan of the transceivers and withstand some optical fibre degradation before start failing. It is expected that thanks to the high speed devices, the system will be have a much shorter reaction time.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Furthermore FPGA internal serializers with speeds of up to 6gps, will reduce the number of components and the size of the different boards.

The Beam Permit signals will be transmitted together with the device serial number, supply status and other diagnostics features in data packets allowing not only to know if the beam permit is on or off but also knowing about the health of the interface connected.

While trying to analyse the system, it is clear that the FPGA will become the most critical element in this scheme and thus potentially a weak point due to the higher complexity of the device. At the same time the short lifespan of the fibres may increase the maintenance cost of the system. However, if it is proven that the dangerous failure rate and reliability improves, this increase in maintenance is affordable.

The conceptual idea has been drafted and the design of the first engineering model will start soon.

Optical Fibre System Layout

When designing a system, it is important to consider the integration of the system as well as the installation. In case of ESS, the rack enclosures will heavily impact on the maximum number of cables that can go in and out of each enclosure together with the time required for cable installation.

In order to allow for an easy integration, each enclosure will be equipped with an FBI_Master module combining all the signals from devices located within each rack enclosure into one single permit line. This line will be then connected to the main master or Master of Masters (MoM) located close to the actuators.

Following this strategy allows for the reservation of space for cables and racks within each enclosure ensuring the presence of an FBI_Master module in each enclosure and maintaining low cable installation cost.

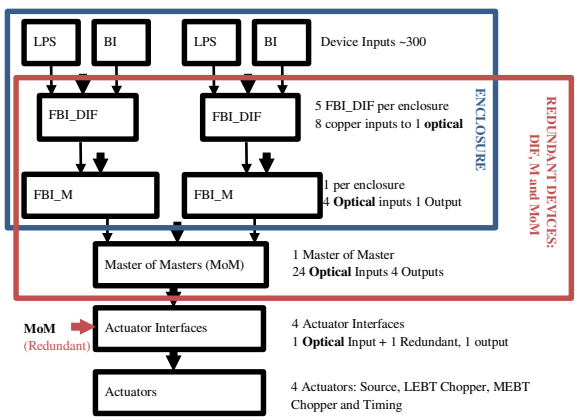


Figure 4: FBI System Optical Layout.

CONCLUSIONS

A first prototype based on the copper connections has been developed already but other design concepts are under investigation. Future iterations will take the ESS architecture more into account, trying to minimize the total installation cost.

An FMEDA analysis can improve the design significantly, such that weak points in the design can be easily detected. Further improvements will be done for the copper based FBI System, reducing the dangerous failure rate and improving the systems reliability.

REFERENCES

[1] A. Nordt et al., "Preliminary Risk and Hazard Analysis of the ESS Machine Protection System with Emphasis on Production and Property Losses in the ESS-LINAC", ESS Report No 0003796 (2013).
[2] IEC61508 Edition 2.0
[3] IEC61511 Edition 1.0
[4] B. Todd, A Beam Interlock System for CERN High Energy Accelerators, Brunel University, London, United Kingdom (2006).
[5] B. Puccio et al, "Machine Interlock System in LHC Point 7", CERN, Geneva, Switzerland (2008).
[6] R. Andersson et al, "A Modified Functional Safety Method for Predicting False Beam Trips and Blind Failures in the Design of an Accelerator Beam Interlock System", MOPGF126, these proceeding, ICALEPCS'15, Melbourne, Australia., (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INTEGRATION OF PLC'S IN TANGO CONTROL SYSTEMS USING PyPLC

S.Rubio-Manrique, M.Broseta, G.Cuní, D.Fernández-Carreiras, A.Rubio,
J.Villanueva, ALBA-CELLS, Cerdanyola del Vallés, Barcelona, Spain

Abstract

The Equipment Protection Systems and Personnel Safety Systems of the ALBA Synchrotron are complex and highly distributed control systems based on PLC's from different vendors. EPS and PSS not only regulate the interlocks of the whole ALBA facility but provide a large network of analog and digital sensors that collect information from all subsystems; as well as its logical states. TANGO is the Control System framework used at ALBA, providing several tools and services (GUI's, Archiving, Alarms) in which EPS and PSS systems must be integrated. PyPLC, a dynamic Tango device, have been developed in python to provide a flexible interface and enable PLC developers to automatically update it. This paper describes how protection systems and the PLC code generation cycle have been fully integrated within TANGO Control System at ALBA.

INTRODUCTION

ALBA[1], member of the Tango Collaboration[2][3], is a third generation Synchrotron in Barcelona, Spain. It provides light since 2012 to users through its 7 beamlines,with 2 more under construction. The ALBA Control Section (ACS) is currently formed by 16 engineers devoted to the development of our Tango-based SCADA frameworks (Taurus[4], Sardana[5][6] and PANIC[7]) and PLC systems.

The ALBA Equipment and Personnel Protection Systems[8][9] (EPS and PSS) are distributed PLC-based systems autonomous from the Tango Control System. Both EPS and PSS are homogeneous systems based on single vendors (B&R and Pilz respectively). While the tasks to be done by the PSS are clearly specified and delimited by the ALBA Safety Group, the EPS PLC's became instead a versatile system that has been adapted for interlock, acquisition and motion control in both accelerators and beamlines.

Although other PLC based systems are used in ALBA to control the RF circulators, bakeout controllers and water or air cooling systems; the EPS is the most complex system managed by PLC's, using 58 B&R CPU's and 110 periphery cabinets to collect more than 7000 signals. In addition to the main purpose of protection, several hundreds of signals distributed across the whole system are acquired for diagnostics and control of pressures, temperatures and movable elements

The integration of the management of an independent system like EPS in the Tango Control System required of several phases, starting from the collection of cables from the Cabling Database to the final auto-generation of GUI's for both EPS Expert GUI and operator users (Taurus) .

GENERATION OF CODE VARIABLES

The Cabling and Controls Database

Every cable and equipment installed in the ALBA Synchrotron is registered in our Cabling and Controls Database (CCDB). Developed in 2007[10] by our Management and Information Software section (MIS) using MySQL and web technologies, it was the main support tool for the design and construction phase and now it is still kept updated as the main repository of equipments and configurations in our Accelerators and Beamlines. As of 2015 it lists 385 racks with 7131 equipments of different 874 equipment types. These equipments are connected using 20053 cables of 487 different cable types with a total length of 172.26 Km.

The CCDB provides, for each of our PLC CPU's or remote peripherals, the full list of connected devices, its equipment types, cable configurations, terminal used and the distribution of hardware in racks, including the routing of cables between hardware and control devices (Fig. 1). It also provides fast access to all available documentation for each type of equipment.

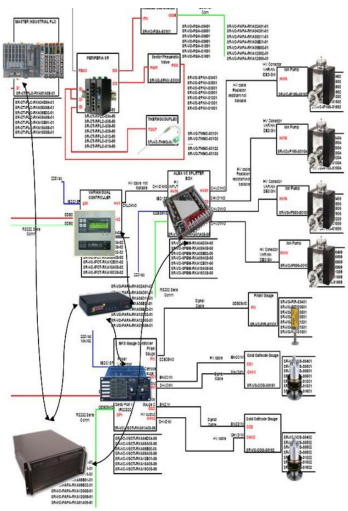


Figure 1: Diagrams of every subsystem have been produced prior to its introduction in our cabling database.

The PLC Auto-Generation Tool

The CCDB python API[11] provides full access to the Cabling Database from our control system tools. The API methods allow to search for equipments and get lists of connections, names and network information. These links are used to enable our Auto-Generation coding tool correlating the information of the equipments from the CCDB with the logics defined for them in the EPS.

To do so, all the common elements of our different Equipment Protection Systems have been standardized in

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

our EPS_LIBRARY project. This library, later exported to our different CPU sub-projects, provides the standard logic behavior for managing each equipment type connected to the PLC and its remotes. These standard procedures include managing in/out digital signals for valves, alarm/warning ranges for analog values, value conversions for thermocouples, linear correlation for 4-20 mA transducers and standard conversion scales for flowmeters amongst others.

For each CPU of the EPS, the PLC Auto-Generation tool (Fig. 2) extracts exhaustive reports of all equipments connected to the CPU and its remotes, parses the naming of these equipments and its logics depending on the cable and terminal connectors used in both sides (along with notations introduced by the Controls or Electronics engineers). This full cable report is later translated to a full list of the PLC variables with its matching PLC code structures.

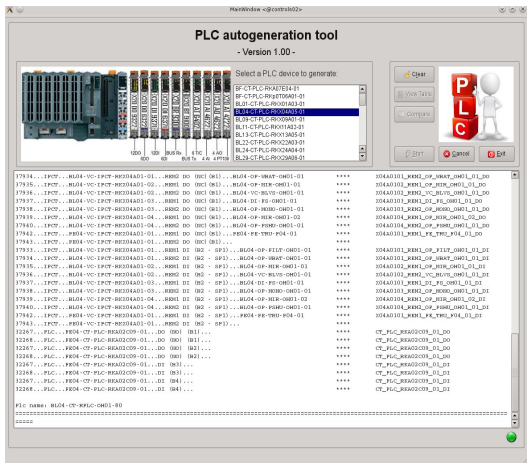


Figure 2: PLC Auto-Generation tool.

These steps does not produce the final code of the PLC, but a detailed guide of the structures to be generated. The control engineer then must coordinate logics of individual variables and program the most specific logics of every beamline or accelerators section, being capable of focusing in the most critical parts instead of the tedious and iterative variable naming.

The processes for auto-generation of code variables were originally developed using visual basic, which required manual extraction from database and generated static reports. Migrating these procedures to python allowed to develop tools that can be executed automatically, enabling the automatization of regular code review and error checking.

The final output of the auto-generation process is the variable modbus mapping spreadsheet, commonly known as the EPS CSV. This file will be used later to generate the EPS Expert GUI and the Tango Attributes used by User-Level GUI's.

PLC TANGO DEVICE SERVERS

PLC's at Alba are accessed using two protocols: Fetch & Write for Siemens PLC's (cooling systems) and Modbus (over TCP or RS485) for the rest of systems, including EPS.

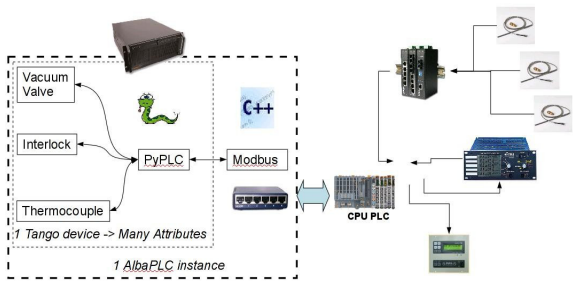


Figure 3: Structure of the PyPLC device server.

All Modbus-based PLC's are accessed using the PyPLC[12] device server. This python device server[13] (in use since 2008) have been developed over the Modbus C++ (Fig.3) device class to allow four different types of access:

- Directly executing Modbus commands in the PyPLC device server, direct actions mapped to specific bits (e.g., open Front-End).
- Reading the whole Modbus address space into array attributes, method optimized to get maximum update frequency, used by the Expert GUI.
- Exporting EPS variables as individual Tango attributes, thus enabling Tango features like ranges, alarms, archiving, labels, etc.
- Exporting EPS variables as individual Tango devices, typically valves with its own Open/Close commands.

The PyPLC exports the most standard Modbus commands (Read/Write Input/Holding Registers) and PLC variable types (Coil / Flag / Int / Long / Float / Ieee Float). This PyTango device server uses the Fandango Dynamic Attributes [14] template to generate new attributes at runtime based on user-defined formulas (Table 1). These dynamic generation of attributes boosted the initial prototyping of the devices, providing enough functionality for the most elementary systems (temperature controllers, stand-alone PLC's) and allowing to customize the attribute generation done by the EPS Auto-generation tools.

Table 1: PyPLC Attribute Formulas, an Array, a Writable Boolean Flag and Two Commands

`TEMPERATURES= DevVarLongArray(Regs(7800,100))`
`DIO_01= bool( READ and Flag(80,7) or WRITE and WriteFlag(81,7,VALUE))`
`Open_PNV01=(WriteBit(193,2,1),1)[-1]`
`Close_PNV01=(WriteBit(193,1,1),0)[-1]`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

User-defined formulas allowed to create meta variables (e.g. BL_READY) or special commands to be used in PANIC Alarms [15] or our Sardana Experimental Framework to control pneumatic devices in the Beamline (Table 1).

As of 2015 both the automatic-generated variables and the user-defined variables are stored as properties in the Tango Database, being the new beamline MIRAS [16] the first in which all standard attributes will be generated from EPS CSV files instead; reserving Dynamic Attributes formulas only for special behaviors. This is done using the AlbaPLC, a PyPLC subclass that implements in the Tango side the data structures of the EPS_LIBRARY.

HUMAN MACHINE INTERFACES TO EPS

EPS GUI's

The current application developed to interact with the EPS system is the EPS Expert GUI (Fig. 4). This application loads all the modbus mapped variables from the PLC and organizes the whole memory structure in several tabbed panels, grouped by variable type.

From the EPS GUI application we can manage, after LDAP user validation, the Alarm/Warning levels of every analog signal, interlock status (active/latched/was_first) for analog and digital inputs, forced and disable states both for interlocks and digital outputs, and the status of the PowerLink word; synchronized in all the EPS cpu's to share interlock signals between them.

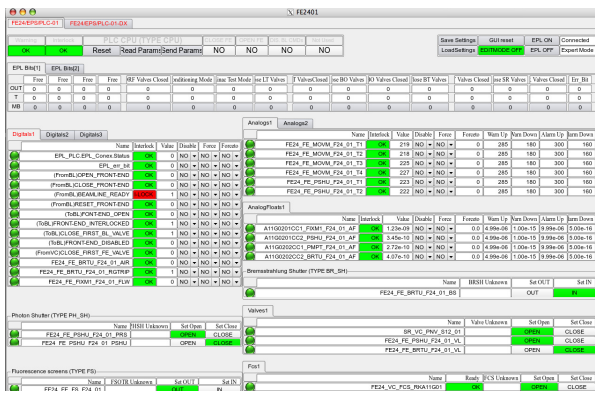


Figure 4: The EPS expert GUI.

The information regarding the several subsystems managed by the EPS (vacuum, magnets, RF, bpm's, front-ends) is later summarized in 2 user-level applications: the ALBA-EPS (that allows fast diagnose and acknowledge of interlocks in the accelerators side) and the Machine Protection System (with specific diagnostic tools in case of BPM or RF interlocks).

Tango, Sardana and PANIC

Many analog and digital signals in ALBA beamlines have been exported to our Tango-based experiment control framework, Sardana. The Sardana suite allows to add Tango controlled hardware as experimental channels, using them as I/O or experimental data (e.g. temperatures and vacuum pressures). In addition, limited motion control have been implemented using PyPLC and dedicated structures in the EPS_LIBRARY, which allowed direct control from Sardana macros or scans developed by scientists.

A similar approach to higher-level procedures involving several Tango devices are implemented using automated actions from PANIC Alarm System[7]. Those macros allow both scientists and PLC engineers to program automated actions on elements controlled by the EPS (valves, shutters) during experiments. These actions do not bypass the EPS, as it is always kept as an autonomous system ensuring the safe state of the installation, but allow the recovery of the system once all the permits conditions are matched.

Figure 5: The VACCA User Interface.

Taurus, VACCA and the EPS User GUI

All those high-level features were enabled by the Dynamic Attributes syntax, that allowed specific interfaces to express intermediate states like warning, external interlocks, complex error codes, attributes managed by multiple registers (e.g. multiple-stage pneumatic elements), etc. Those attributes use Tango Qualities to complement the raw attribute values, passing the information regarding Alarm/Warning/Moving limits and positions with each value sent to clients.

Qualities and state machines become visible in Taurus applications, being converted into colors easy to catch on either control panels or synoptic applications like VACCA [17]. All attributes exported to Tango from the PLC become available to all Taurus widgets (device panels, forms, trends) and can be displayed in tabbed panes as separated elements or in combination with those equipments to be controlled. Those same attributes also allow the user to customize how this information is displayed, either modifying the attribute label or its units and formatting.

This transparent interaction between the Tango and EPS control systems is pushing the need of an standard User-level GUI for the EPS, providing the same information than the Expert GUI but in a way that makes easier to locate and diagnose the causes of an interlock (and the best way to recover). This diagnose feature is being

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

implemented as a PLC source navigator integrated within our current Vacuum Control Application (VACCA), which has been already extended to manage both Archiving and Alarms services.

CONCLUSION

After 8 years of design, installation, commissioning and operation the EPS has become a mature control system by itself, achieving a high level of consistency and reliability.

Its integration within Tango using PyPLC allowed to integrate the PLC systems in all our Tango services. Not only in “passive” ones like Archiving or Alarms, but also as an active part in our experiment-control framework, Sardana.

The next step is the integration of the current tools in an automated cycle of continuous development and delivery. First steps in this direction have been achieved introducing all the PLC's code in SVN repositories, and later migrating to python all the auto-generation tools.

Now, after several years of operation, the two new beamlines under construction at Alba are bringing an opportunity to put all the recent improvements and develop a clean an optimized solution for them. Learning from the previous experience and aiming to refine the new procedures to be later reapplied to the old beamlines.

The current level of integration between Tango and EPS on older beamlines is still much lower than in the accelerators side. This is due to the bigger number of special cases and exceptions while integrating specific devices in the CCDB catalog or the EPS_LIBRARY. It is expected that the new tools as the Auto-generation and the Source navigator will help the PLC engineers to simplify the logics and naming of the existing systems, but there will be always a big percentage of beamline variables too specific to be standardized.

For those cases, enabling all EPS variables as Tango attributes will allow the users to setup and use their own Labels, while keeping the cabling-based tags as attribute names. This kind of mixed naming schema (attribute for control engineers, label for scientists) should reduce the number of naming conventions exceptions and provide enough consistency to close the auto-generation loop.

ACKNOWLEDGEMENT

Many former ALBA engineers have collaborated in the PLC-related projects in the last 6 years: A.Rubio, R.Ranz, R.Montaño, R.Suñé, M.Niegowski, M.Broseta and J.Villanueva.

The collaboration of Tango core developer, Emmanuel Taurel and former ALBA developer R.Suñé was fundamental in the development of Dynamic Attributes templates and debugging of PyPLC performance.

The work of the Generic Software Group at ALBA in the development of Taurus and the Taurus GUI framework has made possible the evolution of the EPS Graphical User Applications.

Last but not least, none of the elements in the auto-generation cycle would have work without the Cabling and Controls Database, developed by the Management and Information Systems section of ALBA.

REFERENCES

[1] ALBA website: http://www.cells.es
[2] TANGO website: http://www.tango-controls.org
[3] A.Götz, E.Taurel et al., “TANGO V8 – Another Turbo Charged Major Release”, ICALEPCS'13, San Francisco, USA (2013).
[4] C. Pascual-Izarra et al., “Effortless Creation of Control & Data Acquisition Graphical User Interfaces with Taurus”, ICALEPCS'15, Melbourne, Australia (2015).
[5] T.Coutinho et al., "Sardana, The Software for Building SCADAS in Scientific Environments", ICALEPCS'11. Grenoble, France (2011).
[6] Z. Reszela et al., “Sardana – A Python Based Software Package for Building Scientific Scada Applications”, PcaPAC'14, Karlsruhe (2014).
[7] S.Rubio et al., “PANIC, a Suite for Visualization, Logging and Notification of Incidents”, PcaPAC '14, Karlsruhe, Germany (2014).
[8] R.Ranz et al, “ALBA, The PLC based Protection Systems.”, ICALEPCS'09. Kobe, Japan (2009).
[9] D.Fernández-Carreiras et al, “Personnel protection, equipment protection and fast interlock systems”, ICALEPCS'11, Grenoble, France (2011).
[10] D. Beltran, et al. “ALBA Control And Cabling Database”, ICALEPCS'09, Kobe, Japan (2009).
[11] S. Rubio-Manrique et al. "A Bottom-up Approach to Automatically Configured Tango Control Systems", ICALEPCS'11, Grenoble, France. (2011).
[12] S. Rubio-Manrique et al., "PyPLC, A Versatile PLC-to-PC python interface", PCaPAC'14, Karlsruhe, Germany (2014).
[13] D.Fernández et al. “Alba, a Tango based Control System in Python”, ICALEPCS'09, Kobe, Japan (2009).
[14] S.Rubio et al., “Dynamic Attributes and other functional flexibilities of PyTango”, ICALEPCS'09, Kobe, Japan (2009)
[15] S.Rubio, et al. “Extending Alarm Handling in Tango”, ICALEPCS'11, Grenoble, France (2011).
[16] MIRAS website: http://www.miras2.es/
[17] S. Rubio-Manrique et al., “Unifying All TANGO Control Services in a Customizable Graphical User Interface”, WEPGF148, ICALEPCS2015, these proceedings.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

UPGRADE OF ABORT TRIGGER SYSTEM FOR Su erKEKB

S. Sasaki#, A. Akiyama, M. Iwasaki, T. Naito, T. T. Nakamura, KEK, Ibaraki, Japan

Abstract

The beam abort system was installed in KEKB to protect the accelerator equipment and the Belle detector, and for radiation safety, from high current beams. For SuperKEKB, the new abort trigger system has been developed. It collects more than 130 beam abort request signals and issues the beam abort trigger signal to the abort kickers. The request signals are partially aggregated in local control rooms located along the SuperKEKB ring and finally aggregated in central control room. In order to increase the system reliability, the VME-based module and the E/O module was developed, and all the abort signals between the modules are transmitted as optical signals. The system also has the timestamp function to keep track abort signal received time. The timestamps are expected to contribute to identify the cause of the beam abort. Based on the feasibility tests with a prototype module, the new module design has been improved and fixed. This paper describes the details of the new abort trigger system.

INTRODUCTION

SuperKEKB is the upgrade of the KEKB asymmetric energy electron-positron collider in Japan [1]. The designed luminosity is 40 times as high as the peak luminosity of KEKB.

SuperKEKB consist of two storage rings, low-energy ring (LER) and high-energy ring (HER). Each ring has a beam abort system to protect the accelerator equipment and the Belle detector, and for radiation safety, from high current beams. The abort trigger system collects more than 130 beam request signals and issues a beam abort trigger signal to abort kickers.

In KEKB, the modules of the abort trigger system were connected with twisted pair cable to transmit the electrical request signals and low pass filters were inserted to reduce electrical noise. The low pass filters caused time delay and the total system response time was about 100 μs. For SuperKEKB, the optical request signals are transmitted and the low pass filters have been removed. The response time is improved to be less than 20 μs.

The new system has the timestamp function to keep track of the abort signal received time. The resolution of the timestamp is 0.1 μs. It is expected to contribute to identify the cause of beam abort.

This paper describes the design and the status of the new abort trigger system.

#shinya.sasaki@kek.jp

SYSTEM DESIGN

Modules

The system is composed of two kinds of modules. First is 2ch E/O module which converts electrical signals from abort request source to optical signals. It can receive 3 kinds of electrical inputs: TTL, RS422, or relay, and can treat the input as active-high or active-low. They are set by slide switches in front panel. Figure 1 shows picture of the 2ch E/O module.

Figure 1: Picture of the 2ch E/O module. Input type and polarity can be set by slide switches (1424).

Second is VME-based 8ch beam abort optical input module (18K15). It gathers 8 optical signals and output an OR optical signal. Since it latches the input signals until reset, pulse input can also be detected and it keeps outputting once request signal is detected. Above functions are processed on FPGA, software is not running on the module. The module can be controlled and monitored via VMEbus. Figure 2 shows picture of the VME-based 8ch beam abort optical input module.

Figure 2: Picture of the VME-based 8ch beam abort optical input module (18K15).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Timestamps

Abort request signals are usually occurred sequentially. Therefore an order of the request signals are important to identify a cause of the beam abort. The new abort trigger system records timestamps when it received the request signal. The timestamps reveal the order of sequential request signal.

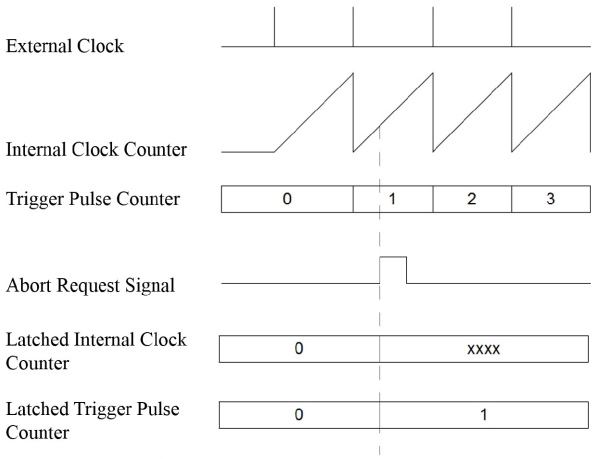


Figure 3: Timing diagram of two counters for timestamps. Internal clock counter counts up at 10 MHz. When the module detects an abort request signal the counters are latched until reset.

The timestamps are generated from two counters, an internal clock counter and a trigger pulse counter, in 18K15. The internal clock counter count up at 10 MHz synchronized to FPGA operation clock. The trigger pulse counter counts an external clock. Each counter is 32-bit and cleared when the module is reset. The internal clock counter is also cleared when the external clock input is detected. After the module reset, each counter starts to count up after the external clock is detected. The external clock synchronize the counters of the all modules in the system. When the module detects an abort request signal the counters are latched until reset. Figure 3 shows the timing diagram of the counter.

The resolution of the timestamp is 0.1 μs since the internal clock counter counts up at 10 MHz. We can notice relative time differences from the counters. However absolute time cannot be calculated without software processing. We can calculate absolute time with the time when an external clock source send a signal.

System Configuration

Figure 4 shows entire system configuration. The electrical abort request signals from equipment are converted to optical signals on 1424 E/O modules. The optical signals are partially aggregated in Local Control Rooms (LCR) located along the SuperKEKB ring and finally aggregated in Central Control Room (CCR). The longest distance from LCR to CCR is 2 km and its transmission time is about 10 μs.

The system needs external clock source to synchronize the timestamp among 18K15s. We use a software trigger system [2] as external clock. It provides the synchronization pulse and/or the interrupt signal to Input/Output Controllers (IOCs) or devices. The time when the software trigger system send a synchronization trigger is recorded on the IOC at CCR.

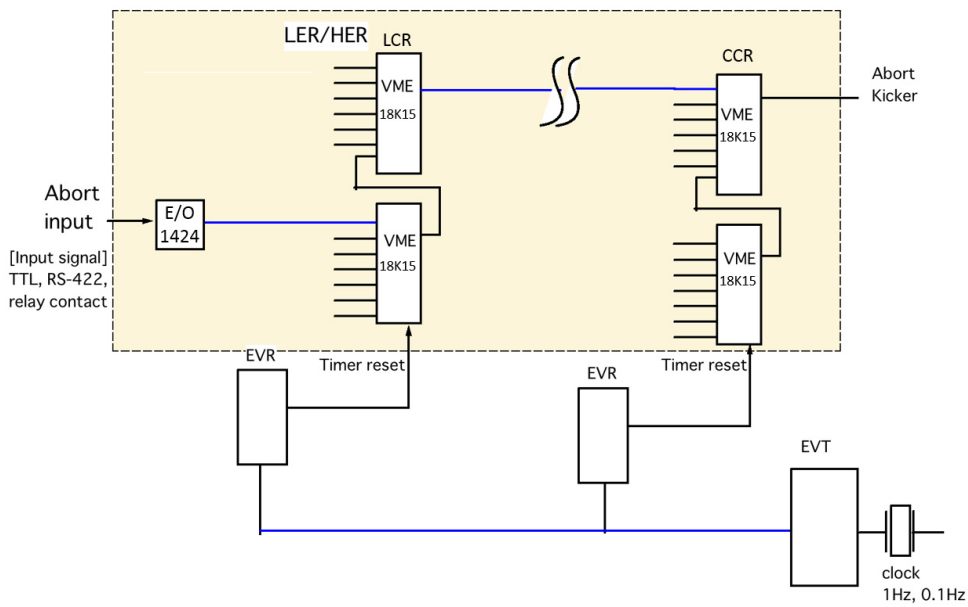


Figure 4: Schematic diagram of the abort trigger system for SuperKEKB. Abort request signals are converted to optical signals on E/O modules and aggregated on VME-based modules. The signals are partially aggregated in LCR and finally aggregated in CCR. The time when EVT (EVent Transmitter module) send a synchronization trigger is recorded on the IOC at CCR.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PERFORMANCE TESTS

Response time

We have tested the response time of 1424 and 18K15. The setup of the test is shown in figure 5. We have measured TTL outputs of the modules equivalent to optical output. Function generator generates pseudo abort request signal which is 500 ns pulse width.

Figure 6 is result of the test. The system detected 500 ns pulsed request signal. It takes about 150 ns for 18K15 to output after E/O module output. This latency is cable delay and processing time of 18K15. Considering that the longest cable delay from LCR to CCR is 10 μs, the latency is sufficiently low and the latency of the abort trigger system mainly depends on the cable delay.

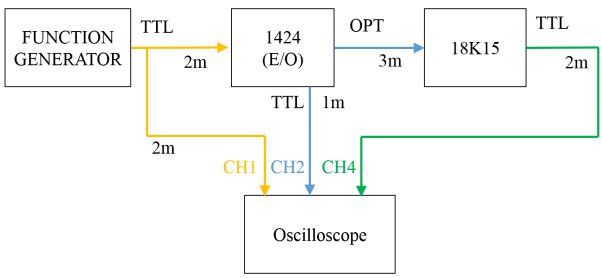


Figure 5: Schematic diagram of the response time measurement.

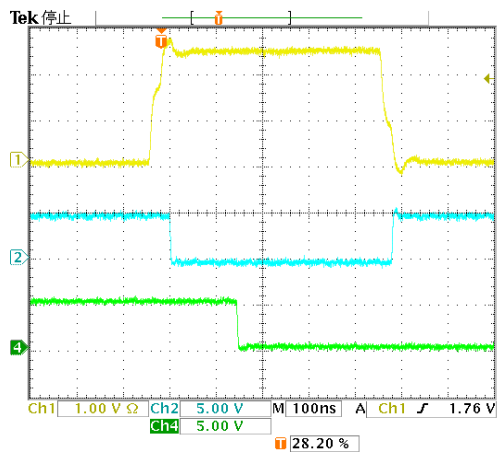


Figure 6: Response of 1424 and 18K15 to 500 ns pulse request signal input. CH1 is pseudo abort request signal generated from Function Generator. CH2 is TTL output equivalent to output of E/O module. CH4 is gathered output on 18K15.

Module control and timestamp

Since the SuperKEKB control system was based on EPICS [3], we have developed EPICS software to control and monitor 18K15.

We have tested the timestamp function with EPICS control system. Abort request signals were input to 18K15 on arbitrary timing and we confirmed latched counters shown in figure 7. The counters revealed an order of abort request signals.

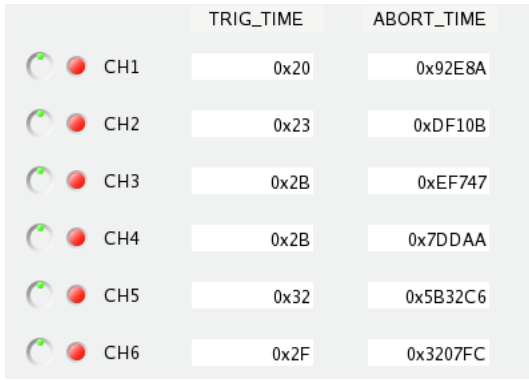


Figure 7: Latched counters of 18K15. TRIG_TIME is the trigger pulse counter and ABORT_TIME is the internal clock counter. The counters show CH1 is the first detected signal.

Figure 8 shows timestamps of time and date format. The timestamps are calculated by software processing based on recorded time which the software trigger system send a synchronization trigger. A relative error among timestamps is several microsecond caused of an error of internal 10 MHz clock on 18K15s and a synchronization error among 18K15s. On the other hand, an absolute error of timestamps is several millisecond caused of an error of the recorded time on IOC at CCR.

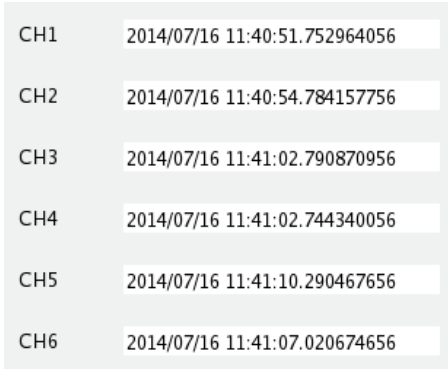


Figure 8: Timestamps of time and date format calculated by software processing.

CONCLUSION

We have developed the new abort trigger system for SuperKEKB. The total response time of the system significantly depends on cable delay and it is improved to be less than 20 μs. The system can record timestamps when it receives abort request signals. Timestamps reveal the order of sequential request signals.

REFERENCES

[1] Y. Ohnishi et al., “Accelerator design at SuperKEKB”, Prog. Theor. Exp. Phys., 2013, 03A011.JACoW.org

[2] T. Naito et al., "Performance of the timing system for KEKB", ICALEPCS’99, Italy, Oct. 4-8, 1999

[3] http://www.aps.anl.gov/epics

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

DEVELOPMENT OF A NETWORK-BASED PERSONAL DOSIMETRY SYSTEM, KURAMA-micro

M. Tanigaki, Research Reactor Institute, Kyoto University, Kumatori, Osaka 590-0494, Japan
Y. Nakanishi, Electronics Laboratory, Shikoku Research Institute Inc.,
Takamatsu, Kagawa, 761-0113, Japan

Abstract

KURAMA-micro, a personal dosimetry system with network and positioning capabilities, is developed for the continuous monitoring of radiation exposure of individual in a large group, based on their action histories. Typical target users are the residents returning to their hometown after decontamination, and the workers involved in the decomissioning of the Fukushima Daiichi nuclear power plant. A KURAMA-micro unit consists of a semiconductor dosimeter and a compact DAQ board with a GPS module and a ZigBee module. Each unit records radiation data tagged with their measurement time and locations, and uploads the data to the server over a ZigBee-based network once each unit comes near the access point installed in the area of daily activities of users. Location data are basically obtained by a GPS unit, and an additional radio beacon scheme using ZigBee broadcast protocol is also used for the indoor positioning. A field test of the prototype of KURAMA-micro was performed in Fukushima city, and the radiation trends were successfully observed.

INTRODUCTION

The magnitude-9 earthquake in eastern Japan and the following massive tsunami caused a serious nuclear disaster of Fukushima Daiichi nuclear power plant. Serious contamination was caused by radioactive isotopes in Fukushima and surrounding prefectures. KURAMA [1] and KURAMA-II [2] were developed to overcome the difficulties in radiation surveys and to establish air dose-rate maps during and after the present incident. KURAMA has been successfully applied to various activities in the radiation measurements and the compilation of radiation maps in Fukushima and surrounding areas. KURAMA-II, an advanced version of KURAMA with autonomous operation and pulse height spectra measurement, has been used to establish the continuous monitoring scheme in residential areas in Fukushima prefecture. Fifty local buses and official cars equipped with KURAMA-II have been continuously operated in Fukushima prefecture, and radiation maps have been released to the public on weekly-basis. The results from KURAMA-II also contribute the predictions of air dose rates of the future in Fukushima.

Along with such wide-scale monitoring, the importance of more personalized monitoring intending to track the actual exposure of individual to the radiation are recognized to clarify the dependence of radiation exposures to human activities. For example, it is well known that the indoor radiation exposure becomes lower than outdoors, but the time length of stay indoors/outdoors vary depending on one's occupation, age, place to live etc. To clarify what kind of activities cause the higher radiation exposure is important to establish effective measures to minimize the radiation exposure. As the recovery from the nuclear accident in Fukushima progresses, strong demands of such personalized monitoring arise for a large group with higher risk of radiation exposure, such as the residents returning to their hometown after decontamination, or the workers involved in the decomissioning of the Fukushima Daiichi nuclear power plant.

For such purpose, personal dosimeters that measure the cumulative radiation dose, such as film badges, electronic pocket dosimeters, have been widely used, but the lack of positioning and time information made difficult to analyze in the view point of the human activities. Additionally, the difficulty in collecting data periodically from a large number of people obstruct the analysis of such data to extract the general tendency of radiation exposure towards human activities.

KURAMA-micro, a personal dosimetry system with network and positioning capability, is developed for such purpose. KURAMA-micro consists of a semiconductor dosimeter and a DAQ board with GPS and ZigBee modules. The data of KURAMA-micro is collected via an ad-hoc ZigBee based network to minimize the troubles of users. The development of a prototype KURAMA-micro is finished and field tests are on the way. In this paper, the outline of KURAMA-micro and the results from a field test in Fukushima are introduced.

SYSTEM OUTLINE OF KURAMA-micro

KURAMA-micro stands on a similar concept as claimed by KURAMA/KURAMA-II, and intends to be the alternative choice of conventional personal dosimeters. Therefore, the system must overcome the difficulties arising from its usage. For example, the unit must be comparable in size to other personal dosimeters, and the operating time should be long comparable to the replacement period of conventional film badges, typically one month. Therefore KURAMA-micro is designed to achieve better performance on power-consumption or sufficient compactness rather than on the detection efficiency or the measurement precision.

KURAMA-micro consists of the radiation detection part and the DAQ part (Fig. 1). The radiation detection part is a conventional accumulated radiation dosimeter with I2C connection to the DAQ part. The DAQ part has an I2C interface for the radiation detection part, a GPS unit for positioning, and a ZigBee unit for the data communication.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

This ZigBee unit is also used for a supplemental positioning where the radio waves from GPS satellites are not available.

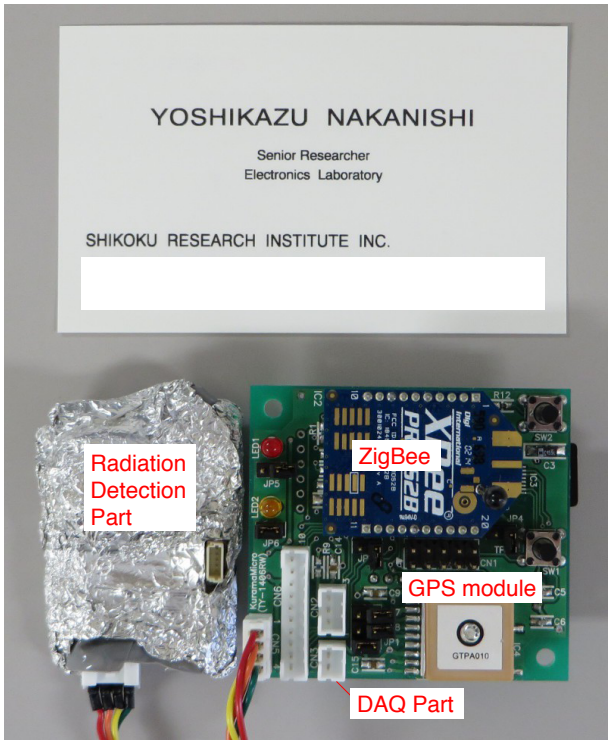


Figure 1: A prototype KURAMA-micro unit. The board on the right is the DAQ part based on OpenATOMS architecture. The left part shielded by aluminum foil is the radiation detection part. Both are implemented into a box of business card size. The power required for the unit is managed by a 3.7 V rechargeable Li-ion battery.

The radiation detection part of KURAMA-micro consists of a typical semiconductor detector, charge-amplifier, shaping amplifier, and a microprocessor with ADC capability. A conventional photodiode (Hamamatsu S6775) is used as a semiconductor detector. The charge amplifier and shaping amplifier are configured by using Smart Analog series of Renesas Electronics [3]. Smart Analog is a programmable analog IC designed for processing small sensor signals. This IC includes several amplifiers with programmable gain, general-purpose operational amplifiers, high-pass and low-pass filters, and DACs, and those connections and operation modes are programmable. Additionally, Smart Analog has several low power consumption modes, in which the circuits in the IC are operated intermittently or turned off partially upon user's settings. Such low power mode is useful for the devices required for the low-power operation such like KURAMA-micro. Each pulse from the Smart Analog IC is sent to ADC in the microprocessor, then accumulated as a pulse height spectrum. The accumulation period lasts until the request for air dose from DAQ part comes, then the accumulated air dose is calculated from this pulse height spectrum by G(E) function method [4].

The DAQ part of KURAMA-micro is based on OpenATOMS(open Advanced Topological On-demand Monitoring System) [5] developed by Shikoku Research Institute Inc. OpenATOMS is a platform developed for sensor networks in field, especially for the cases that the continuous monitoring is required for a long period under the lack of sufficient power supplies or a reliable network communication. In this platform, each sensing device is connected to a microcomputer board named as NICE(Networked Intelligent CEll). One of the features in NICE is the low power consumption by an advanced power management scheme in which the power of most part in NICE can be switched on/off by the embedded timer. With this feature, a NICE board can be operated for more than a month by a conventional dry cell. Each NICE board joins an ad hoc ZigBee network maintained by a local gateway server connected to a TCP/IP network. All the data are once collected and stored in a local server as an XML based format, then extracted by remote users or data servers upon request. Typical applications of OpenATOMS are the monitoring the ground parameters to detect landslides in mountainous areas, or the condition in hothouses for agricultural products.

These features of OpenATOMS almost satisfy the requirements of KURAMA-micro. The power management system implemented in NICE is quite useful for the minimization of power consumption in KURAMA-micro, and the data management vis the ad hoc network based on ZigBee realizes the data collection without the manipulation of users. But there are still challenges. One of the challenges in the case of KURAMA-micro is that the units do not always join the ZigBee network because each unit always move around the residential area along with the carrier. This results in a large power consumption if the continuous seek of an access point is simply allowed. Therefore, the seeking time of the network is limited based on the activities of the people carrying KURAMA-micro by using the power management capability of NICE. For example, the seeking time is limited to the daytime of weekdays in the case of schoolchildren, expecting the attendance of their classrooms and the connection to the access points placed at the classrooms.

Another challenge in KURAMA-micro is the positioning scheme where the radio waves from GPS satellites are not available, mainly indoors and underpasses. This time, radio beacons using ZigBee broadcast protocol are used as the alternative positioning for this purpose. These ZigBee-based beacons are settled place by place, and the location of the measurement is treated as that of received beacon. This scheme is not only enables the approximated positioning in indoors and underpasses, but also effective to improve the accuracy of the height information in buildings to distinguish the difference of radiation exposure in floor level. The accuracy in height is generally very poor in GPS, poor enough not to distinguish the floor level.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

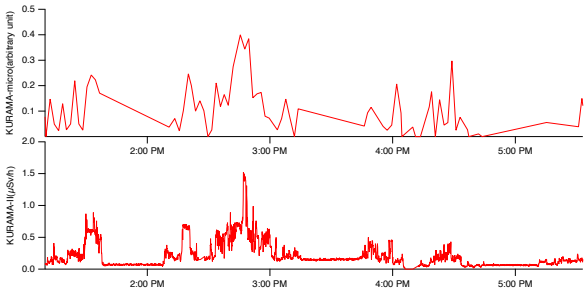


Figure 2: The time-domain trend of air dose rate obtained by KURAMA-micro(upper graph) and that obtained by KURAMA-II(lower graph) for comparison. The measurement period for KURAMA-micro and KURAMA-II are one minute and three seconds, respectively. The air dose rate from KURAMA-micro is described in arbitrary units because the absolute calibration for the unit is not performed at that time.

TEST OPERATION IN FUKUSHIMA

To confirm the concept of KURAMA-micro, a field test of a prototype KURAMA-micro carried by an examinee was performed in the center area of Fukushima city. The purpose of this field test was to examine whether KURAMA-micro actually tracks the radiation exposure to a person under activities of daily living. During the field test, the examinee strolled around the center of Fukushima city, mainly outdoors and occasionally inside the building. The examinee was accompanied by a person measuring the air dose rate by KURAMA-II for comparison. The measurement period of KURAMA-micro was set to 1 minute instead of 10 minutes for detailed analysis of the effect from surrounding condition to the radiation measurement.

The radiation trends of KURAMA-micro and KURAMA-II in time domain is shown in Fig. 2. A moderate correlation was found between the results of both systems. Some differences observed in the trends were caused by "micro hot spots" that were occasionally observed in Fukushima city, such as remaining contamination at hedges along the road. Such small hot spots produce a localized radiation field, which often cause the difference on the air doses of examinee and the accompanied person. Other differences were due to the improper sequence implement in this prototype of KURAMA-micro. DAQ board was programmed not to request the radiation data to the radiation detection part until the positioning had been determined, so the air dose data was often kept accumulated beyond the normal request time period. The positioning by GPS frequently failed due to the buildings along the street, therefore the request period tended to be longer than that was set.

The radiation map generated from the result of KURAMA-micro was compared with that from the result of KURAMA-II units on local buses(Fig. 3). In the central area of Fukushima city, a radiation map with the grid size of 100 m × 100 m is generated by KURAMA-II units on local buses on a weekly basis [6]. Through the averaging procedure of KURAMA-micro data to 100 m × 100 m grid, differences caused by "micro hot spots" and intermittent failures of GPS were averaged. These maps are similar except several regions, and these were mainly identified as the effect of measurement points such as inside the building or the underpasses of large roads.

CURRENT STATUS AND FUTURE PROSPECTS

Currently, KURAMA-micro is under development, mainly for the reduction of interference by incoming noise. The reduction of such noise is crucial for the semiconductor detectors since they usually have a high impedance charge amplifier in their front end. As soon as the sufficient noise reduction is achieved, several extended field tests will be performed. Along with the noise reduction, tunings for the power reduction is on the way, such as the optimization in the power management of the GPS module. As for the current prototype KURAMA-micro, the operation time reaches around three weeks with 3.7V 3100 mAh battery.

One of the expected applications of KURAMA-micro is the personal dose monitoring of school children(Fig. 4). In this application, a KURAMA-micro carried by a schoolchild collects the radiation data and positioning data throughout the day. The positioning data is basically collected by a GPS unit in KURAMA-micro, and radio beacon units using ZigBee broadcast protocols are placed in the public buildings in the school district and in houses of school children. Such radio beacons are expected to be placed easier because the area of the activities of school children is mainly in the school district. The frequency of data collection is planned 10 min, to reduce the power consumption and not to disclose the activities of school children too much from the viewpoint of privacy. The data accumulated by each KURAMA-micro unit is collected via the access point in classrooms of their school while schoolchildren stays in their school. The col-

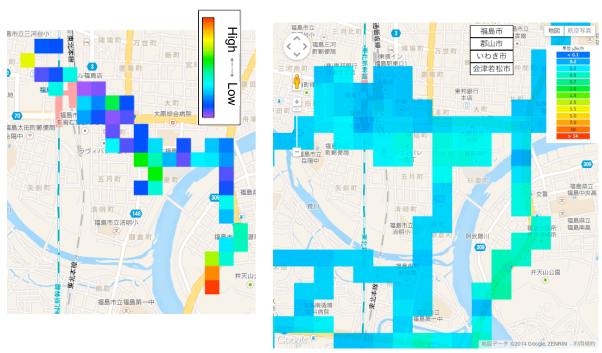


Figure 3: The air dose rate map from the data of KURAMA-micro(left side) and that from KURAMA-II on local buses(right side) at the same measurement period. The values were averaged for every 100 m × 100 m. The air dose rate from KURAMA-micro is described in arbitrary units because the absolute calibration for the unit is not performed at that time.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

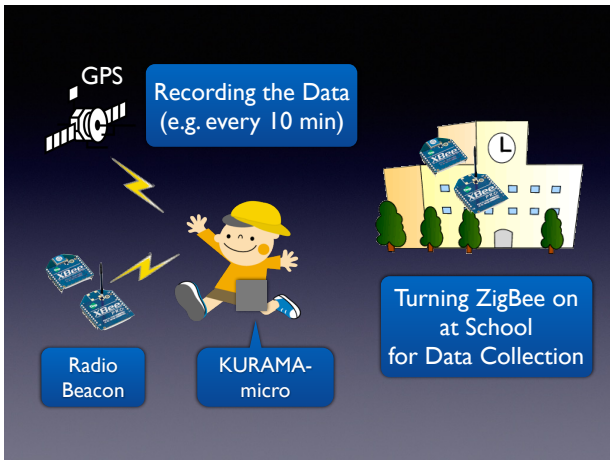


Figure 4: An expected operation of KURAMA-micro.

lection time for all the data of schoolchildren in a class is estimated to be several hours by taking into account the transmission speed of ZigBee, almost the same as the average staying time of school children in a classroom. KURAMA-micro units are replaced every month, as that in the case of other conventional dosimeters like film badges.

ACKNOWLEDGEMENTS

Authors are grateful to Mr. Yamasaki, Mr. Ishii, Mr. Takagi at Shikoku Electric Power Co., Inc. for the continuous support on the management of this development project. The author(M. T.) is also grateful to Mr. Matsumoto and Mr. Dozono at PI system Co., Ltd. for the discussions on the circuit design of the radiation detection part.

REFERENCES

[1] M. Tanigaki, R. Okumura, K. Takamiya, N. Sato, H. Yoshino and H. Yamana, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **726** (2013) 162 .

[2] M. Tanigaki, R. Okumura, K. Takamiya, N. Sato, H. Yoshino, H. Yoshinaga, Y. Kobayashi, A. Uehara and H. Yamana, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **781** (2015) 57 .

[3] `http://www.renesas.com/products/smart_analog/index.jsp?campaign=tb_prod`

[4] S. Tsuda and M. Tsutsumi, Jpn. J. Health Phys. **47** (2012) 260 (Japanese).

[5] Y. Nakanishi, Institute of Electronics, Information, and Communication Engineers **94** (2011) 852 (Japanese).

[6] The radiation maps are periodically released as a collaboration project among Kyoto University, Fukushima prefecture, and Japan Atomic Energy Agency from the following website: `https://www.pref.fukushima.lg.jp/sec/16025d/soukou.html`(Japanese). Kyoto University also releases the radiation maps of four major cities in Fukushima by their own KURAMA-II on local buses from the following website: `http://www.rri.kyoto-u.ac.jp/kurama/kouiki/kurama2_test.html`(Japanese).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INTEGRATION OF HETEROGENEOUS ACCESS CONTROL FUNCTIONALITIES USING THE NEW GENERATION OF NI cRIO 903X CONTROLLERS

F. Valentini, T. Hakulinen, L. Hammouti, P. Ninin
CERN, Geneva, Switzerland

Abstract

Engineering of Personnel Protection Systems (PPS) in large research facilities, such CERN, represents nowadays a major challenge in terms of requirements for safety and access control functionalities. PPS are usually conceived as two separate independent entities: a Safety System dealing with machine interlocks and subject to rigid safety standards (e.g. IEC-61508); and a conventional Access Control System made by integration of different COTS technologies. The latter provides a large palette of functionalities and tools intended either to assist users accessing the controlled areas, either to automate a certain number of control room operator's tasks. In this paper we analyse the benefits in terms of performance, cost and system maintainability of adopting the new generation of NI multipurpose cRIO 903x controllers. These new devices allows an optimal integration of a large set of access control functionalities, namely: automatic control of motorized devices, identification/count of users in zone, implementation of dedicated anti-intrusion algorithms, graphical display of relevant information for local users, and remote control/monitoring for control room operators.

INTRODUCTION

The GS/ASE group is responsible for the realization and the maintenance of Personnel Protection Systems ensuring the safety of personnel accessing to the various accelerator complexes of CERN. The group was responsible for the conception of a new generation of *access points* [3] to gradually replace the traditional rotating gates, some of them aged of few decades.

Even though the different system functionalities were clearly identified and specified since the beginning of the project, the realization strategy favored a construction by the integration of off-the-shelf (COTS) products rather than to privilege any form of custom development. This choice was mainly motivated by the limited internal resources, making it necessary to externalize the activity to a system integrator, and by the fact that the various functionalities to be provided were already present on the market with well-known and robust commercial solutions. Another motivation was the conviction that constructing a new system out of modular simple blocks would be more easy and rapid than to start a development from scratch. However, in our experience, it turned out that dealing with COTS can be quite a risky activity for a wide variety of reasons.

In [1], presented in the *International Conference on Software Engineering* of 1995, Garlan, et al., introduces the term of *architectural mismatch* to explain the different problems inherent in integration of COTS software components. Typically, many problems are caused by incompatibilities between the programming languages, the operating platforms or the database schemas of the products being integrated. The authors argue that at the origin of any architectural mismatch problem there is the fact that COTS components make several assumptions about the system architecture and its operational environment; consequently, when these assumptions conflict or do not match with each other, the simplicity of assembling complex systems from integration of COTS immediately disappears.

The biggest challenge for designing an access point turned out to be the integration of numerous heterogeneous components such as PLCs to command the movements of the motorized doors, *front-end* to deal with the real-time authorization management, *biometry* for authenticating users, *video system* for remote surveillance of the areas, *public address* system for broadcasting of audio messages, and many other devices. In this context our team is constantly seeking new solutions and technologies allowing to minimize the number of CPUs and information that need to be handled and exchanged between the different subsystems.

This paper will especially focus on the impact analysis that a completely COTS oriented realization strategy had on the architecture of the first series of our access points and it highlights the actual interest in the National Instruments technology [5] as a possible solution for the implementation of more rational system architectures.

THE ACCESS POINT CONCEPT

This section describes the ideas and the concepts behind any access point at CERN. Many would claim that an access point is simply made of "*one door, equipped with expensive position sensors, a video camera and a microphone allowing remote connections with guardians or operators*". However it is much more than that. An access point in a facility like CERN plays a crucial role in defending the installation against intrusions and accesses of people not holding the security requirements. At the same time, it ensures the safety of personnel every-day working inside the potentially dangerous areas.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Figure 1: Standard configuration of an Access Point.

In order to accomplish its double mission, an access point (see Fig. 1) has to provide a certain number of important functionalities:

- Personnel Access Supervision: a dedicated device (PAD) is in charge of regulating the entrance of personnel. The device is an air-lock closed by 2 motorized doors and equipped with redundant anti-fraud detection systems dealing with all typical situations of piggybacking and tailgating intrusions.
- Material Access Supervision: an additional specific device (MAD) is in charge of controlling the transfer of bulky material to and from the controlled areas. The device is made of two motorized doors that cannot be opened at the same time; the internal volume is then surveilled by two human detection systems based on motion detection and video image analysis algorithms, in order to reduce the possibility for a person to gain easy access to the zone.
- Dynamic Information Dispatch: personnel accessing the controlled areas needs to be dynamically informed about the safety conditions inside the zone. The information system is made up of two independent components: a group of LEDs for critical safety warning visualization, directly animated by a PLC; and a standard local screen hosting a graphical user interface. This latter shows information about the operational state of the zone and it drives users though the complete access procedure.
- User Identification: in order to access the controlled areas every user must be identified and his access privileges must be checked against the central authorizations database. The identification of users is done via an RFid badge reader and a biometric eyes scanner.
- Personnel Safety Tokens Distribution: as additional protection, every accessing user is provided with a safety token that is automatically released by a distributor. The possession of the token ensures that beam operations cannot be reestablished.
- Interphone System: an audio/video communication system allows local users to contact the CERN control room operators.
- Remote Maintenance: due to the large number of access points installed all over the CERN complex, the maintenance team needs to be able to perform a certain amount of operations remotely. Extra functionalities were added afterwards in order to power off individuals controllers, access to detailed diagnostic and operational data or to send specific commands to the different electronic devices.

The resulting architecture is the outcome of a process of integration of different hardware and software subsystems, each one basically in charge for realizing a specific well determined function. Just to give an idea, the main subsystems controlling an access point (LHC type) are:

A control unit (UTL) for PAD: the controller, based on a proprietary OS, is in charge of acquiring the RFid badge ids, verifying the users' credentials in a central database and, additionally, for commanding the PAD motorized doors movements.

A Siemens PLC S7-200 for PAD motors: this PLC is provided by the PAD supplier and it is used to directly drive the door motors according to UTL commands.

A control unit (UTL) for the MAD: the controller, based on a proprietary OS, is dedicated uniquely to command the MAD motorized doors.

A microcontroller for MAD motors: this dedicated electronics, Microchip based technology, is provided and programmed by the MAD supplier and it is used to physically drive the door motors according to UTL commands.

Two control units for biometric: one Linux based PC is in charge of the processes related to the eye scanned image, while another dedicated unit simply checks the authorization on the biometric database and communicates the user identified to the PAD's UTL. This info is sent then to the main control front-end that send it back down to the information screen controller for displaying, locally, the name of the user.

A microcontroller for the key distributor: the distributor is an industrial product coming with its own electronic control system to handle all operations on the keys.

A standard Windows PC: another PC is used to run a specific commercial SCADA software that, communicating with a central front-end, displays all live information for local users. Additionally, this PC runs the detection software that is used to analyze the material passed though the MAD.

A Siemens S7-1200 PLC: this was added during the operational phase to perform simple remote maintenance commands. This was the only practicable solution because of the complexity of modifying any of the other existing controllers at this scope.

An I/O remote Ethernet module: the module was added to allow to the MAD video analysis software to acquire

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

the state of the MAD door, since it was not possible to extract this information from the MAD UTL.

MAIN PROBLEMS ENCOUNTERED FOR THE ACCESS POINT DESIGN

What appears the most shocking in this architecture is that basically every functionality of an access point is realized via one or more industrial controllers. Very frequently these controllers are made by powerful electronics units, equipped with sophisticated microprocessors, modern operating systems and provided with data communication protocols (Ethernet, serials RS 485/422, etc.) as with digital or analog input output interfaces. So, in principle, almost any of the controllers could be programmed to execute the whole set of functions instead of being underutilized and, despite all this, the implementation of some basic functions (as displaying on the local screen the name of last identified user) has turned out to be extremely difficult.

Another critical point is represented by the system maintainability due to the large number of access points actually in operation. Considering a total of 56 units between the LHC and PS with an average of 10 critical controller each, estimating a gross MTBF of 3.4 years for every component, an average time to repair of 3 hours at two technicians: this leads to a total time of **987** working hours per year only for dealing with failures. Additionally to that, extra costs for preventive maintenance (e.g. installation of Windows security patches) have to be considered.

From our point of view many difficulties that are found during the integration of commercial COTS are caused by typical architectural mismatch problems. The attention to a certain number of factors and initial assumptions is essential to avoid a large amount of problems:

1. *Assumptions about the nature of components.*
2. *Evaluation of the communication needs between the different components.*
3. *Flexibility and evolution requirements.*
4. *Maintainability requirements.*

After the return of experience obtained by the design, the construction and the maintenance phases over the past 8 years [3], many efforts have been made in order to find new concepts and methodologies allowing to simplify and homogenize our systems. A second more robust generation of access points have been recently put in operation to regulate all accesses of the PS accelerators complex. Despite that the hardware and software architecture has been entirely reviewed and that the final product presents evident improvements [4], a certain complexity is still affecting the system. The heterogeneity of the various functionalities still requires that a large number of controllers and software products operate together.

CURRENT PRESPECTIVES

Currently, a promising compromise solution between integration of existing components and custom development is offered by the *National Instruments* (NI) technology [5]. This especially thanks to the synergy offered by two NI products: the new generation of Compact RIO 903x controllers and LABVIEW as software development environment.

The typical CompactRIO system is a combination of a real-time controller, reconfigurable I/O Modules and an FPGA chip (see Fig. 2). The real-time target includes an Intel microprocessor for implementing network communication, data logging, control algorithms, and it provides support for deterministic execution thanks to the presence of a real-Time OS. The FPGA module, from Xilinx, is integrated in the same chassis of the microprocessor but it is completely independent; it is commonly used to implement high-speed controls, inline data processing, or complex timing and triggering operations, but also to realize interlocks and digital signals logics [2].

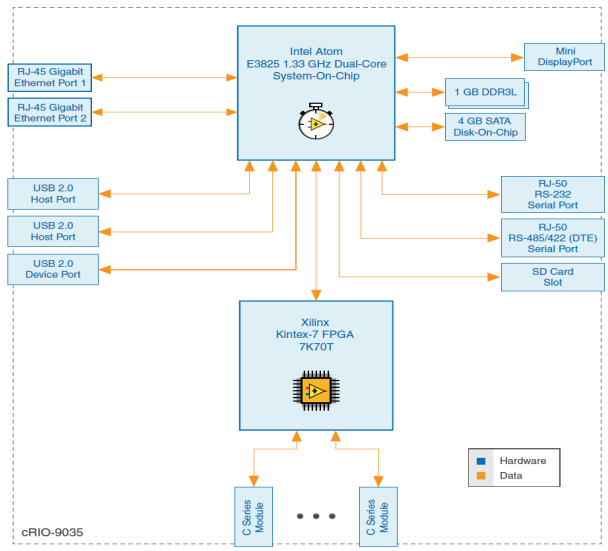


Figure 2: NI cRIO RT & FPGA internal schematics.

The new generation of Compact-Rio controllers (903x family) are particularly interesting due to the fact that NI replaced their proprietary OS with a new version based on Linux Real-Time. This adds much more flexibility to the controller; on one side allowing users to run their own software in parallel with the LABVIEW applications; on the other side it offers the possibility to interface the C-Rio with a wide palette of devices such as: USB keyboard, USB mouse, video monitors, smart cameras, etc. as is usually done in any standard Linux machine.

The second element that makes it possible to employ the cRIO controller to a wide palette of application is offered by LABVIEW. It consists of a graphical software development environment allowing one to construct complex software applications by simply composing and connecting together "*ready-to-use*" blocks. The predefined palette of connectable blocks is very wide and it contains solutions for the most common exigencies of software developers, as: file management, database connections, OPC communications, physical I/O controls, image pro-

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

cessing, complex vector/matrix manipulation, mathematical analysis, etc.

SOME INITIAL FEEDBACKS

At the present time our team is studying the possibility to drastically reduce the complexity and the amount of electronics required for access point control by adopting the NI cRIO 903x as unique controller. A real scale prototype is under construction, but we have already some feedback on a certain number of core functionalities that were developed in LABVIEW and tested on a cRIO 9030.

Displaying of local information HMI: the new cRIO 903x series is equipped with a HDMI video output port that is perfectly suitable to drive a local screen. Some test GUI applications have been easily realized via LABVIEW in order to test the large palette of graphical components available.

OPC UA data server / remote supervision: the OPC UA server is the solution adopted to make the cRIO communicate with the external world. LABVIEW allowed us to realize a first test application by just connecting a total of 11 predefined blocks to implement an OPC UA server. This sends periodically monitoring data, computed or acquired by the controller, to Technical Infrastructure Monitoring system, that is one of the main SCADA supervision tools for the operators of CERN control room.

Connection to Oracle databases: connections to the authorizations database are performed in order to verify the users' access privileges and to verify a certain number of conditions (e.g. valid activity permit). These functionalities, currently realized by a dedicated electronics (UTL), can be easily programmed in LABVIEW in order to be executed in the cRIO. This is realized thanks to the database toolbox inside the LV *connectivity* package. An interesting possibility to be investigated is the installation, directly on the cRIO Linux real-time target, of a local database instance, anyone suitable for Linux environment (e.g. MySQL). This would allow to store locally a copy of the relevant central database tables so as not to be affected by network cuts or other problems.

Image processing & intrusion detection: the software for video analysis in charge of detecting unauthorized presence of personnel inside the Material Access Device (MAD) has been rewritten in LABVIEW in order to be executed directly on the cRIO real-time target. The exercise was particularly interesting because, thanks to the richness of functionalities offered by the image processing toolboxes, we obtained a very simplified implementation of the software. Contrary to the actual C++ implementation, the LABVIEW version of the software meets much more closely maintenance and evolutionary requirements.

Badge RFid acquisition: Tests to control a STid RF badge reader have been successfully performed using the VISA objects inside the *Data Communication* tool box. The model of RFid reader in use at CERN is of type ISO 15693 (13.56 MHz) and connected via a serial RS485 link. The simple application developed in LABVIEW showed that the control of a standard CERN badge reader was possible and very easy to realize, simply via the connections of a few blocks.

Additional functionalities providing the physical control of PAD and MAD motorized doors, of key distributors and various other devices have to be still implemented, furthermore, performance and robustness tests have to be conducted during several mounts.

CONCLUSIONS

The choice between a fully COTS assembly activity against a more optimized integration, including *in-house* developments, is driven by several evaluation criteria, of which the principals are in relation with:

- Evaluation of available internal resources;
- Time to develop, document and fully qualify certain functionalities;
- Level of complexity of the system in terms of HW/SW components and their interconnections.
- Sustainability of annual maintenance costs.

The REX, gained after several years of system integration, has however highlighted that to gain in maintainability and evolutivity a simple COTS integration may be not sufficient and new concepts or more advanced integration tools needs to be considered. The exigencies for obtaining a system capable of performing heterogeneous functionalities being, at the same time, flexible, capable to host new functional requirements, easy to maintain and to develop prompted us to consider another integration principle, based on National Instruments solutions, for the next generation of access points.

The present prototyping activity showed that the utilization of the CRIO 903x controllers, in combination with LABVIEW development environment, could be suitable for our needs. Although more data is essential to better evaluate performances and robustness of this technology, the first results obtained with NI are encouraging to pursue this investigation.

REFERENCES

[1] D. Garlan et al., "*Architectural Mismatch or Why it's Hard to Build Systems out of Existing Parts,*" Proc. 17th IEEE/ACM, Seattle (USA) 1995.

[2] T. Hakulinen et al., "*Building an Interlock: Comparing Technologies for constructing Safety Interlocks*," MOPGF143, Melbourne (AU), these proceedings.

[3] T. Ladzinski et al., "*Access Safety Systems - New Concepts from the LHC Experience*," ICALEPCS11, Grenoble (FRANCE), WEPMU008, 2011.

[4] P. Ninin et al., "*Refurbishing of the CERN PS Complex Personnel Protection System*," ICALEPCS13, San Francisco (USA) 2013.

[5] http://www.ni.com

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

COMMISSIONING AND DESIGN OF THE MACHINE PROTECTION SYSTEM FOR FERMILAB'S FAST FACILITY*

A. Warner#, L. Carmichael, R. Neswold, J. Wu, N. Liu and D. Crawford FNAL Batavia, IL 60510, U.S.A

Abstract

The Fermilab Accelerator Science and Technology (FAST) Facility will provide an electron beam with up to 3000 bunches per macro-pulse, 5Hz repetition rate and 300 MeV beam energy. The completed machine will be capable of sustaining an average electron beam power of close to 15kW at the bunch charge of 3.2nC. A robust Machine Protection System (MPS) capable of interrupting the beam within a macro-pulse and that interfaces well with new and existing controls system infrastructure has been developed to mitigate and analyze faults related to this relatively high damage potential. This paper describes the component layers of the MPS system, including a FPGA-based Permit Generator and Laser Pulse Controller, the Beam Loss Monitoring system design as well as the controls and related work done to date.

INTRODUCTION

The FAST Facility comprises an electron injector based on the Advanced Superconducting Test Accelerator (ASTA)[1], a radio frequency quadrupole (RFQ) based proton injector and the Integrable Optics Test Accelerator (IOTA) storage ring. The electron beam is produced by a 1.3 GHz RF photo-injector and then accelerated to ~50 MeV by two 1.3 GHz SRF cryomodules, each containing a single 9-cell cavity. The beam will then be injected into the linear accelerator which consists of a 12-m long, 1.3 GHz 8-cavity superconducting cryomodule (CM2). This is a Tesla type III+ cryomodule[2] driven by a 5 MW klystron. The electron beam energy gain will be approximately 300 MeV at this stage.

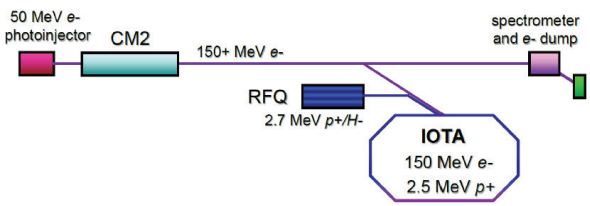


Figure 1: Schematic of FAST Facility.

The facility is also being expanded to accommodate further advanced accelerator research and development with the installation of a 2.5 MeV proton/H- RFQ accelerator. This accelerator starts with a 50 kV, 40 mA proton (or H-ion) source coupled to a pulsed 325 MHz RFQ to 2.5 MeV with a 1ms pulse duration for injecting into IOTA. This ring is 39 meters in circumference and will also be capable of storing electrons from 50 MeV to 150 MeV in energy. Figure 1 shows the placement of the ring in the FAST facility layout.

The Machine protection System (MPS) is being developed in stages that are commensurate with the commissioning goals for FAST. The primary objectives from the MPS point of view are to mitigate beam induced damage to the machine components and to provide a comprehensive over-view of the entire accelerator based on the input status of all the relevant subsystems [3].

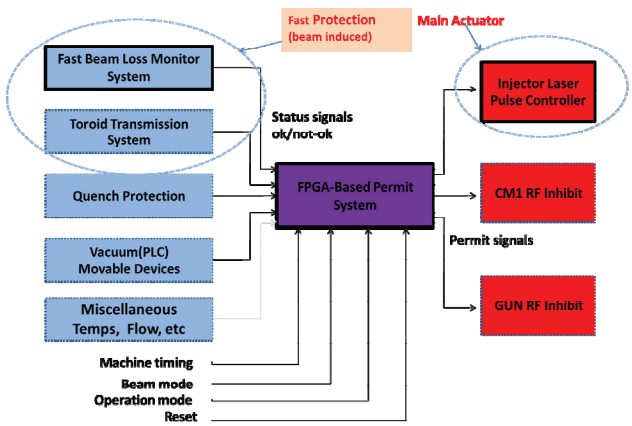


Figure 2: MPS Overview.

Figure 2 illustrates the overall MPS design which is divided into 3 layers; a sensor layer to collect sub-system status, a process layer that utilizes the status to generate the permits and an actuator layer to receive the permits and inhibit the beam. The initial stage of this development involved the design of the Laser Pulse Controller (LPC).

LASER PULSE CONTROLLER

The LPC is designed to be the primary actuator for beam inhibits. Its main function is to provide a gate to the gun laser system via the Pockels cells with a width that corresponds to the total number of 3 MHz pulses allowed without crossing the programmable threshold for losses. The maximum width of the gate is 1 ms which would accommodate a maximum of 3000 bunches. It is designed to inhibit the system within the 1ms macro-pulse window. The response time of the system has been demonstrated to be << 1 microsecond corresponding to less than 3 bunches in the machine. The LPC also provides a number of timing channels to the accelerator complex including a pulse which corresponds with the arrival of the first bunch; so-called first bunch trigger. This is an adjustable trigger delay with < 40 ps of jitter and tuneable to steps of

*Operated by Fermi Research Alliance, LLC, under Contract No. DE-AC0207CH11359 with the United States Department of Energy
#warner@fnal.gov

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

100ps. The laser itself operates continuously at 3 MHz with 200 fs of intrinsic jitter. The LPC is built on a VME platform with a fully programmable general purpose FPGA board. It has inputs for the requested beam modes (intensity limits) defined by the logic layer of the MPS, the operational modes (which defines the beam paths in the machine), the MPS permit signal, the 3 MHz machine timing, and a macro-pulse trigger. It also has ability to control a mechanical shutter used to block laser light from reaching the cathode when it's necessary.

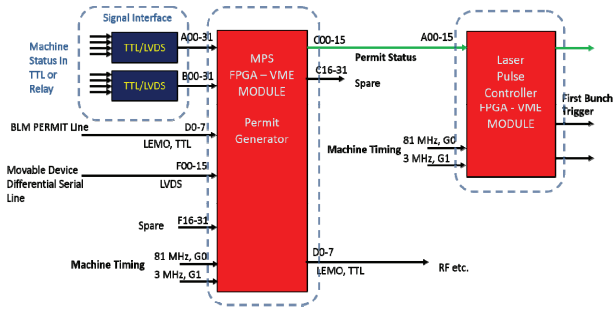


Figure 3: MPS FPGA block diagram.

MPS MAIN PERMIT SYSTEM

The main MPS permit generator board is the central component of the system that serves to collect status (OK/Not-OK) information from the various machine subsystems. The information is used in conjunction with user input such as operational mode and beam mode requests to generate a permit condition. The subsystems interface with the system through several modules that are designed to maintain signal integrity and provide noise immunity by converting input signals to Low Voltage differential signal levels (LVDS).

The permit system is built on a general purpose FPGA board designed to take advantage of its expandability and customizability. The boards can be programmed "on the fly" via VME without any external hardware tools and without disconnecting the boards from the system, and without resetting the system or turning off the crate. A flash memory on the boards stores the programming file.

The boards have been organized into functional blocks to accommodate various input signal types. Machine status signals in slow TTL or relay contact logic levels from vacuum PLCs etc. are converted into LVDS format and sent to ports "A" and "B". Movable device status is encoded onto a serial differential link and fed into half of port "F" and beam loss monitor status, which is critical for fast protection is fed into "D" port via a 30 MHZ carrier pulse train. Output signals to the LPC are received via port "C" and several other TTL output signals are provided via the port "D" for RF inhibits and test monitoring via oscilloscope. In addition to this, several ports allow for future expansion. Individual subsystem and channels can be masked directly at the hardware level or via configuration control software based on modes.

BEAM LOSS MONITOR SYSTEM

The beam loss monitors (BLMs) are the primary protection devices against fast losses and are designed to protect the machine by inhibiting the beam or limiting the beam intensity in response to losses above damage threshold levels. In addition, the same loss monitor signals are also used for beam tuning and diagnostics. The monitors are made of plastic scintillator material coupled to photo-multiplier tubes (PMTs). They deliver a measurement of beam and dark current losses to the control system as well as generate a fast alarm signal when the beam losses exceed user-defined thresholds. The time resolution of the loss measurement provides the ability to distinguish single bunches within each macro pulse. This is achieved at the sampling frequency of 3 MHz (the bunching frequency of the machine) with a repetition rate of 5 Hz. Including cable delays; the interruption signals from the system are generated within 1μs after the loss has occurred during commissioning to the 50 MeV beam dump. Since dark current and beam can only be produced during the RF pulse of the gun and the acceleration modules, it is sufficient to monitor a time window slightly longer than 1 ms per macro-pulse interval; however for alarm generation, continuous monitoring was done to minimize the dependency on time.

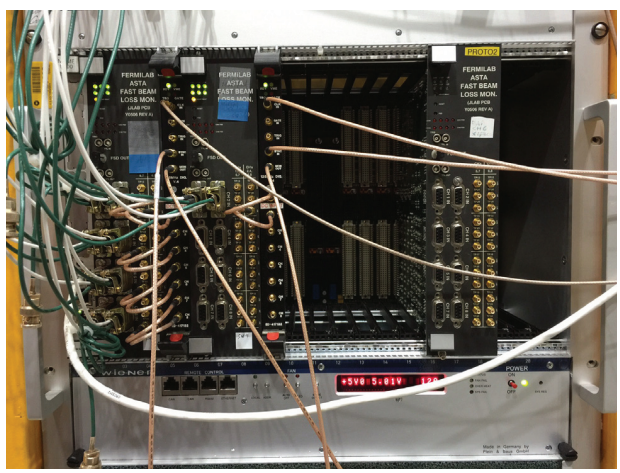


Figure 4: BLM Boards.

The front-end signal processing boards used are based on electronics designed by Jefferson Lab for their 12 GeV loss monitor system upgrade [4]. These BLM boards have been modified to meet the specific machine requirement at FAST. The main design change was to process the amplified signals from these boards using faster 125 MHz digitizer boards. The signals are further processed by an on-board FPGA to provide the require threshold levels and protection system shut-down signals.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

USER INTERFACE

The user interface is a critical component of the MPS. It provides operators with the tools necessary to find the source of detected trips and provides the following functionality:

- Trip visualization
- Configuration and control
- Post-mortem analysis

A web page, as shown in Figure 5, serves as the entry point to the user interface. This page provides a global view of the permit system, a list of all pertinent applications and has a global log of all critical MPS messages. Trip visualization is provided through several Synoptic displays (Figure 6) which allow users to view the overall trip status and then to drill down to a specific trip.

Configuration control is provided through several Java applications. FAST Facility operators use these applications to select desired beam and operational modes, setup subsystem status masks and set BLM limits. Post-mortem analysis is essential to recovering from a trip. Detected trips are cached on the main MPS board. A background process (Finite State Machine) timestamps all trips and saves them into a repository. A Java application is used to display the logged trip history.

Commissioning was accomplished by manually setting the subsystem status masks and BLM threshold limits. In further runs, as operational knowledge is obtained, the goal is to create a mapping between the modes, masks and BLM limits so that mode changes would allow the correct masks and BLM limits to be automatically downloaded to the permit system.

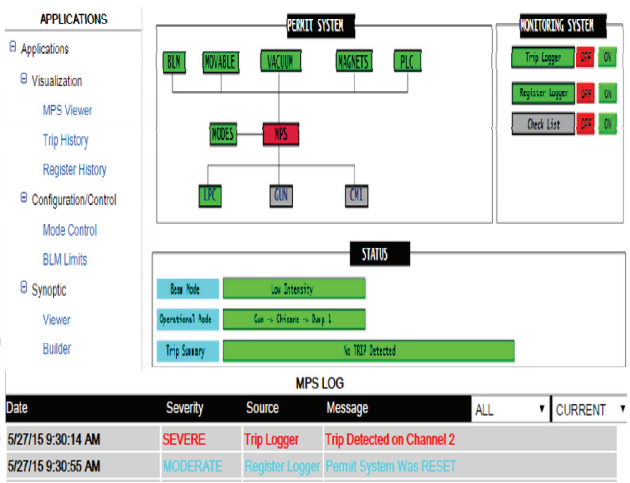


Figure 5: MPS Web Portal.

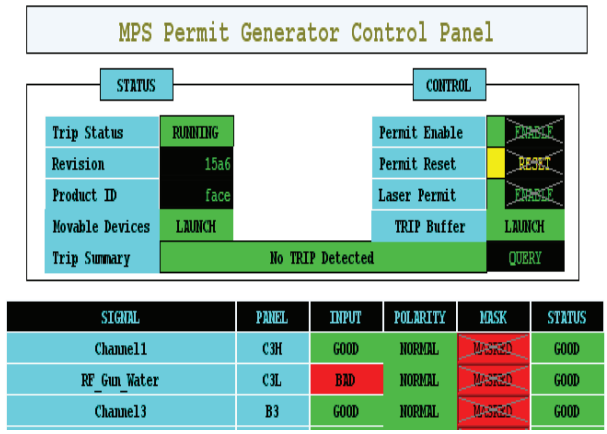


Figure 6: Configuration/Control.

CONCLUSION

The main components for the FAST Machine Protection System have been successfully commissioned. Commissioning began with successfully delivering 20 MeV beam to the low energy 50MeV dump for this first stage. The reaction time of the system is less than 1 us. The loss monitor system is sensitive to losses < 10 pC/bunch. The LPC meets the jitter specifications for tuning and controlling the number of bunches and bunch spacing. The main permit board was successfully interfaced to the control system. Trip visualization, configuration control and post-mortem analysis applications have been successfully developed. For the next phase of commissioning, the system will be expanded to protect the machine from 150MeV injection into IOTA.

REFERENCES

[1] M. Church, *et al.* "Design of the Advanced Superconducting Test Accelerator", Beams-doc-4212, September 2012

[2] M. Church, *et al.* "Status and plans for an SRF test facility at Fermilab", SRF'11, Chicago, August 2011, MOPO006; http://www.JACoW.org.

[3] A. Warner and Linden Carmichael, "Development of a Machine Protection System for Fermilab's ASTA facility", ICALEPCS'13, San Francisco, October 2013,MOPPC071; ; http://www.JACoW.org.

[4] J. Yan, K. Mahoney "New Beam Loss Monitor for 12 GeV Upgrade", ICALEPCS'09, Kobe, Japan, October 2009,WEP092; http://www.JACoW.org

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SAFETY INTERLOCK SYSTEM FOR A PROTON LINAC ACCELERATOR

Y. Zhao, Q.Ye, Y.Y.Du, J.He, F.Liu

Division of Accelerators, Institute of High Energy Physics, Beijing 100049, China

Abstract

The C-ADS Inject-I test facility is under construction in IHEP. An interlock system based on redundancy PLC is developed for machine protection and personnel safety. Device status, radiation dose, temperature of cavities and chambers are collected for machine state judge and interlock . A MPS(Machine Protection System) are working together with the interlock system in the control loop, and protect the machine in Four levels for different situations.

INTRODUCTION

The C-ADS linac include two Injectors and a main driver linac. Each injector is designed as a spare of another. The injector I proton linac in IHEP is consist of an ECR ion source, a LEBT (low energy beam transport line), a RFQ (radio frequency quadrupole accelerator), a MEBT(medium energy beam transport line) and superconducting spoke cavities[1]. The interlock system is designed and set up to protect personnel and devices from radiation hazards. It also will be a backup protection of MPS(fast Machine Protection System). The interlock system has permission signal, terminate beam signal and state signals collected from front device. Permission signals have the highest priority to access the Personal protection system, a key to the tunnel door will work together with the permission signals for safety. The state signals from Ion source, Power supply, LLRF (Low Level Radio Frequency) control system of RFQ, Vacuum and so on, indicate the corresponding system state or control "on and off". The terminate beam signals will shut the Ion source down when emergency.

HARDWARE DESIGN

Phoenix RFC (Remote field controller) -460R has been chosen as the main controller for the interlock system. Two RFC can automatically build a high-performance connection to implement redundant through fiber optics for synchronization. Two Profinet interface in the RFC connected the controller into the Profinet ring with RSTP(Rapid Spanning Tree Protocol) to achieve fast ring detection, One LAN interface in the RFC is connected to the control network for. Before each cycle, One RFC which is set as the PRIMARY redundancy role will control the process，and transmits the data to another RFC which is set in Hot Standby mode as BACKUP redundancy role. If one Controller fails, the other will take over immediately [2].

The Axioline modules include bus coupler, I/O modules. The I/O modules complete the function signal in/output are connected to the bus couplers links to the Profinet ring, it's the most fast I/O system in the world by now[2],the update time for each I/O modules is less than 1μs. The bus couplers has two Profinet interface which support PRL(The Phoenix Redundancy Layer), it will help to adopts valid values from I/O modules to the available Profinet controller.

The power supply of controller and Axioline modules are independent to ensure the reliability. Trio-diode modules are fixed for each two power supply units of the same type connected in parallel on the output side for redundancy. Those two power supply units will be isolated from one another. Figure 1 shows the redundancy diagram and Table 1 shows the signals from devices.

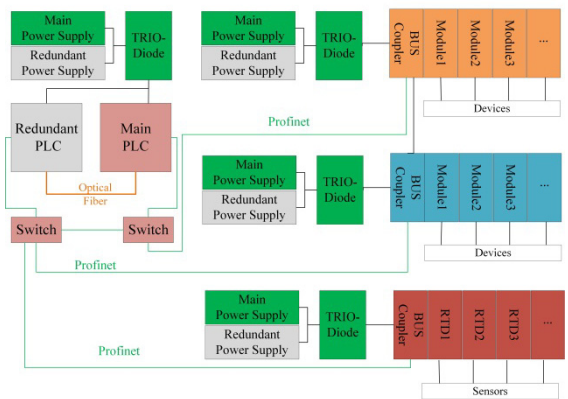


Figure 1: The redundancy diagram of interlock system.

Table 1: The Signals From Devices

Device	Input	Output
Ion Source	1	1
Vacuum	8	1
Personnel Protection	4	3
Power Supply	1	1
RFQ LLRF	1	1
Cryogenic	1	0
Super conductor LLRF	2	2
Bunching LLRF	1	1
MPS	1	6
Temperature	130	4
Flow meter	1	1

The fault tolerance requirements of C-ADS injector I are quite critical. So the reliability of the hardware is also considered. The index R (reliability), MTTF (mean time

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

to failures) and λ(the failure rate) are always used to measure the reliability of the control system . The MTTF and λ of each component at 25°C are list in the table in Table 2[2]. As the equipment and cables are located in the stable Power Supply room, the extremely environmental interference can be eliminated.

Table 2: The Reliabilities of the Hardware

No.	Modules	MTTF/h	$\lambda/(10^{-7}h^{-1})$
1	Power Supply/24VDC	500 000	20.00
2	Trio-diode	10 000 000	1.00
3	RFC 460R	281 627	35.5
4	Bus Coupler	3 950 853	2.50
5	Axioline DI	310 104	32.20
6	Axioline DO	310 104	32.20
7	Axioline RTD	310 104	32.20
8	Switch	2 891 123	3.46

PROGRAM DESIGN

The main control system run in the RFC is designed in PC Works 6.20.327(Phoenix contact) and can be download via Profinet. An EPICS IOC has been developed based on the EPICS driver for RFC 460R to complete the control and data-transfer. The driver is embedded in the main file，the so called "send/receive" protocol is adopted to exchange data block between the PLC and the EPICS IOC[3]. User-defined length data can be transferred during each program circle. The data type or other properties can be defined in the IOC. The IOC has been successfully used online. All operation is completed in the console of control room. The control architecture is shown in Figure 2.

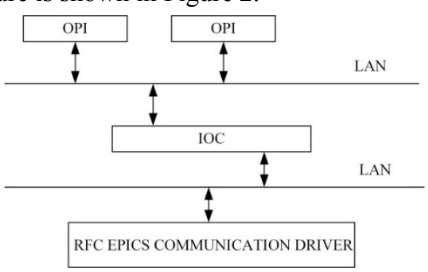


Figure 2: Interlock system control architecture.

PROTECTION LOGICAL DESIGN

The interlock system includes four states: stop, running, emergent stop and debug mode.

Stop mode is the safest mode for machine and personnel, the ion source and power source are disabled when the machine runs at stop mode.

Running mode can only be manual switched to when all the device state and temperatures are normal.

Emergent stop include manual emergent stop and interlock emergent stop. Manual emergent stop is the --- protection level, can be switched on the console panel in the central control room. Interlock emergent mode is automatically switched by PLC programme.

Debug is a special mode, because the whole machine is under construction and phase operation, the RFQ, bunching or the spoke cavities system need non-interlock state. The details of logical control is shown in Figure 3.

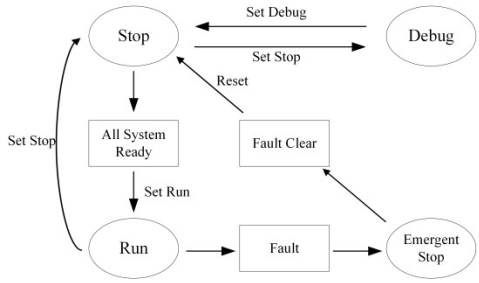


Figure 3: The control logical state.

There are 88 temperature sensors for RFQ water cooling, 12 for bunching cavities, 10 for beam dump, and 20 for vacuum chamber. As the RFQ resonance frequency is nearly linear with the temperature of the cooling water [4], the temperatures are precise to 0.01°C. And if any of the RFQ vanes or the walls temperature is above 28°C, it will trig interlock to stop the power or the beam. The bunching cavities' upper limit is 40°C, the dump and vacuum's upper limit is 60°C. The total state of flow meters for RFQ cooling water is also connected to work together with the temperature sensors.

At present stage, interlock provides 6 protection levels with MPS, for all LLRF state signals are connected to the MPS. According to the different unmoral situations, Interlock will terminate beam or let MPS stop the power source.

The first level - when the RFQ cavities wall, beam dump, vacuum chamber's temperature are higher than the upper limit, the interlock will disable ions source.

The second level – when the RFQ cooling water temperatures are high or the flow meters signal is fault, the interlock will disable the ion source, and let the MPS disable the power source of RFQ.

The third level – when the bunching cavities temperatures are high or the flow meters signal is fault. The interlock will disable the ion source, and let the MPS disable the power source of bunching.

The forth level – when the cryogenic system is not ready, the interlock will disable the ion source, and let the MPS shut the superconductor's power supply and the super conductor cavities' power source.

The fifth level – when the personnel protection system is on alarm, interlock will disable the ion source, and let the MPS shut all power sources.

The sixth level – is used as interlock manual emergent stop. It's the most critical situation, and will shut all the devices once be switched to.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SUMMARY

The interlock system based on the redundant design, combined with the EPICS control platform is more reliable to protect the machine. The different interlock levels ensure the effective protection and avoid unnecessary loss or operation. As the Injector I start to commission, there will be more various complex conditions, both hardware and software will upgrade to meet the actual demand.

REFERENCES

[1] Tang Jingyu et al. Conceptual Physics Design on the C-ADS Accelerator[R]. IHEP-CADS-Report.2011.
[2] http://www.phoenixcontact.com.cn/products.
[3] Liu G, et al. Epics driver for phoenix contact redundant PLC[C]//Proc of IPAC2013.3176-3178
[4] Xin Wenqu et al., “Resonance Control Cooling System for 973 RFQ at IHEP”, Nuclear Physics Review, 2013.6

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REALIZATION OF A CONCEPT FOR SCHEDULING PARALLEL BEAMS IN THE SETTINGS MANAGEMENT SYSTEM FOR FAIR

H. Hüther, J. Fitzek, R. Müller, A. Schaller, GSI, Darmstadt, Germany

Abstract

Approaching the commissioning of CRYRING, the first accelerator to be operated using the new control system for FAIR (Facility for Antiproton and Ion Research), the new settings management system will also be deployed in a production environment for the first time.

A major development effort is ongoing to realize requirements necessary to support accelerator operations at FAIR. The focus is on the pattern concept which allows controlling the whole facility with its different parallel beams in an integrative way. Being able to utilize central parts of the new control system already at CRYRING, before the first FAIR accelerators are commissioned, facilitates an early proof of concept and testing possibilities.

Concurrently, refactorings and enhancements of the commonly used LSA (LHC Software Architecture) framework take place. At CERN, the interface to devices has been redesigned to enhance maintainability and diagnostics capabilities. At GSI, support for polynomials as a native datatype has been implemented, which will be used to represent accelerator settings as well as calibration curves.

Besides functional improvements, quality assurance measures are being taken to increase code quality in prospect of productive use.

COMMISSIONING OF CRYRING AT GSI

At the time of writing, the CRYRING heavy-ion storage ring, a Swedish in-kind contribution to the FAIR project, has been set up at GSI, with only few additional components still to be installed. The machine features an electron cooler, an RFQ linear accelerator and two injectors for different types of ions. See Fig. 1 for an overview of the CRYRING ring section.

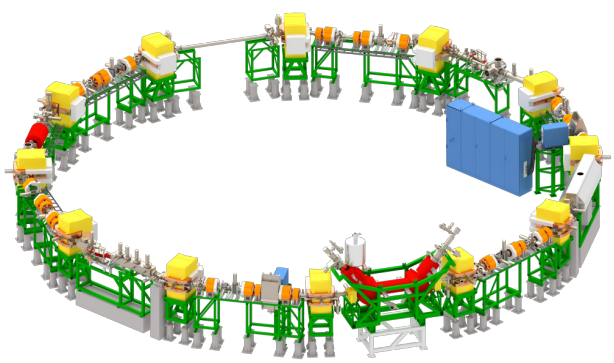


Figure 1: Overview of the CRYRING storage ring as setup at GSI (injection lines not shown), W. Geithner, GSI.

From a controls perspective, its commissioning represents a major milestone, as CRYRING will be the first machine solely operated via the new FAIR control system. Currently, the commissioning group still relies heavily on low-level control, but by the end of the year, they will have shifted their primary work place from the accelerator tunnel to the control room, utilizing the whole control system stack through the high-level applications provided.

TEST-BED FOR THE NEW FAIR CONTROL SYSTEM

Since CRYRING is equipped with its own injector line, it can continue operation even at times when the existing GSI accelerator chain is shut down for necessary FAIR upgrade and civil construction work, making it an ideal test-bed for the new control system. This way, it will contribute to validating concepts and technologies under real-world conditions [1], ensuring that the control system components work properly, individually and as a whole, and that business processes within and between involved parties are effective.

Although operating CRYRING does not imply the same requirements on the settings management system as the future FAIR facility will, core concepts necessary for highly flexible future operation scenarios can nevertheless be tested. As such, the beam-oriented approach to scheduling, designed for parallel beam operation at FAIR, will be utilized at CRYRING for the first time.

PARALLEL BEAM SCHEDULING CONCEPTS FOR FAIR

The designated operation modes of FAIR put demanding requirements on the new control system currently in development. To optimize the number of concurrent research programs, the facility will provide up to five beams in parallel with pulse-to-pulse switching between different particle types. Additionally, great flexibility shall be provided, allowing to change the parallel operation schemes on a daily basis.

Beam production chains and patterns are the central technical concepts within the new LSA-based settings management system to fulfill these requirements. Representing a major change in perspective, beam production chains establish a beam-oriented view on the facility, as compared to the accelerator-oriented view towards settings management dominant at GSI up to this point.

Beam production chains are defined using beam processes as atomic building blocks. Beam processes represent a specific procedure on the beam within one accelerator (e.g. injection, ramp, extraction). Within a beam production chain, the order of all beam processes necessary to provide the settings for producing a certain beam is described, from its

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

source up to its target, spanning all machines and transfer lines involved.

To be able to coordinate multiple beams traversing the facility in parallel, beam production chains are grouped into patterns. An example of typical parallel operation for the modularized start version of FAIR is given in Fig. 2.

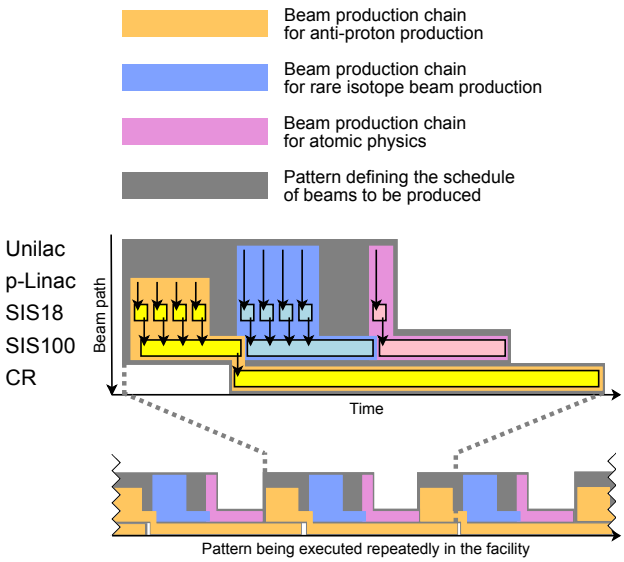


Figure 2: Example for parallel beam operation showing scheduling of beam production chains into patterns. HESR accumulating the anti-protons is omitted.

STATUS OF THE FAIR SETTINGS MANAGEMENT SYSTEM

LSA at GSI has made significant progress since it was first deployed as a test instance in 2008 [2]. In close cooperation with CERN, a structural redesign aimed at modularization led to more efficient handling of change management and release processes while also fostering extensibility [3]. First successful tests with beam at the heavy-ion synchrotron SIS18 were performed in 2010 [4]. More advanced machine experiments were made possible in the subsequent years through implementation of GSI-specific features, like flexible beam process lengths and optics definitions which are relative regarding time [5].

Development is currently focused on CRYRING and implementing the pattern concept. Although still at a prototype state and major features for controlling the entire future FAIR facility to be implemented, LSA now supports all requirements initially needed for CRYRING commissioning. Machine physicists were able to calculate reference settings using the pattern scheduling mechanisms and interface tests using mock-up front-end device controllers are being conducted. Testing data supply with actual devices is expected to start in November.

Two of the most recent enhancements leading to this achievement shall be described in more detail here.

RECENT ENHANCEMENTS OF THE FRAMEWORK

Polynomial Data Types

Developers at GSI implemented a polynomial data type directly in the central value package of the LSA framework. Before polynomials were available, all functions data had to be provided as discrete functions.

There are a lot of benefits to using polynomial data types instead of discrete functions. Polynomials produce continuous values rather than x-y-pairs for potentially millions of steps. Hence less data has to be stored because polynomials are represented as an array of its coefficients only. All polynomials can have an interval describing for which range of x-values they are defined, so it is possible to build sequences of polynomials over different intervals to easily represent complex functions. Consequently, it is possible to provide a more accurate representation of a function than before.

Combining a polynomial with an interval is inevitable for settings management, but it also brings up the problem that it can be interpreted in two different ways: A bounded polynomial can be treated absolute or relative to its interval. Using the absolute interpretation, the polynomial $p[n]$ is evaluated as $p[n].interpolate(x)$ independent of the boundaries, whereas on the relative interpretation the polynomial $p[n]$ is evaluated as $p[n].interpolate(x - b)$, where b is the lower bound. Therefore all polynomials are treated as relative in the current implementation.

The magnet group at GSI provides calibration curves for the magnets in the central component database also as polynomials. Before the polynomials data type was introduced, these polynomials had to be rasterized into discrete functions before they could be used in LSA. Now, calibration curves can be imported in their native, unmodified format as received from the magnet group and handled as polynomials internally as well. The same applies to function settings calculated by LSA, which are sent to the hardware, e.g. the function generator for ramped devices [6], as polynomials.

Data Supply

The LSA subsystem responsible for data supply to devices received a major overhaul in 2014. The project was carried out as a joint effort between CERN and GSI, with one developer from Darmstadt staying in Geneva for several months. The core motivation, equally important for both parties, was to improve diagnostics capabilities of the system.

While data supply results were formerly presented in a flat data structure for the whole supply process, the new hierarchic result classes allow operators to trace errors down to individual parameters, e.g. in case a set value is rejected by a front-end device controller. Navigating the hierarchy level of data supply results is possible in a very convenient way, starting at the beam process that was sent, via device adapters handling value conversion, down to set calls to the middleware, showing exactly where the error occurred. These levels of aggregation can also be identified in the corresponding class diagram shown in Fig. 3.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

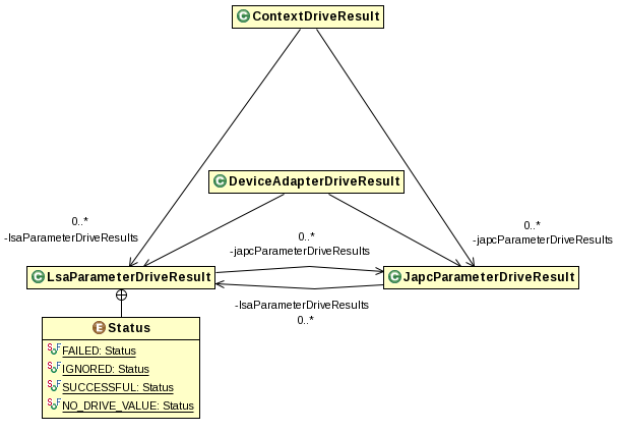


Figure 3: Class diagram showing the refactored LSA data supply results structure.

The enhanced diagnostics capabilities have already proved beneficial when supplying settings to large numbers of devices. As the need to handle error states can hardly be avoided during commissioning, they will also be a valuable tool for CRYRING.

Furthermore, the subsystem's code, which has been evolving for nearly a decade, received a major refactoring. Coding styles were unified, classes restructured and general maintainability and extensibility enhanced.

QUALITY ASSURANCE MEASURES

Both the implementation of the pattern concept as well as the aforementioned refactoring, being mission-critical core components of LSA, were carried out using a test-driven development approach to facilitate focus, correctness and test coverage. Beyond that, aiming to minimize potential errors within and between LSA components during commissioning of CRYRING, a variety of additional quality assurance measures have been implemented.

The machine model code, written by physicists, is being reviewed both technically by the LSA software development team and content-wise by other accelerator physics experts. During these sessions, to complement the traditional inspection approach, automatically generated reports utilizing a variety of static code analysis tools are consulted to support framework and machine model developers in writing correct and comprehensible code.

Besides methodology and code quality, the third pillar of quality assurance measures currently employed is integration testing. For CRYRING, there are roughly around 15 FESA (Front-End Software Architecture) classes to be run on front-end device controllers, either already implemented or currently under development. Each will be handling at least one, but usually several devices of the same type. To ensure that front-end device controller software works as expected from an LSA perspective, a test suite to perform automated integration testing has been set up. The tests typically operate on mock-up instances of FESA classes, which mimic the behavior of the actual hardware they are designed to control. At a later stage, tests may also be performed on front-end device controllers connected to real machine components. To maximize efficiency, the underlying testing concept distinguishes and reuses sets of generic tests for all device classes, device-type-specific tests and device-specific tests.

OUTLOOK

As previously mentioned, there are still major features to be implemented in LSA in order to fully support flexible parallel beam operation envisioned for the future FAIR facility. Taking SIS18 booster mode as an example, scheduling a certain beam production chain multiple times as a sub-chain will be necessary to consistently model the successive injections into SIS100. Another challenging task will be to support multiple patterns scheduled at the same time, with one main pattern being executed repeatedly and one or more other patterns serving experiments that require beam on demand only.

In 2016, tests are planned for transferring beams from the storage ring ESR, which is part of the existing GSI facility, to CRYRING. This endeavor requires synchronization between the previous control system still being used at ESR and the new control system used to operate CRYRING. Once LSA has been equipped with additional features specifically targeted at storage ring operation at GSI, most importantly beam manipulation during the cycle for experimentation phases, this will open interesting possibilities e.g. for working with rare isotopes at very low energies at CRYRING [7].

REFERENCES

[1] R. Bär et al., "News from the FAIR control system under development", PCaPAC'14, Karlsruhe, Germany, WPO004.

[2] R. Müller, J. Fitzek, D. Ondreka, "Evaluating the LHC Software Architecture for Data Supply and Setting Management within the FAIR Control System", ICALEPCS'09, Kobe, Japan, THP012.

[3] R. Müller et al., "Benefits, Drawbacks and Challenges during a collaborative Development of a Settings Management System for CERN and GSI", PCaPAC'14, Karlsruhe, Germany, TCO101.

[4] J. Fitzek, R. Müller, D. Ondreka, "Settings Management within the FAIR Control System based on the CERN LSA Framework", PCaPAC'10, Saskatoon, Saskatchewan, Canada, WEPL008.

[5] H. Hüther et al., "Progress and Challenges during the Development of the Settings Management System for FAIR", PCaPAC'14, Karlsruhe, Germany, WPO005.

[6] S. Rauch, M. Thieme, "Managing multiple Function Generators for FAIR", PCaPAC'14, Karlsruhe, Germany, FPO017.

[7] F. Herfurth et al. "The Low Energy Storage Ring CRYRING@ESR", COOL'13, Murren, Switzerland, THPM1HA01.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

NUCLOTRON AND NICA CONTROL SYSTEM DEVELOPMENT STATUS

Evgeny V. Gorbachev, Vasiliy Andreev, Alexander Kirichenko, Dmitrii Vladimirovich Monakhov, Sergey Romanov, Tatyana Vladimirovna Rukoyatkina, Georgy Sergeevich Sedykh, Valery Volkov, JINR, Dubna, Russia

Abstract

The Nuclotron is a 6 GeV/n superconducting proton synchrotron operating at JINR, Dubna since 1993. It will be the core of the future accelerating complex NICA which is under construction now. NICA will provide collider experiments with heavy ions at nucleon-nucleon centre-of-mass energies of 4-11 GeV. The TANGO based control system of the accelerating complex is under development now. This paper describes its structure, main features and present status.

INTRODUCTION

NICA complex will consists of heavy-ion and polarized particles sources, RFQ injector, heavy- and light-ion linear accelerators, superconducting booster synchrotron, Nuclotron and two superconducting collider rings [1].

The control system of the NICA complex aims at accomplishing few main tasks:

- Management of large amount of equipment which is distributed on the accelerator complex area.
- Realization of different regimes of the accelerator complex working cycle – colliding or fixed target experiments, various ion types and energy.
- Strict synchronization of accelerators in the chain.
- Comprehensive beam diagnostics during the entire cycle.
- Providing protection and safety measures.

We can specify few important features that the NICA control system has to provide:

- Centralized administration and monitoring of the control system components.
- Reliable operation, quick recovery after possible failures.
- Access to equipment has to be restricted for certain personnel with rights limitation according to user roles.
- Ease of support, modification and scaling during long accelerator complex life-time which will operate until 2045.
- Rapid development and easy deployment of the control system, taking limited time and man-power into account.
- Possibility to integrate third-party control systems as some components of the accelerator complex will be designed and constructed by external organizations.

CONTROL SYSTEM LAYOUT

The control system is distributed network of computers which communicate by means of transport protocol over TCP/IP. The NICA control system uses the TANGO controls [2] as the middleware providing such communication protocol.

TANGO is the modern distributed control system framework based on CORBA. The fundamental unit of TANGO is a device, which is an abstraction hiding real equipment or program component behind the standard interface. TANGO provides high level client application interface which has necessary programming classes to implement client-server communications - synchronously or asynchronously execute commands, read or write attributes, or use events to acquire the data from the TANGO devices. TANGO incorporates a number of tools to build efficient control system environment including centralized administration and monitoring, access control, logging system, data archiving and code generation for rapid development of the TANGO device servers using C++, Java and Python.

Three layers of the NICA control system components can be distinguished (Figure 1):

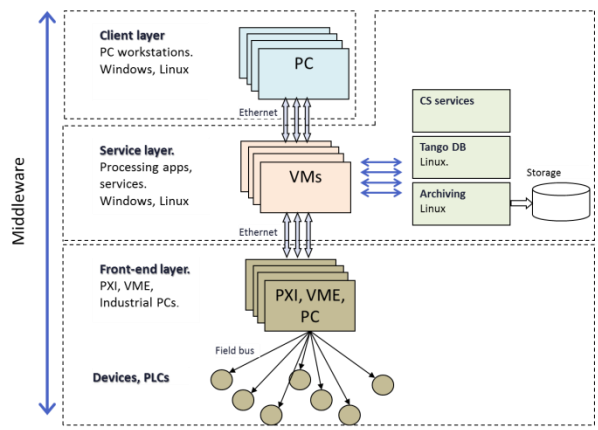


Figure 1: NICA control system layout.

1) Front-end layer consists of industrial computers, intellectual controllers, crates that directly control equipment and acquire data from sensors. Front-end computers run low-level TANGO programs that realize data acquisition, equipment handling and hide protocol and connection details from higher layer components.
2) Service layer consists of high level TANGO devices representing entire subsystems. They collect data from front-end TANGO devices, process it and realize algorithms to control some large subsystems. Those high-level programs provide standard TANGO interface for entire subsystems allowing client software to execute commands, read and write attributes without knowledge of subsystems structure. Besides, the service layer provides a set of services that are necessary for efficient control

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

system functioning – administration, control system hardware and software management, monitoring, data archiving, development services.

3) Client layer represents the accelerator complex state to operator, visualizes acquired data and allows operator to perform control tasks. We would like to provide global interface to allow operator to access the entire accelerator complex control with possibility to navigate to its components.

Front-end Layer

Primary requirements for the front-end layer hardware are easy and rapid development, good reliability and performance, easy maintenance during the accelerator complex operation. The development and support of custom hardware for the large accelerator complex is a very resource-consuming task. It is decided to use commercial hardware for most of the control systems tasks. The base of the NICA controls front-end layer is National Instruments PXI, which is modular eurocard packaging platform with PCI and PCI express busses with additional synchronization and trigger lines.

PXI platform offers a wide range of modular instruments of many types including digital and analog I/O, industrial interfaces, digitizers and scopes, signal generators and many others. National Instruments provides Windows and Linux drivers for most of the hardware and supports Labview as well as text based languages.

While most of the control system tasks can be implemented by modular instruments, some unique signal generation, processing and synchronization tasks can be realized using programmable logic devices. National Instruments provides few FPGA solutions to implement custom hardware. One of them is FlexRIO product family, which consists of PXI and PXI express modules with Xilinx FPGA, large amount of on-board memory and standard or custom I/O modules that are attached to the front panel of FlexRIO and have access to pins of FPGA. Developer can implement custom FPGA firmware as well as custom input and output signal processing, conditioning and conversion.

Another solution is CompactRIO controller, which is FPGA and processor based platform with standard or custom I/O modules. It is running real-time OS and controlled over Ethernet.

The TANGO device servers for a range of National Instruments hardware performing common acquisition tasks were developed. They include drivers for digitizers and scopes, analog and digital input and output, timers and counters, digital multimeters, temperature sensors acquisition modules.

Software for FPGA based boards consists of two parts: FPGA firmware and FPGA interface program performing data exchange between controller and FPGA. LabVIEW provides rapid firmware development using graphical language and VHDL. Interface programs can be developed as the TANGO devices using FPGA Interface C API. Those TANGO devices are running on PXI or CompactRIO controllers providing an easy way to integrate FPGA based solutions into the TANGO environment.

Developed drivers allow quick deployment of almost any National Instruments acquisition board. Each driver can perform a single input or output task so it is necessary to run few TANGO devices to realize all features of an acquisition board. The acquisition properties such as sample frequency, samples number, trigger source, and others are configured as the TANGO devices properties.

Client Layer

TANGO provides the set of graphic user interface toolkits for rapid development of graphical client applications in all supported programming languages – C++, Java and Python. There are also LabVIEW TANGO bindings that allow developing full featured TANGO client applications using rich Labview visualization features. Most of the existing desktop client applications for the NICA control system are developed in LabVIEW with the TANGO bindings.

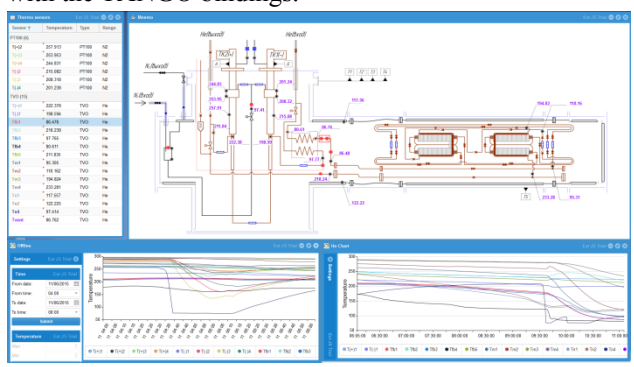


Figure 2: Thermometry web application for the NICA superconducting magnets test bench.

Few auxiliary TANGO devices were developed to simplify creation of web client applications by providing access to TANGO from web browser:

- The TANGO device server RestDS realizes REST protocol to access TANGO devices attributes and commands. It contains embedded http server providing access to TANGO devices attributes and commands at URL combined from host, port, device and attribute names. The reply is JSON string consisting of attribute name, value, timestamp and quality. POST method can be used to execute command with input arguments.
- The TANGO device server WebSocketDS also contains embedded http server and is intended for accessing attributes via WebSocket protocol. The attributes names and the device name are specified in properties, and the data update rate is configured via polling period.
- The TANGO device server TangoWebAuth performs authentication of users to restrict access of web applications to TANGO devices. The access rights are configured in a special database.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The developed auxiliary TANGO servers allow creation of web application in JavaScript without any intermediate application servers. Thermometry web application for the NICA superconducting magnets test bench is shown in Figure 2.

Service Layer

High level applications of the service layer do not interface with any hardware, so they can run in any place of the control system and can be virtualized. Virtualization is an effective way to deploy control system applications providing easier management of virtual machines (VM), better tasks isolation, fine tuning of CPU time or disk space allocation and more efficient usage of hosts resources. Virtualization also makes possible to obtain high availability features – any virtual machine can be quickly restored or migrated from failed physical server to run on a different host.

Proxmox VE is used as a virtualization solution. It is a free server platform for creation and management of virtual infrastructure including virtual machines, local and shared storage and high-availability clusters. It supports both Kernel based Virtual Machines (KVM) and Linux container-based virtualization which provides isolated Linux containers with no performance impact.

Shared storage is the vital part of virtualization to provide high availability of virtual machines. There are few requirements for the shared storage:

- Good performance as we need to run many images on it.
- Redundancy to allow using VM images even if some disks or servers failed.
- Scalability – storage must be expandable without losing performance to provide the data storage for future control system tasks and data acquisition.

The NICA control system uses distributed object storage cluster Ceph [3] as shared storage solution. Ceph storage cluster consists of nodes which communicate with each other via network and distribute and replicate data dynamically. Ceph's RADOS block device (RBD) is an ideal way to store VM images that are stripped and replicated across the entire storage cluster. Ceph RBD is thin provisioned so unused image space can be reclaimed.

It is possible to run Ceph services directly on Proxmox VE nodes, so the same hosts are used to provide both distributed storage and virtualization cluster. The cluster presently consists of four nodes each with dual eight core E5-2600 Intel Xeon processors, 32 GB of RAM, five reliable 15k RPM SAS2 disks and enterprise class solid-state drive (SSD) for operating system and Ceph journal devices. The network equipment includes two 1 Gbit network switches to separate public and Ceph cluster network. One of them will be upgraded to 10 Gbit network switch to provide network capacity suitable for Ceph cluster network.

The high availability of control system is achieved by hardware solutions which include using of uninterruptable power supplies (UPS), redundant power supplies for servers, enterprise class long-lifetime drives and data redundancy for virtual machines and databases.

All virtual machines run on Proxmox VE cluster with images stored on CEPH storage with replication factor of three. Hard drives failures are handled by Ceph and not affecting the operation of the control system at all. All virtual machines are periodically backed up to external Network File System (NFS) storage located on redundant array if disks (RAID). The uninterruptable power supplies are constantly monitored by using Network UPS Tools (NUT) [4] to ensure proper VMs shutdown on low charge.

The TANGO database is essential for the NICA control system operation. This is a MySQL database that runs in Linux container on fast local SSD for performance reasons. The high-availability of the TANGO database is achieved by means of MySQL replication. Semi-synchronous MySQL replication with two master MySQL servers with fully symmetrical configuration is deployed. Each master MySQL server is a slave of the other MySQL server but only one of them is getting external connections to avoid possible conflicts. A proxy solution for the TCP–based applications HAProxy [5] is used to periodically check health of MySQL servers and route external MySQL traffic to primary or backup MySQL backend server. HAProxy runs on high-available VM to ensure its permanent availability. Periodic dump of the TANGO database from MySQL slave server is used as additional measure to provide hourly TANGO database backups for two weeks of operation.

The storage space scalability is provided by Ceph which retains or improves its characteristics when adding more disks and more nodes.

Storage space and throughput, CPU cores number and amount of RAM can be increased by adding more nodes to the control system cluster. As a result, more data and VM images can be stored and more virtual machines can be run. The only drawback is a short burst of network traffic during data rebalancing to new storage daemons.

CONTROL SYSTEM ADMINISTRATION

The TANGO device servers are running on many distributed computers, hence the ability to control all the control system components remotely is necessary. TANGO provides a special service Starter which is running on all control system computers. Its main task is to start all necessary TANGO device servers in proper order at system startup and keep them running.

The TANGO manager Astor is used to visualize the state of entire control system and provide tools to start or stop the TANGO device servers remotely, access their logs and settings in the TANGO database. Operator can see if computers and TANGO device servers are running or not. However, a computer can have performance problems or the TANGO device can be in fault state which cannot be seen using Astor.

An additional TANGO monitoring system which consists of two parts was developed. First part is a

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

TANGO device to monitor computer resources. It runs on each computer of the control system and periodically collects information about the disk, processor and memory usage. Another part is a TANGO device intended for monitoring states of every TANGO device of certain subsystem. The monitoring client application (Figure 3) represents states and statuses of the all TANGO devices and computers of the control system. Alerts and notifications are sent to developers in case of problems.

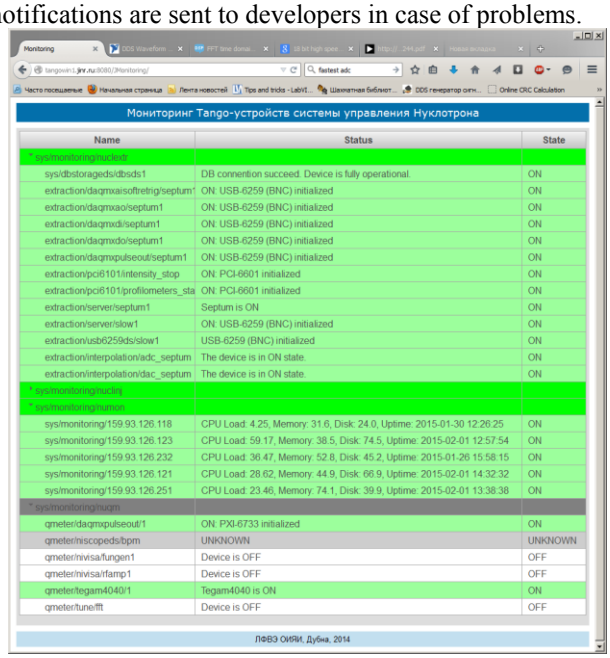


Figure 3: TANGO monitoring.

Apart from the Tango monitoring, monitoring solution Zabbix [6] is used to monitor network traffic, hypervisors resources, Proxmox VE cluster, UPS devices, TANGO database performance, Ceph cluster performance and health, local and shared storage space.

Another important aspect of control system operation is access control. We want to restrict access to control system software for certain computers and users and give them corresponding rights to perform specific tasks. Network access limitation, including private sub networks for subsystems and proper firewalls configuration is one of the possible measures. Another possibility is Tango AccessControl service that restricts connections to TANGO devices to certain users connecting from certain computers. The access checks are performed on the client side before opening connection to the TANGO device.

Additional server-side authorization service was developed to provide more secure access control and more flexible rights restrictions. It uses additional authorization TANGO device which accepts username/password pair from client application and authenticate the user using Linux Pluggable Authentication Modules (PAM) and information from MySQL database. If authentication is passed, a temporary session is opened for the specific user, IP address and process ID. TANGO device which needs to check the user's rights makes request to authorization server from command execution method with TANGO device name, command name, client IP and process ID. After that, authorization server checks the IP and process id pair against opened sessions and checks user rights from the database.

It is possible to precisely tune the access rights using MySQL regular expressions. Operator-expert rights separation can also be implemented.

CONTROL SYSTEM MANAGEMENT

The NICA complex consists of few large facilities, such as accelerators and transfer lines. Each of them has a number of subsystems. Every subsystem consists of large number of equipment, TANGO drivers, cables, computers, network equipment which are developed and maintained by certain people.

The NicaControls [7] database was developed to store this information and describe relations between its components. Using these relations one can find documentation for hardware or software, links to source code and executable files, developer name, rack location, computer IP address and other useful information.

Information in the database is added during the control system development. The NicaControls database is also used by other control system components. The TANGO monitoring system periodically queries NicaControls database to include or exclude the TANGO devices from monitoring. The server-side access control also gets access rights information from it.

CONCLUSIONS

Design of the NICA control system infrastructure was presented. It provides rapid development of hardware and software using the TANGO Controls and National Instruments equipment, reliable and scalable operation by using virtualization and Ceph storage cluster, comprehensive administration and monitoring, efficient management of the control system equipment and software by using the NicaControls database.

Several Nuclotron subsystems and prototypes for the future NICA accelerators were developed using the described approach.

REFERENCES

[1] G.Trubnikov, N.Agapov, O.Brovko at al., NICA project at JINR, Proceedings of IPAC2013, Shanghai, China, May, 2013
[2] TANGO Controls: http://www.tango-controls.org
[3] Ceph: http://www.ceph.com
[4] Network UPS Tools: http://www.networkupstools.org
[5] HAProxy: http://www.haproxy.org
[6] Zabbix: http://www.zabbix.com
[7] Gorbachev E.V., Sedykh G.S. The equipment database for the control system of the NICA accelerator complex, Proceedings of ICALEPCS2013. – San Francisco, CA, USA, 2013, Pp. 1111-1113

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

IMPROVING SOLEIL COMPUTING OPERATION WITH A SERVICE-ORIENTED APPROACH

A.Buteau, B. Gagey, G.Abeille, Synchrotron SOLEIL, Gif-sur-Yvette, France
JC Fouquet, JCF company, Paris, France

Abstract

SOLEIL Computing division is in charge of managing computing infrastructures and applications on a 24/7 basis for SOLEIL staff and beamlines users.

During the last years SOLEIL accelerators and beamlines have been facing an ever growing dependency on Information Technologies to be able to deliver their service to their respective users. During the same period the Computing division had to manage an increasing number of technologies and software applications while continuously improving IT operation performances.

All this happened in a "classical" scientific environment where having immediate scientific results seems always more urgent than working on enhancing operational activities and minimizing the workload of IT groups.

This context has been the key driver to start the project of optimising our IT operational practices by adopting the ITIL [1] methodology with the following objectives in mind:

- Enhance the quality of IT services delivered
- Decrease the time spent by IT teams in operational activities to be able to keep resources focused on projects and development

The present paper will describe the overall vision of the project "Improving SOLEIL IT operation with a service oriented approach" and the strategy to make the methodology adopted efficiently by all IT groups.

THE MOTIVATIONS TO CHANGE OUR I.T OPERATIONAL PRACTICES

SOLEIL Computing Division Organisation

SOLEIL Computing division is composed of about 40 people organized in 4 groups:

- The ICA group is in charge of software development for Controls and Data Acquisition
- The ECA group is in charge of electronics for controls and data acquisition
- The ISG group is in charge of databases administration and enterprise applications (such as financial ones, technical documents management, etc.)
- The ISI group is in charge of IT infrastructures : networks, servers, file storage , desktops

Operational Practices of IT Groups are Very Different

Depending on their role in SOLEIL organization these 4 groups may have different internal customers:

- ICA and ECA groups are very close to "business" activities as they are working only for Accelerators and beamlines
- ISG is closer to SOLEIL Administration division
- ISI group activities are focused on providing IT infrastructures for all SOLEIL activities from offices to control systems

Moreover these groups are using different operational tools (see Fig. 1) to follow-up their daily activities:

- ICA uses JIRA [2] to manage software changes, users requests and development projects
- ECA, ISI use a CMMS [3] (*Computerized Maintenance Management System*) to track operational activities on the hardware components they are in charge of.
- ISG uses Redmine [4] to follow-up their internal software developments
- The "business" operational groups (*accelerators operators and Experiment Hall coordinators*) also use an ELOG system to track the I.T incidents which occur on the synchrotron facility

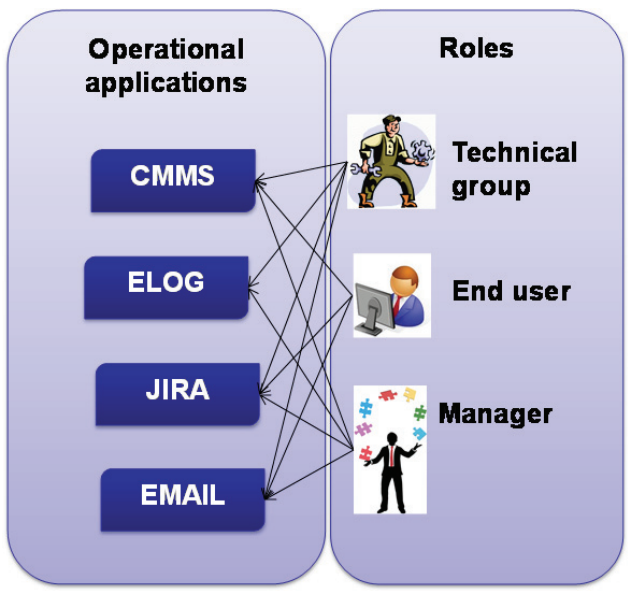


Figure 1: Operational tools used by various support groups.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Last but not least, at the SOLEIL organization level there is no clear definition of business impact of IT incidents/problems that could help all IT people to focus their efforts on the most urgent activities for business.

Be More Efficient and Enhance Services Delivered to End User

Having different operational practices for the 4 IT groups and the "business" operational groups had been possible during the last 10 years but raised daily difficulties that became unsustainable during the last years:

- For an end user the first difficulty is to know to which IT group he must ask for a given service. Then he/she has to choose the right tools (JIRA or CMMS) to post a request
- Then when this request may require actions from various IT groups (*which is more and more common because IT services are always more interconnected*) he/she has to "navigate" between different tools to follow-up this request
- On the other hand, I.T groups have no common vision of the priority/urgency of this request or incident.

This leads to a very chaotic and inefficient treatment chain of the requests/incidents both from the IT resources involved and from the end user perspective.

HOW ITIL CAN HELP US TO IMPROVE OUR IT PRACTICES

Some Reminders on ITIL

ITIL is a set of practices for IT service management that focuses on aligning IT services with the needs of business.

ITIL describes "*processes, procedures, tasks, and checklists which are not organization-specific, but can be applied by an organization for establishing integration with the organization's strategy, delivering value, and maintaining a minimum level of competency*".

ITIL is focused on operational processes rather than norms and is:

• modular: it can be used for very large but also small organizations

• is a quality oriented methodology allowing better definitions of: objectives, operational processes, responsibilities, and indicators.

• defines a common vocabulary between IT people and "business" people

Our Ambitions for the Beginning are very Modest: Focus our Efforts on a Limited Part of the ITIL Methodology

ITIL aims to define best practices for the whole "Service life cycle"

- ITIL Service Strategy: understands organizational objectives and customer needs
- ITIL Service Design: turns the service strategy into a plan for delivering the business objectives.
- ITIL Service Transition: develops and improves capabilities for introducing new services into supported environments.
- ITIL Service Operation: manages services in supported environments

Nevertheless in a first phase, we decided to concentrate our efforts only on "Service Operation" being conscious that changing operational practices of about 40 people and 350 users is something that needs time!

A Preliminary Study Convinced Us of the Interest of the Methodology

To check ITIL can help us in a pragmatic way we decided to analyze the ICA operational practices to check how ITIL could efficiently help us.

To this aim we examined how ICA group was currently working for each of the ITIL Service Operation and Transition processes.

This analysis document was discussed with various operational managers and inside the ICA group, the conclusion being that definitively following ITIL methodology will help us defining many axes to enhance our current practices.

A FEW INITIAL GUIDELINES

Convince and Not Impose

The decision of adopting ITIL has been left in a first time as an option to the managers of the various IT groups.

The ICA group having initiated the step forward was naturally designated as the pioneer group. Its feedback (*pros and cons*) with ITIL practices should then be useful to convince and help other groups to change their way of working in a second phase and to adopt gradually ITIL operational processes.

Promote the Kaizen Philosophy

The philosophy behind our ITIL project is to promote to the idea of a **continuous improvement** of the I.T services we are delivering. In fact only an active participation of operational people will allow to efficiently track IT defaults but also to gather from IT operational people solutions and enhancements proposals.

Keep Using the JIRA Tool to Manage Operational Requests

It was tempting to use a natively ITIL oriented tool (*such as ServiceNow [5]*) to be driven by the tool when putting in place the methodology. Nevertheless we thought it was too risky in our context to change at the same time the operational practices and

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

the CMMS/JIRA/ELOG tools SOLEIL operational people used for years.

We accepted the idea of having for a few months a hybrid situation having within JIRA some requests categorized according the ITIL semantics along with "old style" categorization tickets.

OUR FIRST EXPERIENCE OF ITIL: THE ICA GROUP

An Immediate Benefit: ITIL Semantics Helps Categorizing User's Requests

The first benefit we found what that thanks to its very precise semantics and definitions, ITIL helped us to better prioritize and categorize the continuous flow of JIRA requests (around 100 per week).

Nevertheless, we would like to insist that even if words seem easy to understand, it took a couple of months for a team of experienced software developers to integrate in their daily practices the subtle differences between "problems", "incidents" , "events", "service requests".

Second Benefit: Clarification of the Roles of the Various Operational Actors

ITIL definitions helped us to clarify our organizational practices and the "who does what" for the following activities and their operational interfaces with the ICA group.

- **Service Desk Management**: This 24/7 activity is shared between Machine operators and Experiment Hall coordinators who are in charge of the "level 1" support while ICA group is in charge of "level 2" . The role of the service desk has been clarified and ends up now when a workaround has been found to restore the failing service to an operational state.
- **Incident Management** : Incident managers have been designated within the ICA group to make a "root cause" analysis and create the related "problems" JIRA tickets
- **Problem Management**: Problems managers have been designated to follow-up the problem resolution by the development team.
- **Service Requests Management**: This activity is now systematically done by the member of the ICA team who is on call.

Service Transition Processes: ICA Group Proved to Be Quite Mature

Being a group mostly composed of experienced software developers the ICA practices for ITIL transition processes were without surprise well mastered. In particular:

- **Release Management** : Each of the functional service is delivered to users through a software package which regroups software components and is under the responsibility of an engineer who is in charge of :
  - managing the roadmap
  - integration of the related software artifacts in the various source code management systems and in SOLEIL Continuous Integration Systems [6]
  - Communication with users through Release Notes
  - Level 3 expertise on the software package
- **Software Assets Management:** all ICA deliveries are declared and followed in our Continuous Integration Systems. This guarantees that all Control systems software assets are identified, under version control and production versions are correctly identified.

FIRST EXPERIENCE OF ITIL WITH OPERATIONAL GROUPS

Better Manage Incidents: A Win-Win Deal for All Support Groups

As explained in the first paragraphs the various operational groups (IT and business) were using 4 different tools (not including physical contacts and telephones!) to manage daily incidents.

Focusing our efforts to enhance this operational process would be a win-win approach for all support groups.

We organized a workgroup which duty was to define a standard "Incident form" and the lifecycle of an incident.

The JIRA tool was then parametrized to manage numerically the form (see Fig. 2) and the associated workflows.

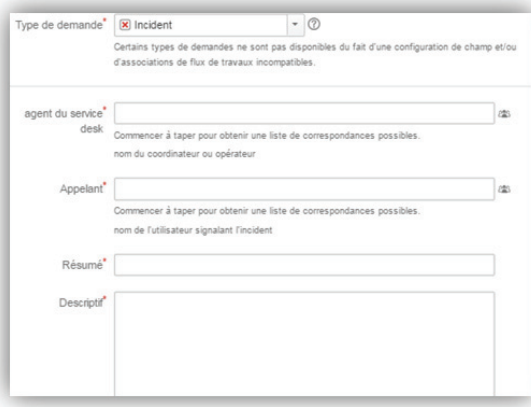


Figure 2: Incident declaration form.

A First Lesson Learned: "Business" People Must Also Be Trained

One of the difficulties we faced very quickly was the fact that if ICA group was trained to ITIL methodology, our

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

internal customers were not. This made the communications on first concepts such as "Problems" or "Incidents" much more difficult. We had then to make "Mini ITIL trainings" removing all mentions to IT. This helped to convince Accelerators and Beamlines people the operational processes we were dealing with were also applicable to their operational practices.

THE STEPS FOR THE NEXT 6 MONTHS

Complete and Publish the IT Service Catalog

ITIL promotes the publication to users of the "IT service catalog" which details the full list of services actually provided by Computing division and how to access them and to get help and contact information.

A preliminary version of this catalog (see Fig. 3) has been written by all IT managers. The first step will be in the next weeks to publish this first version to the groups in charge of Level 1 to help them in their daily activities.

Then in a second stage, it will be discussed with SOLEIL "business" representatives to clarify the commitment of all the parties. In particular this approach will oblige to define for each service who is the "Service Owner" on "business" side and which Service Level Agreement (*a.k.a SLA*) there are expecting.

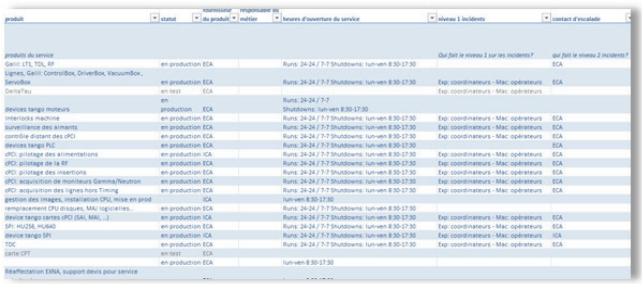


Figure 3: IT Services catalog.

A Service Oriented Portal for End Users

The JIRA form used for Incident Management will then be extended:

- to integrate the services identified in the service catalog to be able to directly assign the incident to the IT group in charge of it
- to other users requests such as "ITIL Service Request"

JIRA /CMMS Connection

Groups in charge of managing hardware components use the CMMS system to follow incidents and changes which occurred on each physical device. Keeping the CMMS tool as it is, was then mandatory for these groups.

On the other hand we want to have a unique entry point to IT services from the end user point of view.

We started a technical study on how to interconnect JIRA and our Tribofilm CMMS [7] using the available software interfaces of these tools.

- A first version will be deployed during Q4 2015 to deal with "Incident management" using mail interfaces of these tools.
- In a second phase (Q1 2016) we will integrate other processes such as "Changes Management "that will require development of Web Services on CMMS side and their integration within JIRA.

After these 2 phases the end users interface with IT support group will then be unique through the JIRA portal (see Fig. 4).

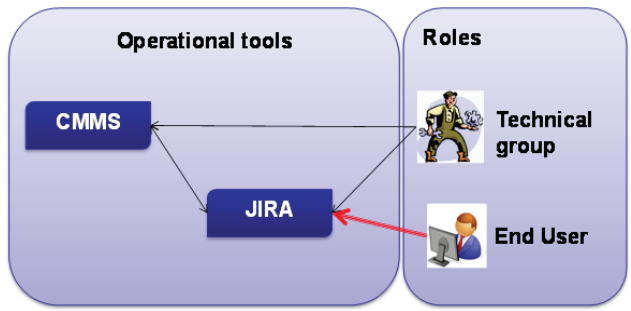


Figure 4: JIRA as the unique entry point.

Put in Place Formal Trainings on ITIL

Formal ITIL trainings will be organized within the next 3 months for all IT and "service desk" people taking into account the experiences gained through the ICA group example and the processes (such as "Incident Management") already in operation.

OUR VISION FOR THE NEXT 2 YEARS

Enhance Our "Change Management "Processes

A lot of work has to be done in close collaboration with "business" representatives regarding the change management process.

We expect this will be a challenging phase because it will require to clarify the "who decides what" on releases plans, projects priorities, etc.

For IT Division Extend ITIL to Other Processes Such As Service Transition or Service Strategy

The services being currently developed by the computing division such as the FlyScan project [8] are now taking into consideration parts of the recommendations of the "ITIL Service Design and Service Transition" processes.

The definition and follow-up of IT Services Strategy will be directly done under the Director General authority which proves the importance IT has now for all the SOLEIL organization.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONCLUSION

ITIL already helped us in improving the service that ICA group delivers to its IT end users. Nevertheless we are also very modest by foreseeing how long the road will be to extend it to all IT groups of SOLEIL. The strategy of a gradual adoption of ITIL and its various operational processes is the only one that can be followed considering the differences in the groups' maturity and cultures.

Another challenge will be in the next years to extend the best practices promoted by ITIL to all "service oriented" activities of SOLEIL.

In fact the quality of the service delivered to the end user scientist can be considered as the result of a chain of services as illustrated by the following schema (see Fig. 5) . If we consider that "***The strength of a chain is the strength of the weakest link***" enhancing only I.T services will not be sufficient to enhance the satisfaction of the end user.

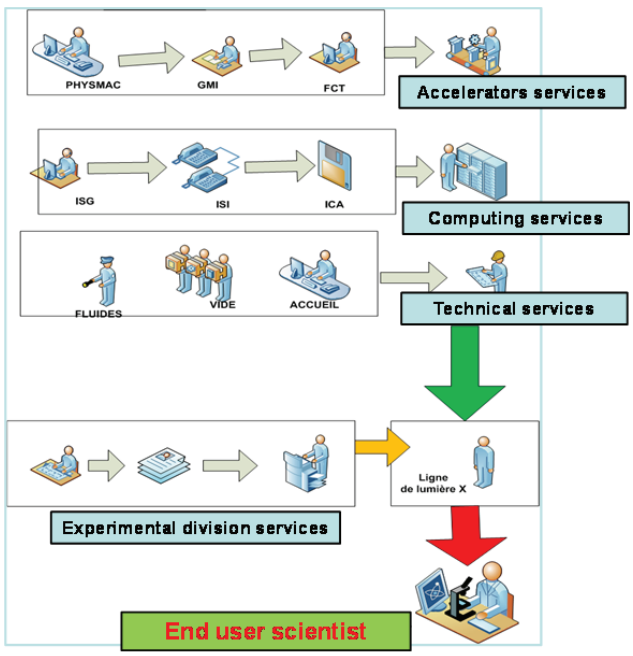


Figure 5: Chain of services to deliver a scientific service to the end user.

To this aim SOLEIL is changing its internal organization and ITIL best practices will be a guide in the next years for all SOLEIL accelerators and technical groups which activities are directly impacting the scientific service delivered to our end users.

REFERENCES

[1] ITIL methodology: http://en.wikipedia.org/wiki/ITIL
[2] JIRA : htpp:// www.atlassian.com/software/jira
[3] CMMS: Computerized maintenance management system: http://en.wikipedia.org/wiki/Computerized_maintenance_management_system
[4] Redmine: http://www.redmine.org
[5] ServiceNow : http://www.servicenow.com
[6] MOD3O02: "Continuous Delivery at SOLEIL" G. ABEILLE, Synchrotron SOLEIL ICALEPCS 2015
[7] TRIBOFILM CMMS product: http://www.tribofilm.fr
[8] WEPGF056: "FLYSCAN: a fast and multi-techniques data acquisition platform" N. LECLERCQ, Synchrotron SOLEIL ICALEPCS 2015

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

BEAM INSTRUMENTATION AND DATA ACQUISITION FOR CRYRING@ESR

T. Hoffmann, H. Bräuning, R. Haseitl, P. Miedzik, T. Milosic, R. Lonsing, A. Petit, A. Reiter
GSI, Darmstadt, Germany

Abstract

At FAIR the re-assembly of the well-known CRYRING accelerator, formerly hosted by Manne Siegbahn Laboratory (MSL) Stockholm, is currently in progress. This compact low energy heavy ion synchrotron and experimental storage ring will be a testing platform for all control system (CS) concepts decided on for FAIR. The CRYRING CS will be based on the system originally developed by CERN which combines the JAVA based application level LSA (LHC Software Architecture), the data acquisition level FESA (Front-End Software Architecture) and the White Rabbit based timing system. All parts have been enhanced with GSI specific functionality. In preparation for the commissioning of CRYRING later in 2015 all required beam instrumentation (BI) equipment including the software is now under development. We present the data acquisition (DAQ) concepts for the various instruments with emphasis on the seamless integration into the overall CS. For standard BI systems, such as digital video imaging, profile and intensity measurement, VME and IndustryPC based DAQ systems are used. For beam position monitoring a new hardware strategy which combines the microTCA and FMC (FPGA mezzanine card) form factors is under evaluation.

INTRODUCTION

The modernized CRYRING accelerator is presently being assembled in Experiment Cave B on the GSI Campus [1]. A schematic overview over site and accelerator design is given in Fig. 1. The accelerator consists of a 108 MHz 300 keV/u RFQ linac with a 50 kV MINIS ion source platform, the ESR injection transfer

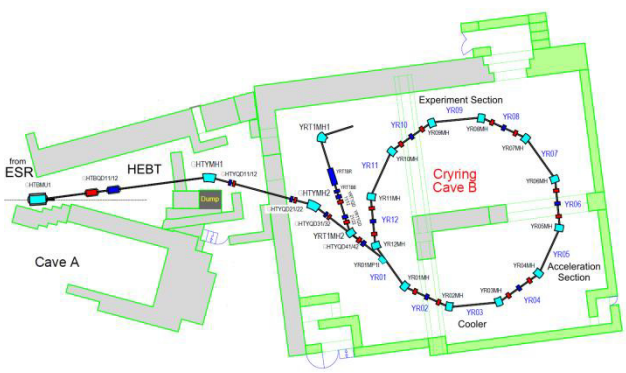


Figure 1: Scheme of CRYRING installation in Cave B.

line and a synchrotron storage ring of 54.18 m circumference including an electron cooler and experiment section. Different singly charged ions in bunched or coasting beam will be provided by the linac. Later, the GSI experimental storage ring (ESR) will inject cooled, highly-charged heavy ions via fast extraction which can be accelerated up to 14.8 MeV/u in case of U^{92+} or 96 MeV/u protons. All accelerator parts are equipped with original beam instrumentation systems designed at MSL as well as new FAIR type solutions. The original MSL detectors and specialised pre-amps were preserved, but all DAQ systems have to be replaced.

The main intention in operating this accelerator, besides the physics aspects, is to provide a test platform with ion beams for many FAIR relevant systems in both hard- and software. Commissioning of the ion source has already started. The RFQ is ready for operation. The linac and the ring will be commissioned next year.

CONTROL SYSTEM

The control- and data acquisition system for CRYRING operation is based on a three tier architecture with FESA as the front-end level, the ZeroMQ based middleware and the top JAVA application level supported by JAPC (Java API for parameter control) and the LSA settings management [2]. CRYRING will be operated and synchronized with White Rabbit (WR) timing [3]. All BI DAQ systems need to be equipped with dedicated WR timing receivers in stand-alone, VME and PCIe form-factors. All those receivers were developed at GSI.

BEAM INSTRUMENTATION

CRYRING is equipped with a variety of beam instrumentation systems. For all of them new DAQ software has to be provided, which is also suitable for further use at FAIR. Presently the following BI systems are under development:

Intensity

A beam intensity measurement, mainly in the linac part, was realized by using Faraday-cups. The signals are digitized in a VME system by a SIS3302, 8-channel, 100 MSa/s, 16 bit ADC. The current amplifier copes with AC and DC beam and the DAQ calculates the total charge plus mean and maximum currents.

For AC beams automatic determination of the signal's baseline for every bunch is supported. For DC operation, the baseline may be measured on user request when no beam is hitting the cup. In both cases predefined values may be used in case of problems with the online measurement.

The region-of-interest to determine the mean current and total charge can be taken from two dedicated WR machine events indicating “beam on” and “beam off” or via user-defined markers. For offline analysis the raw

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

data can be stored. Fig. 2 shows a chopped 500 µs D^+ beam pulse and the correlated intensity in units of Ampere at the linac injector.

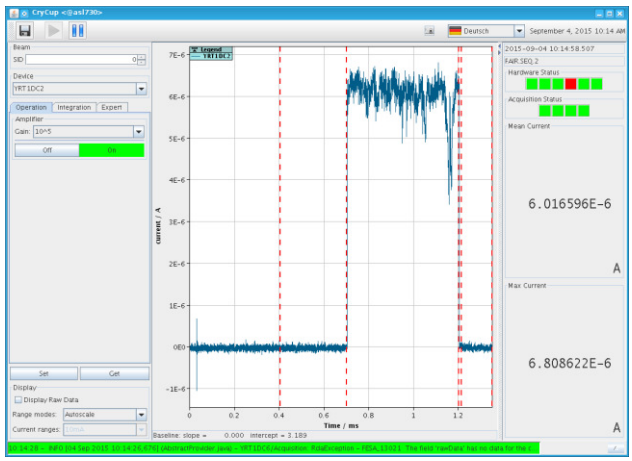


Figure 2: Intensity measurement of a chopped 500 µs D^+ beam pulse with a Faraday-cup.

For absolute beam intensity measurements in the ring both the Bergoz Integrating Current Transformer (ICT) and Bergoz Parametric Current Transformer (PCT) are used. For relative intensity measurements the IPM count rate is registered. As the bandwidth of the analogue outputs of those devices is below 1 MHz, the signals can be converted by a voltage-to-frequency converter (VFC) and sampled by the VME based GSI LASSIE [4] system, which is based on SIS3820 32 Bit / 32 Ch. multi scalers.

As CRYRING is bakeable to achieve an appropriate vacuum pressure, the transformers need to be protected against overheating. Hence temperature data derived from PT100 sensors is logged in LASSIE. The PT100 measurement is also required for temperature-drift compensation of the PCT. In the same setup Hall sensors are sampled to subtract magnetic field perturbations induced by dipoles and quadrupoles nearby. In addition, all other signals, which can be converted to frequencies such as Hall sensors, and BPM sum signals are fed into LASSIE. They are sampled and displayed on the same time basis for easy correlation.

Linac Energy

Phase and energy determination is realized with 3 capacitive ring pickups by time-of-flight measurements. The DAQ consists of a LeCroy Waverunner 6100A oscilloscope (1 GHz, 5 GSa/s) with 4 channels. As there are 5 signal sources (3 pickups, RF signal, debuncher signal), a Keithley switching matrix (Type 2701/7712 with 2x Dual 1x4, LXI) is used to select four signals for a given measurement. The oscilloscope is read out via LAN by a dedicated FESA class using the LXI protocol. The data is sampled over one thousand and more RF cycles. This allows for efficient noise reduction of the signal and oversampling, thus increasing the time resolution to approx. 5ps [5]. The time-of-flight between two pickups is determined by cross-correlation of the signals and yields with the known distance the beam energy.

Position and Orbit

CRYRING is equipped in total with 18 either horizontal or vertical oriented Beam Position Monitors (BPM), which will be controlled and read out by a high data bandwidth µTCA system. As µTCA, especially PICMG MTCA.4 [6], is an emerging form factor within the physics community, this DAQ system shall contribute in gaining experience on this platform as µTCA was defined besides VME and IPC as a standard form factor for FAIR beam instrumentation DAQ systems. The DAQ consists of a 12-slot MTCA.4 chassis hosting five AMC Dual FMC Carriers (AFC). Those are equipped with 250 MSa/s, 16 bit, 4 ch. ADC FMC boards. Both AFC and ADC were developed and produced by the Polish company Creotech under the Open Hardware License [7]. Furthermore the system contains a digital I/O FMC board as trigger input and remote control interface of the BPM preamplifiers.

After some incompatibilities and instabilities between different vendors of the main carrier hub (MCH), the module management controller (MMC) firmware of the AFCs is under revision. The setup will be prepared for usage in a Vadatech VT812 chassis with UTC002 MCH and also in a Schroff chassis with NAT MCH. The heart of the BPM DAQ is the trajectory measurement system (TMS), which was developed for measurements at CERN PS [8]. In a first step the system will operate as an oscilloscope type tracker at 125 MSa/s. Later on the system will be enhanced by full CS implementation with closed orbit, tune and bunch tracking features.

The Base Band Tune (Q) measurement system (BBQ) [9], which was invented at CERN and adapted for SIS18 at GSI, will be additionally installed at CRYRING to validate the TMS results. At CRYRING the DAQ is realized in VME also using the SIS3302. The pick-up used for the BBQ can be optionally used for Schottky analysis. This requires an LXI based network analyser (NWA) to investigate longitudinal and transversal beam parameter, such as tune, emittance and chromaticity.

Profile and Imaging

For the beam profile measurements the CUPID [10] system is used, which is already operational at the GSI

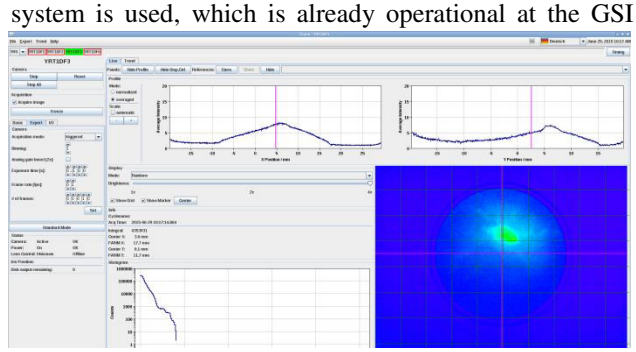


Figure 3: Profile of deuterium beam in the low energy beam transfer (LEBT) during injector commissioning.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

transfer lines and fully compliant with the FAIR standards. In total eight fluorescent phosphor screens are observed by GbE µEye CMOS cameras, which are concentrated on one 10 GbE HP switch and a 10 GbE uplink to an evaluating Kontron KISS front-end IndustryPC. An additional identical camera provides a direct view into the ion source. The typical presentation of profiles and live view is shown in Fig. 3.

Essential for beam cooling operation within the CRYRING are the non-destructive profile measurements using one horizontal and one vertical ionisation profile monitor (IPM) [11]. Ions produced by ionization of the residual gas are accelerated towards a micro-channel plate (MCP) detector with a position sensitive resistive anode. The time and position pulses of each ion hitting the MCP are amplified and shaped in a CAEN N586 spectroscopy amplifier, which are remote controlled via LAN. A peak-sensing multi-event CAEN ADC V785N digitizes the pulse heights. The interaction vertex of the ion can be reconstructed from the 4 signal amplitudes. The FESA class periodically reads the events stored in the ADC, calculates the position and performs the histogramming of the pulse height and position. It also sends periodically or on request the obtained spectra to interested clients. Insofar the handling of the IPM strongly deviates from the typical, machine event driven arm-trigger-readout cycle of other beam instrumentation like the faraday cups.

High Voltage and Fieldbus

High voltage power supplies (HV) are required for Faraday cups, IPMs, MCPs in front of scintillating screens and electrostatic devices at the RFQ injector beam line. Both commercially available modular HV power supply systems from CAEN and Wiener/ISEG will be used. Fesa classes for the CAEN SY5527 and the Wiener MPOD system have been developed. Special care has been taken to present a unified interface to clients regardless of the differences in the hardware. This allowed us to write a generic GUI for HV control which hides the underlying hardware from the user. However, the unified interface does not prevent future hardware specific extensions should they become necessary.

At CRYRING, the newly developed FAIR standard pneumatic actuators and some other slow-control devices will be operated by Siemens Programmable Logic Controllers (PLC). In order to keep the automation system modular a distributed and decentralized Profinet IO remote system is used. This allows safe hardware control in two different ways: decentralized from the CS and locally, while the machine is in maintenance or shutdown mode. The PLC communicates with the CS through a dedicated Siemens communication processor (CP) and by a FESA class, glued together with a CERN made SILECS (formerly called IEPLC [12]) interfacing software.

The system is accessible via the CS, one mobile and two stationary HMIs, an Android tablet and a PC based visualization running an additional webserver with the capability to archive signal, warning and error messages. For the purpose of wireless visualization, the Profinet infrastructure has been extended with two iWLAN Access Points operating at 5 GHz.

OUTLOOK

The installation of CRYRING is well on the way. The injector is already under commissioning and first beam instrumentation devices such as screens or Faraday-cups demonstrate very good performance within the new FESA DAQ concepts. It has shown that LXI based systems are a good choice. The LXI protocol can be used from simple function generator and lab oscilloscope readout up to high performance instrumentation such as Schottky NWA readout and linac energy measurements. Learning LXI paves the way to the full digital control room providing reliable remote control and data readout.

The ring itself is being assembled right now and installations are planned to be finalized in 2016. Next steps are setup of the PROFINET field bus infrastructure and improvement of the BPM system.

REFERENCES

[1] F. Herfurth et al., “The Low Energy Storage Ring CRYRING@ESR”, COOL 2013, Mürren, Switzerland.
[2] R. Huhmann et al., “The FAIR control system – System architecture and first implementations”, ICALEPCS 2013, San Francisco, CA, USA.
[3] M. Kreider et al., “Launching the FAIR timing system with CRYRING”, PCAPAC 2014, Karlsruhe, Germany.
[4] T. Hoffmann et al., “LASSIE: The Large Analogue Signal and Scaling Information Environment for FAIR”, ICALEPCS 2011, Grenoble, France.
[5] A. Reiter et al., “Improved Signal Treatment for Capacitive Linac Pickups”, DIPAC 2011, Hamburg, Germany.
[6] https://www.picmg.org/openstandards/microtca/
[7] http://www.ohwr.org/projects/afc
[8] G. Kasprowicz, “Determination of Beam Intensity and Position in a Particle Accelerator”, PhD Thesis: http://cds.cern.ch/record/1269328/files/CERN-THESIS-2010-081.pdf
[9] R. Singh et al., “Interpretation of Transverse Tune Spectra in a Heavy-Ion Synchrotron at High Energies”, Phys. Rev. ST Accelerators and Beams 16, 034201 (2013)
[10] B. Walasek et al., “CUPID: New System for Scintillating Screen Based Diagnostics “, IBIC 2014, Monterey, CA, USA.
[11] A. Källberg et al., “Progress Report for the CRYRING Facility”, EPAC 1998, Stockholm, Sweden.
[12] F. Locci et al., “IEPLC Framework, Automated Communication in a Heterogeneous Control System Environment”, ICALEPCS 2013, San Francisco, CA, USA.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CURRENT STATUS AND PERSPECTIVES OF THE CRYOGENIC CONTROL SYSTEM OF EAST

Liangbing Hu, Ming Zhuang, Zhiwei Zhou, ASIPP, Hefei, China

Abstract

EAST (Experimental Advanced Superconducting Tokamak) is the first full superconducting experimental Tokamak fusion device in the world which has been carried out ten campaigns since its implementation at the end of 2005. The cryogenic control system for EAST was designed based on DeltaV DCS of Emerson Corporation which has been in operation for the same time period and has been proved to be safe and stable. However, Manny control components have been running beyond the expected lifetime gradually. Many problems from control system have affected the cryogenic system reliability. This paper presents the current status and upgrade solutions of the cryogenic control system of EAST.

INTRODUCTION

EAST (Experimental Advanced Superconducting Tokamak) is the first full superconducting experimental Tokamak fusion device in the world which has been carried out ten campaigns since its implementation at the end of 2005[1]. The cryogenic system is one important subsystem which is to cool down the superconducting magnets and relating components [2]. The total cold mass of EAST is about 250 tons. As shown in Figure.1, the cryogenic system is composed of a helium refrigerator and a helium distribution system. The helium refrigerator is composed of gas management system, compressors station, cold box and 10000 liter Dewar. All heat exchangers, absorbers and four turbine expanders are installed in cold box. The basic design capacity of the refrigerator is 1050 W/3.5K +200W/4.5K +13g/s LHe +12~30kW/80K [3]. To maintain the cryogenic state of the EAST cold components, the helium refrigeration system (HRS) supplies three helium coolants, SHe, liquid helium, and gaseous helium for the SC coils and their feeder lines, current leads, and thermal shield, respectively. The cryogenic distribution system (CDS) has sufficient mass flow to operate each of the SC coils.

Pulsed heat load is the main different factor between the cryogenic system of full superconducting Tokamak system and other large cryogenic systems. The cryogenic system operates in a pulsed heat loads mode requiring the helium refrigerator to remove periodically large heat loads in time. This operation mode must be taken into account for the design of the control system. The EAST cryogenic control system (ECCS) was designed based on DeltaV DCS of Emerson Corporation and has been proved to be safe and stable [4].

However, Manny control components have been running beyond the expected lifetime gradually. Many problems from control system have affected the cryogenic system reliability. This paper presents the current status and upgrade solutions of the cryogenic control system of EAST.

CONTROL SYSTEM OVERVIEW

As shown in figure2, the network of the cryogenic control system is composed of three parts: cryogenic redundant control local area network (LAN), data exchange LAN and main control data server LAN. There is a firewall between the intranet and extranet to ensure the security of the intranet. The cryogenic redundant control LAN employs a three-layer control structure including of the process layer, control layer and supervisory layer. The process layer includes all the field instruments, actuators, sensors and transducers. All the process variables and status information is converted to supervisory by control layer.

The control layer includes two local control cabinets for cold boxes and cryogenic distribution system and a remote control cabinet for compressor station. Local control cabinet includes redundant MD controller, power module and I/O cards. The remote control cabinet includes R5 remote I/O cards and serial card. The remote control system connects to local control by serial cards in terms of MODBUS protocol. New compressors are controlled by PLC and also connect to local control by MODBUS protocol. New PBS turbine is controlled by HEXTR provided by PBS Company and connects to DCS by local bus. New ATEKO turbine is controlled by PLC and as a profibus slave of DCS based on Profibus DP.

The supervisory layer is the interface between the operator and control system, including a professional plus station, four operator stations and an application station as OPC server. In this layer, engineers can configure the program software and manipulate for the cryogenic system.

The data exchange LAN is the auxiliary system of the cryogenic control system. The system includes database server, data acquisition and processing system, web server and FTP server.

The cryogenic data are stored in database server in data exchange LAN connecting to the cryogenic control LAN by application station using OPC protocol. Some important data is sent to the database of the EAST main control system. Some temperature of the magnet coils is acquired by technical diagnosis system. The temperature data are shown in the cryogenic monitor interface. So the process data will be transmitted to the database and

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

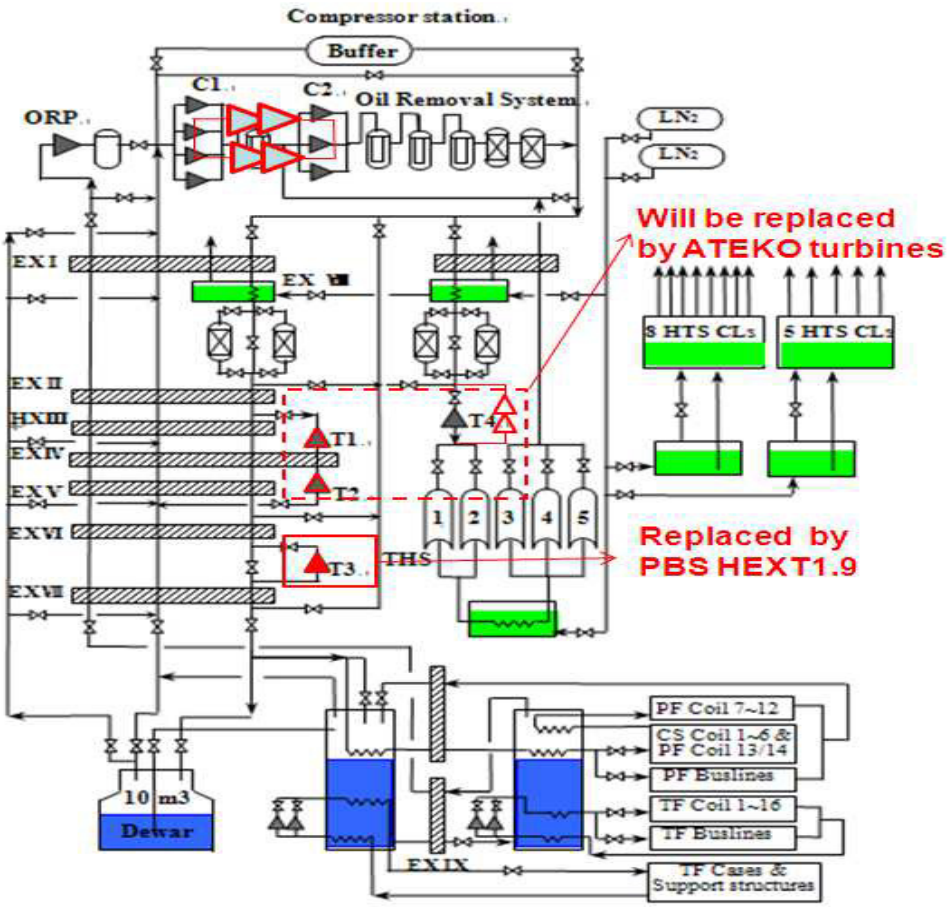


Figure 1 PFD of EAST cryogenic system

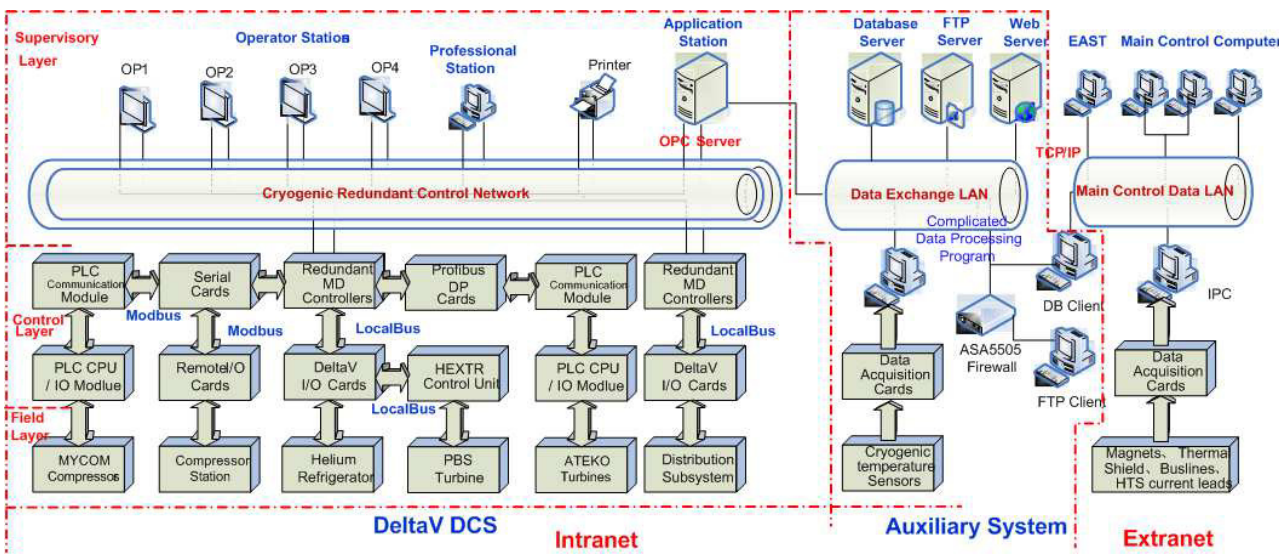


Figure 2 EAST cryogenic control system based on DelatV DCS

exchange system, then send to the DelaV system by OPC protocol.

The data acquisition and processing system has been developed for the temperature and mass flow rate. In order to remotely learn the operation status of the EAST

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

cryogenic system by internet, the remote control supervisory system has been developed based on the TCP/IP architecture.

CURRENT STATUS AND UPGRADE

The ECCS has been in proved to be safe, stable and expand in operation for many years. However, ECCS is still operating on Windows NT operating system. New hardware may not be introduced because of missing driver support. The manufacture will no longer provide supports for the old DletaV version. It's very difficult to backup and recover the control system. Once the engineering station or application station fails, the entire cryogenic control system is in danger of paralysis. Many problems caused by the control system have affected the cryogenic systems reliability. At same time, the performance of control system gradually decreases. The memory of MD redundant controller of DCS is only 32M. The load of MD controller is approach to 90% which affects the running speed of the cryogenic control system. Historical data query is slow, resulting in the operation management efficiency decreased. The alarm system function is limited with many nuisance alarm and records query slowly. In addition to, there are some risks in the communication system because the old type 3COM switch has been off production with no supports. The communication efficiency deceased with error rising, occasional packet losing and network clogging. Some new instrumentation and new solutions can not integrate in DCS system. Therefore, it is necessary to maintenance and upgrade EAST cryogenic control system in order to keep the EAST device safe, reliable and stable in operation.

EAST have two campaigns every year and each campaign will last for 3~5 months. So, the upgrade solution can't affect the EAST campaigns. The two solutions have been considered. Upgrade of the DeltaV system is the near term plan to maintain the EAST operation reliability. The DeltaV version will upgrade from V6.3 to V12.3 which operate in Windows 7. PCs and switch will be replaced [5]. System upgrades will improve the performance of the existing device, eliminate hidden faults and enhance the stability of the DCS system which operates into a new life cycle. New Profibus DP will be installed in the DeltaV system for communication with PLC system.

PERSPECTIVE CONTROL DEVELOPMENTS

The upgrades of the commercial control system are very limited because of the fund shortage. The deltaV system upgrades only include the software and PCs not the device of the control layer. There are still potential risks from the hardware of the control layer. The other solution is to upgrade the ECCS based on the EPICS. The central I/O system will be converted to the Profibus field bus. Control System Studio (CSS) will be as the framework software for the new system. This solution is successful used for the large scale cryogenic control system at DESY [6]. We started the design of new system in this year. The architecture of ECCS based on EPICS is shown in figure 3. Subsequently, we will decide on the specifications for the prototype. The test of the prototype and determination of the specification for the new system will be followed in next year. At last, we intend to introduce the new system based EPICS into operation in parallel with the current system.

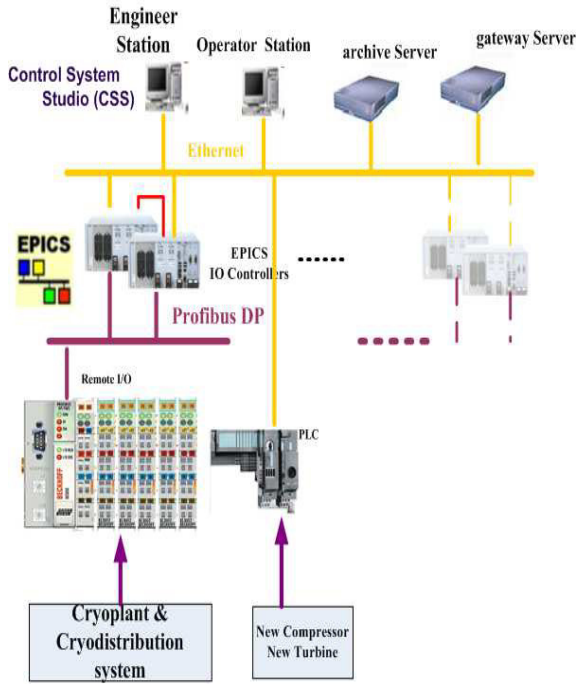


Figure 3: ECCS based on EPICS

CONCLUSION

The upgrades of EAST cryogenic control system have been discussed in this paper. There are one near term and future plan for the upgrades. The DeltaV system upgrades have been implemented on September 2015 and will be tested in the new EAST campaign. In the future, the new EAST cryogenic control system based on EPCIS will be designed and implemented.

ACKNOWLEDGMENT

The authors gratefully acknowledge Mr. Matthias Clausen&MKS-2 group in DESY for their advice. We also thank EAST cryogenic group for their expertise of the cryogenic system.

REFERENCES

[1] Yuanxi Wan, Jiangang Li, Peide Weng et al, Plasma Science & Technology 8(3), 253-254 (2006).
[2] Songtao Wu, the EAST Team, Fusion Engineering and Design 82, 463-471(2007).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

[3] Hongyu Bai, Yanfang Bi, Ping Zhu, QiyongZhang, Keping Wu, Ming Zhuang, Yibin Jin, Fusion Engineering and Design 81, 2597－2603 (2006).
[4] M. Zhuang, L. B Hu，Z.W Zhow, G.H Xia, Fusion engineering and design,2012（87）：2007~2011.
[5] DeltaV Books online, Emerson Process Management Co. 2014. http://www.easydeltav.com.
[6] T. Böckmann, M. Clausen, Chr. Gerke, K. Prü, B. Schoeneburg, and P. Urbschat, AIP Conference Proceedings 1218, 1205(2010).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DESIGN AND STATUS FOR THE ELECTRON LENS PROJECT AT THE RELATIVISTIC HEAVY ION COLLIDER*

J.P. Jamilkowski#, Z. Altinbas, M.R. Costanzo, T. D'Ottavio, X. Gu, M. Harvey, P. Kankiya, R.J. Michnoff, T.A. Miller, S. Nemesure, T.C. Shrey, BNL, Upton, NY 11973, USA

Abstract

The Electron Lens upgrade project at the Relativistic Heavy Ion Collider (RHIC) has reached an operational status, whereby intense, pulsed or DC beams of electrons are generated in order to interact with the RHIC polarized proton beams in both the Blue and Yellow Rings at the 10 o'clock Interaction Region. Interactions between the electrons and protons are utilized to counteract the beam-beam effect that arises from the desired polarized proton collisions, which result in a higher RHIC luminosity. A complex system for operating the e-lens has been developed, including superconducting and non-superconducting magnet controls, instrumentation systems, a COTS-based Machine Protection System, custom Blue and Yellow e-lens timing systems for synchronizing the electron beam with the RHIC timing system, beam alignment software tools for maximizing electron-proton collisions, as well as complex user interfaces to support routine operation of the system. e-lens software and hardware design will be presented, as well as recent updates to the system that were required in order to meet changing system requirements in preparation for the first operational run of the system.

E-LENS GOALS AND DESIGN

The Blue and Yellow e-lenses are installed in the RHIC Ring at Interaction Region 10, in order to partially counteract the head-on beam-beam tune shift effect on the colliding RHIC beams, and thus permit RHIC proton beam operations at higher beam intensities, and therefore higher colliding beam luminosities for the RHIC experiments [1]. First commissioned during the FY2014 run, both electron lenses were successfully operated on a routine basis in a DC mode during the FY2015 RHIC 100 GeV polarized proton run [2].

The main components of each e-lens are the electron gun, electron collector, and superconducting solenoid magnet, though a set of additional systems is required for their routine operation. Beam transport magnets of both superconducting and non-superconducting varieties are controlled through two separate sets of standard VME hardware and software: warm magnets use equipment (PSC, QFG, and PSI) developed for the Injector machines within the Brookhaven National Laboratory (BNL) Collider-Accelerator (C-A) Department, and the cold magnets utilize the RHIC equipment (WFG, MADC) for reference control and readback information. Another system of note is beam instrumentation, which is primarily comprised of BPMs, current transformers, YAG crystal-based beam profile monitors, pinhole raster scan beam profilers, as well as new electron backscattering detectors [3].

TIMING SYSTEM

The Blue and Yellow e-lenses must be capable of operating in two distinct timing modes: pulsed mode for beam diagnostic measurements and system commissioning, as well as DC mode for routine operations. Activity in pulsed mode must be synchronized with the RHIC Event and Beam Sync Links, such that the timing system is capable of pulsing electron beam from the e-lens as a desired RHIC hadron bunch is passing through the Blue or Yellow Ring in IR10. To achieve these requirements, both e-lenses are dependent on the existing RHIC timing links as shown in Figure 1.

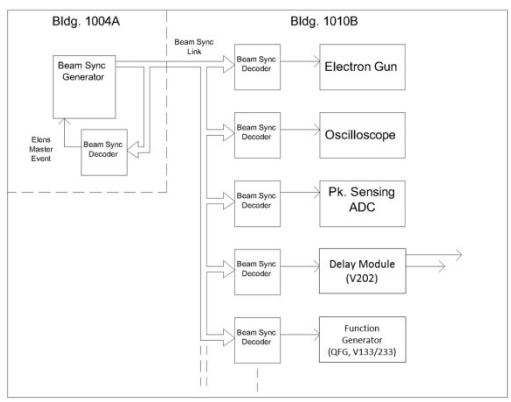


Figure 1: Schematic of the e-lens timing system, including RHIC components (building 1004A) and e-lens-specific components (1010B).

Software Interface

In order to enable the level of system flexibility required to operate the e-lens equipment under many different timing conditions while streamlining the user interface, a server-based ADO program is available to control the system that was written in the Java language. While it was not the first Java server written at the C-A Department, this software interface development project served as an important testbed for use of Java in developing ADO software interfaces rather than the standard C++ option.

*Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy.
#jpj@bnl.gov

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MACHINE PROTECTION

A total of seven independent e-lens subsystems have been identified that require automated protection from abnormal machine conditions. Thirty-seven quantifiable status conditions serve as inputs to the Machine Protection System (MPS) logic for inhibiting these systems. As the scope of the protection functionality inputs and outputs fits well with the capabilities of the familiar National Instruments cRIO FPGA-based platform, it was selected for the hardware implementation. The response time of the MPS logic in similar configurations was measured to be ~1µs, though this is dependent on the choice of input modules.

Software integration of the MPS with the Collider-Accelerator Controls System is achieved using the Simple Network Access Protocol (SNAP), which supports bi-directional communication between Accelerator Device Object (ADO) server-based software and LabVIEW software running on the cRIO device that incorporates the MPS logic.

BEAM INSTRUMENTATION

Beam position monitoring of the electron beam is achieved using RHIC-style dual-plane BPM electronics, which incorporate analog/digital integrated front ends (IFE) using 16 bit digitizers in order to achieve a 1µm resolution over a ±32mm range [4]. Data from the BPMs has been used for steering optimization of the e-lens electron beam relative to the IR BPM position readings for the hadron beams in the Blue and Yellow Rings.

Initially commissioned during the FY2014 RHIC Au-Au run, electron Backscattering Detectors (eBSDs) have become important diagnostic tools for aligning the electron beam from the e-lenses with the hadron beams in the Blue and Yellow Rings at RHIC [5]. Since the scaler readings from the eBSDs are proportional to the collision angle between the two beams, we have been able to successfully incorporate the new data into the existing RHIC Luminosity and IR Steering (LISA) application shown in Figure 2. This tool allows RHIC machine operators to optimize the impact of the e-lenses on the RHIC luminosity on demand.

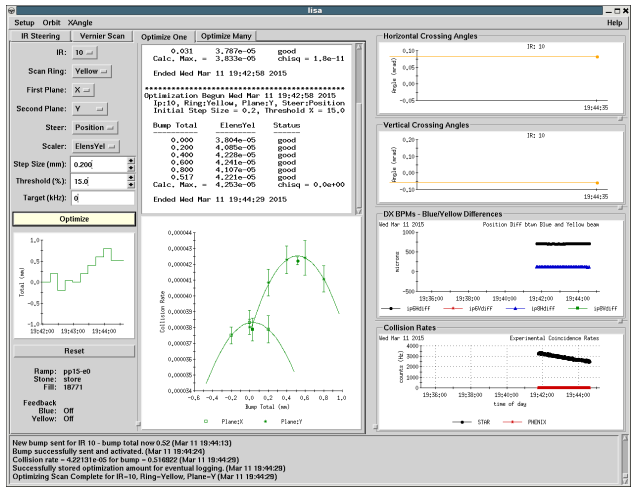


Figure 2: The RHIC Luminosity and IR Steering application (LISA) output after automated optimization of the Yellow e-lens beam alignment using eBSD data.

NEW USER INTERFACE

In addition to the more commonly used user interfaces within the Controls System at the Brookhaven National Laboratory C-A Department, the Syndi editor and display application has been adapted from the original version developed at FermiLab. Due to the compact and complex nature of the electron lenses, synoptic displays created through this toolset have evolved to serve an important role in the operation of the equipment. Figure 3 shows an example of the Blue e-lens Syndi display, which includes statuses of critical sub-systems as well as a select set of control points that are used during routine operations.

Development of the e-lens control pages themselves has migrated since their inception between the Syndi software developers, the project physicist, and ultimately to a member of the RHIC Operations group. While technically capable of embedding complex business logic within each page, we have elected to minimize the amount of such logic within the e-lens pages in favor of server-based software that leverages critical functionality, such as parameter caching. This also reduces the complexity of the page design, and thus improves reliability and maintainability. Our experience during RHIC run FY15 was that such a scheme allowed for maximum flexibility, while adding slightly more complexity over other more commonly used implementations at the BNL C-AD.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

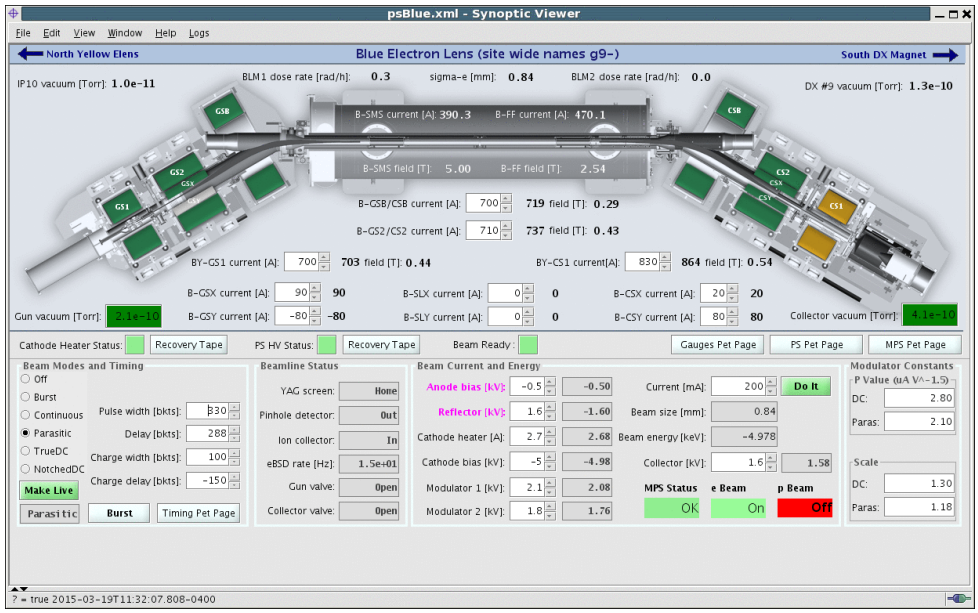


Figure 3: Synoptic user interface for the Blue e-lens created using the Syndi editor.

POWER SUPPLIES

Cryogenic magnet and power supply operations for the e-lenses require coordinated ramping methods, both to avoid unnecessary beam losses and to avoid quenching the superconducting magnets themselves. One method has been developed and exercised during the FY2015 run using ADO server-based software written in C++ that provides such functionality with a streamlined user interface [6] that is capable of controlling RHIC-style Wave Form Generators (WFGs).

CONCLUSION

The complex new Blue and Yellow e-lens systems have been successfully commissioned and operated using a hybrid design of existing controls systems with new equipment and software. COTS hardware is used to good effect as part of the Machine Protection System. Existing software tools, such as LISA, have been adapted to provide novel operating modes that contribute directly to of the new e-lens systems, synoptic interfaces have been developed using the Syndi display editor that have a synergistic relationship with existing interface options.the improvement of the luminosity of the RHIC colliding hadron beams. In order to better visualize the operation

REFERENCES

[1] X. Gu et al., "RHIC Electron Lenses Upgrades," IPAC'15, Richmond, VA, USA p. 3830 (2015); http://www.JACoW.org

[2] X. Gu et al., "RHIC Electron Lenses Upgrades," IPAC'15, Richmond, VA, USA p. 3830 (2015); http://www.JACoW.org

[3] P. Thieberger et al., "The Electron Backscattering Detector (eBSD), A New Tool For the Precise Mutual Alignment Of The Electron And Ion Beams In Electron Lenses," IBIC'14, Monterey, CA, USA p. 129 (2014); http://www.JACoW.org

[4] T. Satogata et al., "RHIC BPM System Modifications and Performance," PAC'05, Knoxville, TN, USA p. 2021 (2005); http://www.JACoW.org

[5] P. Thieberger et al., "The Electron Backscattering Detector (eBSD), A New Tool For the Precise Mutual Alignment Of The Electron And Ion Beams In Electron Lenses," IBIC'14, Monterey, CA, USA p. 131 (2014); http://www.JACoW.org

[6] P. Kankiya et al., "Synchronized Ramping of Magnet Power Supplies for Streamlined Operation at Energy Recovery Linac (ERL) and Electron Lens (ELENS)," MOPGF066, these proceedings, ICALEPCS'15, Melbourne, Australia (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SIRIUS CONTROL SYSTEM: DESIGN, IMPLEMENTATION STRATEGY AND MEASURED PERFORMANCE

J.P.S. Martins, M. Bacchetti, E.P. Coelho, R.F. Curcio, J.G.R.S. Franco, R.P. Lisboa, P.H. Nallin, A.R.D. Rodrigues, L.D.S. Sachinelli, M.E. Silva – LNLS, Campinas, Brazil

Abstract

Sirius is a 3 GeV synchrotron light source that is being built by the Brazilian Synchrotron Light Laboratory (LNLS) [1]. The Control System will connect and operate all the equipment along the whole machine. The main goal of the Sirius Control System is to be distributed and digitally connected in order to avoid analog signal cables. A three-layer topology will be used [2]. The equipment layer uses RS485 serial networks, running from 6 to 12 Mbps, with a light proprietary protocol and CPU boards with proprietary hardware stacked on it, in order to achieve good performance. The middle-layer, interconnecting these serial networks, is based on Beaglebone Black single board computer and commercial switches. Operation layer will be composed of PC's running EPICS client programs. Special topology will be used for Orbit Feedback with a dedicated commercial 10Gbps switch. This paper will discuss the details of the Control System components, the implementation strategy for hardware and software and show some results of the prototypes.

INTRODUCTION

The Sirius Control System is designed to be scalable, distributed and easy to maintain. Currently, we are developing a generic solution for the hardware of the Control System – analog and digital interfaces, and also serial communications interfaces – to provide control features for the most of Sirius systems: vacuum system, pulsed power supplies, magnets power supplies, RF system, etc. The main characteristic of this solution is to be compatible with commercial low cost CPU boards. We achieve this using the SPI standard interface. The very first prototype for the Sirius Control System hardware was a homemade ARM CPU board with analog and digital modules, called PUC (Universal Control Board). Several tests have been done with this board and modules, and some results will be shown in this paper.

Besides the basic input/output, the hardware supports special features for synchronous operations, like curves for energy ramping of the booster and cycling magnets, and a circular buffer for post mortem acquisition.

With this strategy, the Controls Group will provide a low budget generic solution for hardware and software, which is reliable and of good performance in order to meet the requirements of operation of the various systems from Sirius.

HARDWARE IMPLEMENTATIONS

The main goal of the hardware solution provided by the Controls Group is to be easily adaptable to the equipment being controlled, avoiding the traditional "crate mounting" for a distributed installation, and sometimes inside the own equipment, preventing analog signals to travel over long distances.

Hardware Platform - GESPICON

This platform is under development and aims to be independent of a custom CPU board (like the PUC Base Board). Thus, any commercial CPU board, even the Beaglebone Black, can be used as embedded controller for the interface modules; it just has to have an SPI peripheral and some GPIOs. Figure 1 shows the components of the platform, called GESPICON (Generic SPI Controller). The connection of the stacked boards with the controller is done by a special bus, called SPIxxCON bus. This bus has the traditional SPI pins for serial data interchange, and some GPIOs for configuration of the transactions. It has also the pins for power supply and the optional use of quad-SPI features. The protocol of this bus is capable of addressing 8 module boards with 8 SPI devices in each one. Thus, one CPU controller can manage up to 64 SPI devices attached to it.

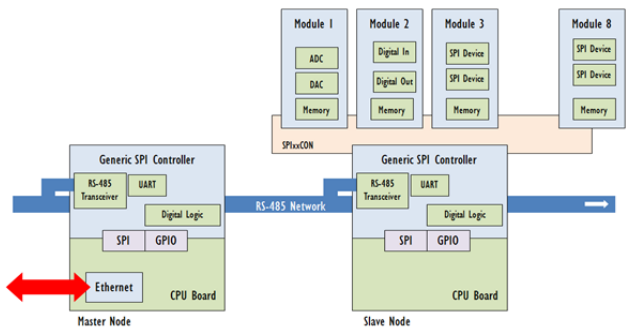


Figure 1: Schematic of the GESPICON hardware platform.

Another important feature of this implementation is the possibility to embed SPI flash memories on the interface modules. Thus, any interface can have static information used by the controller for auto-configuration. At boot time, the CPU module will search for each stacked module memories and download the data on it, which can be:

- Source code (Python or C) for the driver interface of the module;
- EPICS parameters for IOC configuration;
- Any parameter of the module devices;
- BSMP entities description (defined next page);

This system makes the CPU software independent of the interface modules drivers. Any module that would be developed in the future will be compatible with the CPU,

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

with no need of software upgrades. These features increase the modularity and the flexibility of the whole system.

PUC CPU Module

The PUC (Universal Control Board) was the very first prototype developed by Controls Group. The CPU module (called "base board") is a homemade device embedded with an ARM Cortex M4 microcontroller, with 256 KB of RAM memory and 2 MB flash memory, running at 180 MHz. The base board also has flash and RAM memories (external from microcontroller), which are used to save waveforms and to hold fast acquisition data. The main communication interface is the RS485 serial, running at 6 Mbps baudrate. Another communication facility is the Ethernet interface. The MAC (Media Access Control) layer is embedded in the microcontroller and the LwIP (Lightweight IP Library) implements the TCP/IP stack.

Analog and Digital Modules

The digital board has 8 digital inputs and 8 digital outputs, TTL level, for digital commands and readings.

The analog board has one ADC (AD7634) and one DAC (AD5781) both with 18 bits resolution, for analog input and output. The working range of the converters is from -10V to +10V. This means that the theoretical resolution is 17 bits for 0 to +10V and 16 bits from 0 to +5V. This module was specially designed due to the fact that these kind of high precision interfaces are unusual to find commercially.

Three basic tests were done with these interfaces: long term stability, linearity and repeatability. The setup for all tests was three PUCs with one analog board module stacked. The analog output was connected to analog input of the same board. The measurements were done by a 2-channel Agilent 34420A 7½ digits multimeter and a HP 34401A 6½ digits multimeter. The multimeters were configured with an integration time of 100 NPLC (Number of power line cycles), and the ADCs with an integration of 20k samples over one second.

Long Term Test was performed setting the analog output once and measuring the multimeters and analog input during 24 hours. Some results are presented in Fig. 2 (analog output) and Fig. 3 (analog input).

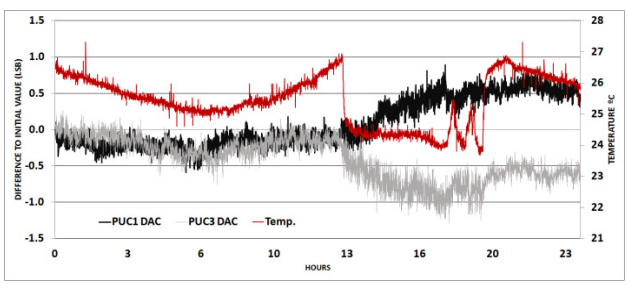


Figure 2: Long term of analog outputs from PUC1 and PUC3, fixed on -9V and +9V, respectively, along with temperature (right axis).

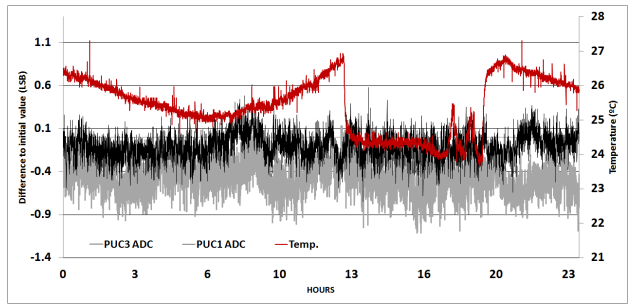


Figure 3: Long term of analog input from PUC1 and PUC3 (with temperature).

The result shows that stability is quite adequate. The drift of the voltage references of the converters increases as the output reaches the full-scale value (10V), but stays in a reasonable value.

Linearity Test was performed setting the analog output in 5 ranges of voltages (-9V, -5V, 0V, 5V and +9V), with 512 points around the center values. Table 1 and Table 2 below present some results of analog output module differential non-linearity and integral non-linearity tests, respectively.

Table 1: Analog Output Module Differential Non-linearity

DNL	PUC1		PUC2		PUC3	
Range	Min (LSB)	Max (LSB)	Min (LSB)	Max (LSB)	Min (LSB)	Max (LSB)
-9V	-0,235	0,232	-0,165	0,168	-0,396	0,402
-5V	-0,118	0,117	-0,090	0,098	-0,165	0,177
0V	-0,283	0,289	-0,275	0,273	-0,039	0,019
+5V	-0,138	0,138	-0,085	0,085	-0,161	0,170
+9V	-0,193	0,203	-0,156	0,184	-0,295	0,254

Table 2: Analog Output Module Integral Non-linearity

INL	PUC1		PUC2		PUC3	
Range	Min (LSB)	Max (LSB)	Min (LSB)	Max (LSB)	Min (LSB)	Max (LSB)
-9V	-0,096	0,762	-0,280	0,122	-0,717	0,104
-5V	-0,226	0,129	-0,248	0,059	-0,158	0,217
0V	-0,366	0,000	-0,409	0,000	-0,017	0,083
+5V	-0,082	0,257	-0,134	0,157	-0,133	0,263
+9V	-0,224	0,294	-0,304	0,096	-0,249	0,428

The results of INL and DNL of all boards are adequate, staying in ±0.5 LSB range for the analog outputs, and ±1.5 LSB range for the analog inputs.

Repeatability Test keeps the same setup. The analog outputs were set repeatedly with the following values:

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

+9V, +5V, 0, -5V and -9V, during 12 hours, at each update, the multimeters and ADCs were read. No missing steps were found in all measurements. The results of the relative error at the 5 ranges are presented in Fig. 4 below.

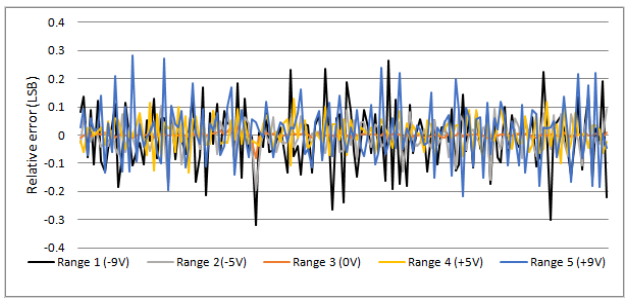


Figure 4: Repeatability test of PUC1 analog output (relative errors).

Beaglebone Black

The Beaglebone Black [3] is an open hardware single board computer based on ARM Cortex A8 running at 1 GHz. The rich set of features of this hardware makes it a good solution for an embedded platform:

- 512 MB DDR3 RAM, 4GB on-board eMMC flash storage;
- Linux operating system support;
- Flexibility and modularity;
- Fanless;
- 2X real-time 32 bits microcontrollers (PRU processors);

The Beaglebone Black fits perfectly to be used as a high level platform for communication with the Control System Hardware, and also as a CPU board for the custom analog and digital interfaces of the Control System.

To implement the fast RS485 interface, we designed a cape board for the Beaglebone Black using an external UART (MAX3107), which will reach up to 12 Mbps baudrate. The interface to this component is done by the PRU (Programmable Real-Time Unit), a real-time processor embedded in the ARM CPU of the Beaglebone Black (which has two PRU units). To optimize the software design, we created a C library to interact and manage the PRU firmware. This library has an API for sending and receiving bytes from the external UART, and also for configuration of the device. The result is a high-performance serial device with great processing power and all the facilities of an embedded Linux system.

SOFTWARE IMPLEMENTATIONS

The Sirius Control System will use EPICS middleware, one of the most used frameworks in Particle Accelerators Controls System. For the low level communication, we developed a light proprietary protocol, called BSMP (Basic Small Messages Protocol).

BSMP (Basic Small Messages Protocol)

The BSMP describes two layers of the network model: transport and application. The transport layer transmission unit is the *packet*. Each packet holds a *message*. The packets in a serial network have a well defined format: one byte for destination address, n bytes for the message and one byte for checksum. The addressing can be individual or in multicast groups. The protocol message in the application layer has the first byte for command, two bytes for size and finally n-bytes for the payload. Thus, the whole BSMP packet has only 5 bytes of overhead, for payloads up to 65535 bytes.

Devices using the BSMP protocol manipulate protocol entities, which can be up to four categories: variables, group of variables, curves and functions. To manage and implement the BSMP protocol, a C library (and a Python binding) was developed. The API of the library implements all the routines for data encapsulation, de-encapsulation, entities values attribution and protocol validation. To develop a new hardware and use BSMP, the developer only needs to configure the corresponding entities.

EPICS Device Support

In order to use the EPICS facilities, we developed a Device Support for the PUC prototype. An EPICS Soft IOC for the Beaglebone Black master was configured, because the low level hardware access is done via message based devices. The Device Support driver was created using asynDriver framework [4]. In order to embed the BSMP and PRU Serial libraries on the driver, we developed an Asyn Driver Support, managing the low level communication tasks. Above the driver support, there is an Asyn Port Driver in C++, implementing all the parameters that the hardware could have: analog I/O, digital I/O, waveform I/O and static configurations. The same strategy can be used to develop the EPICS Device Support of the new hardware over the SPIxxBUS. The strategy of message-based devices also permits the use of other software for low level interfaces and data structure, like Redis IO suite [5].

SYNCHRONOUS OPERATIONS

Sirius Control System must support synchronous procedures for many of the machine systems. As a main example, there is the booster energy ramp, performed by the magnet power supplies and RF systems. The Control System hardware supports synchronous operations broadcasting serial messages over the RS485 network, in order to reduce the number of signals and cables from the Timing System. Thus, the triggers only need to reach the single board computers, which will send the broadcast high-priority messages over the serial network, as fast as possible.

To validate this strategy, several tests were performed. The object of measurement is the latency between the reception of the trigger on a single board computer and a flag assertion on a slave node of the RS485 network. The

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

first test uses a Beaglebone Black with our homemade RS485 cape as the master and three PUCs as slave nodes. The trigger signal was a 3.3Vpp pulse, 5% duty cycle, 5 kHz, produced by a HP 3314A Function generator. As soon as the trigger is detected at the Beaglebone Black GPIO (PRU Unit), the master sends a high priority serial packet over the RS485 network at 6 Mbps baudrate. The flag used to measure the reception of the node is the trigger for the DA converter of an analog interface module. This setup emulates the booster energy ramp. We run the experiment 4 times, acquiring 2 minutes of data in each run. Table 3 shows the average of the obtained values from the 4 runs.

Table 3: Latency Measurement Using Beaglebone Black as Master and PUC as Slave

Latency	Min (us)	Max (us)	Avg (us)	Std. dev (ns)
PUC 1	13.91	14.00	13.94	17.13
PUC 2	13.94	14.00	13.97	14.54
PUC 3	13.95	14.04	13.98	18.45

Other tests were done to evaluate the Beaglebone Black as master and as slave on the RS485 network. In this setup, the trigger signal is generated by the own Beaglebone Black master node, using the PWM peripheral. It's a 3Vpp pulse with 2% duty cycle and 100 Hz. This signal is in a loop back to a GPIO of the PRU processor, which sends the broadcast serial message over the network (at 6 Mbps baudrate). The slave Beaglebone Black PRU detects and decodes the broadcast message, raising a flag when the packet is decoded. The results of the measurements of the reception latency are presented on Table 4.

We also tested the Beaglebone as master and slave but running the software from the ARM core, under Linux environment. The latency increased to hundreds of microseconds and the jitter increased to tens of microseconds, which invalidates this topology.

Table 4: Latency Measurement Using Beaglebone Black as Master and Slave, Running Firmware on PRU Processor

	Run #	Min (us)	Max (us)	Avg (us)	Std. dev (ns)
PRU (master) x PRU (slave)	1	23.89	23.99	23.94	18.91
	2	23.89	23.99	23.94	18.73
	3	23.89	23.99	23.94	19.36
	4	23.89	23.99	23.94	18.97

The repetition rate of the Sirius Booster for top-up operation is 2 Hz. The specification of the ramping curves is about 1000 steps each curve. Thus, for 2 Hz operation, the period of each step is 500 us. The results of the first two tests prove that the system can achieve the necessary performance for booster operation. Even if the repetition rate increases to 5 Hz, (with 200 us each step) the latency and jitter of the serial broadcast triggers match the requirements. The results can be improved when using 12 Mbps baudrate on the serial network.

Another feature of this topology is the selective interrupt configuration, when the slave nodes respond to specific broadcast messages, allowing more flexibility for the profile of ramping curves. This can be useful for machine operation.

CONCLUSION

The design of the main components of Sirius Control System was presented. The implementation strategy is to provide a robust, modular, flexible and distributed hardware platform for machine operation, in addition to EPICS compatibility on the software side. The analog modules characterization tests are quite reasonable. The synchronous operations support proved to match the specification of the Sirius booster energy ramp. The next designs under development by the Controls Group will increase the performance and flexibility of the whole system.

REFERENCES

[1] L. Liu et. al, "Update on Sirius, the new Brazilian Light Source", MOPRO048, Proc. IPAC2014, http://jacow.org
[2] J.G.R.S. Franco et. al, "Sirius Control System: Conceptual Design", MOMIB01, Proc. ICALEPCS2013, http://jacow.org
[3] http://www.beagleboard.org/black
[4] http://www.aps.anl.gov/epics/modules/soft/asyn
[5] http://redis.io

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ARIEL CONTROL SYSTEM AT TRIUMF - STATUS UPDATE

R. Nussbaumer, D. Dale, K. Ezawa, K. Fong, H. Hui, M. Iranmanesh, J. Kavarskas, D. Morris, J.J. Pon, S. Rapaz, J.E. Richards, M. Rowe, T. Tateyama, E. Tikhomolov, G. Waters, P. Yogendran, TRIUMF, Vancouver, Canada

Abstract

The Advanced Rare Isotope & Electron Linac (ARIEL) facility at TRIUMF has now reached completion of the first phase of construction; the Electron Linac. A commissioning control system has been built and used to commission the electron gun and two stages of SRF acceleration. Numerous controls subsystems have been deployed including beamlines, vacuum systems, beamline diagnostics, machine protect system interfaces, LLRF, HPRF, and cryogenics. This paper describes some of the challenges and solutions that were encountered, and describes the scope of the project to date. An evaluation of some techniques that had been proposed and described at ICALEPCS 2013 are included.

SCOPE OF CONTROL SYSTEM

The e-Linac was to include a total of 3 stages consisting of 2 cavities per stage except for the first stage, which has only one, of Superconducting Radio Frequency (SRF) acceleration. The performance of the installed cavities has allowed postponing the third cryomodule. With the installation of the third cavity greater than 30MeV will be achievable. The second accelerator stage is planned for upgrade to 20MeV acceleration in one year from this writing. Deployment of the control system has been performed in subsystems, with the intention to isolate interaction between the subsystems, as well as to dedicate personnel to the various subsystems. Different broad control system technologies are used in each subsystem, and this allows personnel to focus on smaller classes of methodologies.

Numeric Facts and Figures

Devices under control	1160
Discrete IO Points	11155
EPICS IOCs	22
IOC Hosts	16
EPICS Device Support types	24

Figure 1: Scale of the control system at this writing

DEPLOYMENT STRATEGIES

A stated goal of the e-Linac control system was to re-use as much existing design and methodology as possible. The TRIUMF Controls Group has a significant body of work and methodology that was built up from prior projects. This includes a system of tools to build EPICS runtime databases, EPICS Operator Interface (OPI) screens, consistency checking in PLC interlock programming, and development of a consistent look and feel of control room consoles.

Part of the technology deployed in earlier TRIUMF control systems was based on consistent device styles chosen by other groups, especially vacuum, power supplies, and beam diagnostics groups. Despite the intent to develop no new methods and technologies, the style of control system interfaces used by equipment specialists from other disciplines did not accommodate the controls group strategy, and considerable re-work of existing methods was required. This paper will focus on some details of methods not previously included in control systems deployed at TRIUMF.

CONTROLS SUBSYSTEMS

Vacuum Systems

A new strategy for interface to vacuum systems was required when the vacuum group chose to introduce the concept of portable pumping carts. In this system, a small number portable carts containing turbo pumps, backing pumps, vacuum gauges and related valves was fabricated. These carts are arranged to pump down vacuum spaces at 26 discrete locations along the beamlines, according to stages of commissioning and following maintenance procedures requiring vacuum breakage.

Each turbo pumping cart is equipped with a small PLC drop that contains sufficient Input/Output support to drive and read the vacuum devices contained on the cart. A central vacuum controls PLC provides the program logic and EPICS interface for supervisory control and display. Asynchronous connection and disconnection of the central PLC is successfully accomplished by exploiting the ability of the Schneider[1] Quantum PLC to perform IO scans. IO scanning allows the PLC program to detect connect events and disconnect events seamlessly, and to map IO registers to beamline location-specific process variables.

The PLC reads dedicated loopback cables at each pumping cart and connection port, in order to permit the location of each connected cart to be tracked. This information is used to allow OPI displays to reflect the connection status of each cart and each beamline connection port in real time.

The ability to seamlessly connect and disconnect field IO at runtime has resulted in estimated saving of CDN $28000 in PLC hardware by removing the need for redundant dedicated PLC IO hardware. In addition, the reduction in cabling, estimated at 10000 metres, served to relieve the constraint of cable routing space availability to the Electron Hall where the accelerator and beamlines are

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

located. This strategy is now considered a significant success.

Cryogenics Systems

The SRF accelerator modules rely on the use of liquid helium cryogenics, and sub-atmospheric pumping to establish operation at 2 degrees Kelvin of the accelerating cavities. To accomplish this, a hybrid of commercial and in-house cryogenics system has been developed. Initially, the strategy was to model the prior work in the TRIUMF ISAC-II facility. In this arrangement, a commercial turnkey helium liquefaction plant would be augmented by an in-house cold-distribution system. Air Liquide Advanced Technologies[2] (ALAT) was chosen as the vendor of the helium liquefaction system (cold box). An interface to TRIUMF equipment and signalling was defined, allowing some aspects vital to operation of the cold box, to be provided and installed by TRIUMF, and signals provided to the ALAT PLC.

The cold delivery system and sub-atmospheric pumping system are composed of various valves, pressure, temperature and pressure monitors, as well as four Busch[3] (model here) mechanical pumps. These are controlled by one Schneider Quantum PLC with IO drops at three different geographic locations.

The convention used to deploy control systems heretofore by TRIUMF controls has been a strictly device oriented system. In this arrangement, there is little coupling of device control, except in ways that are well defined where devices provide status information that is used in machine protection interlocks. Automation of processes has been implemented in only very limited ways.

As a departure from this, there is a requirement to provide some level of autonomous behaviour. The intention is to remove or reduce the need for human operator interaction with the cryogenics system, as well as to improve system performance by exploiting the faster response time capability of a PLC. This level of automation is expected to be accomplished as a new layer of logic, which can be used independently of the machine protection and remote control role of the control system. Strategically, this is intended to allow a segmented roll-out and commissioning of the automation portion of the control system. Experts from the Cryogenics Group are collaborating closely with the Controls Group to define what form this will take.

Initially, there was a clear intention to use the ALAT cold box as a turnkey-only package. Due to the dedication and expertise of one PLC expert in the Controls Group, some modifications to the delivered system have been identified and proposed. These include improvements to the way the ALAT system interfaces to two Kaiser Helium compressors, and segmentation of some PLC controls to reflect geographic separation and functional relationships between elements of the ALAT cold box. These enhancements will also facilitate the aforementioned automation efforts, and the interface to the supervisory EPICS controls.

Beam Optics Systems

Beam optics in the e-Linac uses conventional electromagnets for steering and focusing the electron beam. The control system, then, is responsible for control of power supplies. As a departure from prior work at TRIUMF, the magnet power supplies used in the e-Linac are mostly equipped with built-in Ethernet LAN interfaces, and support a SCPI style of command and query language.

These power supplies are controlled directly from EPICS IOCs, using the combination of EPICS asyn and streamDevice support packages. The system of EPICS records used to control power supplies through analog and digital interfaces was duplicated, using the streamDevice over asynDriver/TCP EPICS devices support. This was successful in creating user interfaces and behaviours that closely resembled prior implementations. In addition to the LAN/SCPI interfaces, three power supplies were purchased from BiRa[4] Systems. These are conventional off-the-shelf power supplies that are augmented with a closed loop stability controls system, and are driven by a proprietary EPICS device support layer.

Monitoring of power supply parameters requires continuous polling, which introduces high volumes of network traffic. In order to isolate the effect of the beam optics traffic from the rest of the control system network, dedicated EPICS IOCs with dual Ethernet interfaces were used to control the power supplies. The strategy is to use one Ethernet LAN as a field bus for the power supplies, while the other network interface provides a conventional TCP/IP interface to the IOC host, including the EPICS Channel Access protocol traffic.

For reasons of economics, the power supplies chosen were of a unipolar nature, even though most beamline requirements dictated bipolar operation in order to support magnet degaussing. In-house polarity switches were built, and these are controlled by the EPICS IOCs using VME based digital IO to drive and monitor the state of the polarity switches. EPICS record logic to support these polarity switches has been successfully implemented.

Beam Diagnostics Systems

Several classes of beam diagnostics devices not previously implemented in a TRIUMF control system have been introduced. These include a Fast Wire Scanner (FWS), several viewscreen systems allowing beam physicists to visualize the electron beam, and a non-intercepting RF-Shield used to measure beam current in real time, and in-house designed and fabricated Beam Position Monitors (BPMs).

As well as new device types, the convention previously used to control stepping motors has been changed from the now obsolete Oregon Micro Systems (OMS) VME based motor controllers to Galil[5] controllers. In a previous endeavour at TRIUMF, crafting device support code to augment the EPICS Motor Record interface to the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Galil controllers was found to be unsuccessful by. New motion control for beam diagnostics devices uses streamDevice EPICS support to communicate with the Galil motor controller on a dedicated Ethernet LAN. Control system logic mimicking the function of the EPICS Motor Record was developed, in order to maximize compatibility with prior work, and with a view to later successful implementation of motor record code to support the Galil controller.

There exists in certain instances, the possibility of moving beam diagnostics devices to collide in space within the beamline. A dedicated Schneider PLC has been implemented with the sole purpose of redundant protection against such collisions.

Viewscreens using the EPICS asyn-based Area Detector package were created by a University based partner, and provided to TRIUMF as turnkey packages. These use the same Galil model motor controllers as other beam diagnostics devices. The original delivery provided EPICS IOC hosts that were configured with Ubuntu Linux, and booted from spinning magnetic media. The operating system and filesystem structure for these has been replaced with the diskless booting system that is now a standard in EPICS based controls at TRIUMFBeam Position Monitors (BPMs) were developed by the Beam Diagnostics group, in collaboration with the Controls Group. Communication with the BPMs is via TCP/IP using EPICS streamDevice to write commands and read query replies.

RF Controls

Low Level RF (LLRF) controls are produced by the LLRF Group at TRIUMF. This permits a system of fast feedback control of the LLRF system, while interfacing to the e-Linac EPICS control system on a supervisory basis. For the e-Linac Control System, a new EPICS interface was developed by the LLRF Group, enabling the use of conventional streamDevice interfacing, and replacing a legacy system that had become inconsistent with current practice.

The SRF cavities receive High Power RF (HPRF) energy from two Ampegon[6] Klystron power supplies. These are turnkey systems containing embedded Siemens[7] PLCs. The principle control system interface to HPRF equipment is via these embedded PLCs, using an in-house EPICS device support package.

HOST SYSTEMS AND NETWORKING

A small variety of EPICS IOC host types have been deployed in the e-Linac Control System. VME based IOCs are used where direct hardware interfaces are required. This strategy is inherited from prior work. Other alternatives were contemplated briefly, however there were too many elements that either required VME, such as the Machine Protect System Long Ion Chamber detector interfaces from Thomas Jefferson Laboratory, or would require replacement of in-house designs that had already been done for the VME form factor.

Other EPICS IOC hosting has been implemented using small form factor commodity PCs equipped with Intel Atom processors and dual ethernet interfaces. Because of the heavy use of EPICS streamDevice over Ethernet TCP/IP, commodity off-the-shelf hosts are deemed suitable, as long as they are capable of running Linux as an Operating System. These are also used to communicate with PLCs, both in-house programmed Schneider PLCs, as well as Siemens PLCs that are embedded in the turnkey cryogenics system as well as two turnkey high power Ampegon Klystron power supplies

Dell server class x86 computers are used as fileservers to host all EPICS binaries, runtime databases, configuration files, and to store user data. These are running Debian Linux, which has been adopted as the standard for fileserver and development host applications in the TRIUMF Controls Group.

The front-end IOC hosts use a version of Linux that was tailored in-house for use in EPICS IOC hosts. The OS is booted over the network using the industry standard PXE protocol. Both the Linux kernel and a customized RAM based root filesystem are downloaded from TFTP servers are boot time, and the hosts run disklessly, mounting fileserver NFS shares for non-volatile storage. The custom root filesystem inherits startup arguments from the PXE configuration file, so that specific applications can be centrally assigned to specific hosts.

This system has been updated with some new features since it was reported in ICALEPCS 2011[8]. In addition to scripting that uses kernel arguments to launch specified applications, a system of host-specific (in contrast to application-specific, since multiple EPICS IOC applications can run on a single host) configuration procedures can be launched from the central boot host. This is used for such things as installing iptables based firewall rules that allow the host to provide transparent access to PLCs on the fieldbus network interface, and to launch DHCP servers for many of the controlled devices that use TCP/IP Ethernet interfaces. In addition, a system of shared applications that were chosen not to be included in the RAM based root filesystem can be accessed at runtime for test and diagnostics purposes.

A control room dedicated to the e-Linac has been established, using conventional deskptop computers, and running the Debian version 6 Linux, and the KDE desktop framework. A total of five console computers arranged with four monitor and six monitor video configurations is serving the purpose of e-Linac commissioning, and is expected become part of a TRIUMF-wide integrated control room following completion of the ARIEL facility.

Network Structure

The network architecture used in the e-Linac controls system provides for two tiers of Ethenet TCP/IP networks. One is the general purpose Control System network which carries EPICS Channel Access traffic and other TCP/IP traffic such as NFS and SSH communications between the IOC hosts control room consoles, control

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

system user hosts, and fileservers. The other tier is a system field bus oriented networks which are isolated behind IOC hosts. These carry traffic that is only related to the devices under control of the EPICS IOCs. Individual IOC hosts use their respective fieldbus networks to send and receive data between the IOC and each of the devices under control. This strategy is intended to keep overall traffic levels on the principle network low, as well as to allow isolation of any device which may become faulty and disruptive to other control system elements. In addition, it allows conservation of network IP addresses, since the field bus networks are assigned private LAN addresses from the 192.168.0.0 class.

Beyond the standard two tiers of ethernet networks, the PLCs in use in the e-Linac Control System use their own private ethernet networks. These PLC networks carry only Modbus and QEIO traffic between PLCs and their IO drops, and further segment the overall network infrastructure.

The use of a highly segmented network topology, coupled with fully functioning Linux computers as IOC hosts allows easy network diagnostics, configuration and monitoring of network traffic. Colour coding of ethernet cabling for the respective tiers promotes easier understanding of network connection and topology in the field.

CONTROLS GROUP ECOSYSTEM

In prior control system projects at TRIUMF, a system of tools and methodologies has been developed to allow for high productivity, consistent product output, and low error rates in control system production and deployment. The tools used to perform these tasks include numerous in-house scripts, libraries, and databases, and tools. These are used to generate EPICS OPI screens, EPICS runtime databases, and to generate much of the configuration data that is used in running EPICS IOCs.

Base 3.14.12.3
Alarm Handler
EDM
StripTool
Gateway
TRAR (TRIUMF Archiver)
TDCT (TRIUMF Database Configuration Tool)

Figure 2: EPICS Components.

These tools have been relied upon in roll-out of the e-Linac control system, and have been augmented in a few ways. A web-based application now generates DHCP server configuration files for all EPICS IOCs that use TCP/IP communications with the devices that they control. This allows easy configuration and rollout of field bus oriented networks. EPICS synoptic screens for beam optics and diagnostics devices are now produced by scripts, and configured through the web application and related database. This simplifies on step of producing these screens, which are dense in content, and have many details that are difficult and time consuming to produce interactively.The e-Linac Control System uses the EPICS components listed in Figure 2.

CONCLUSION

Interfacing with turnkey systems continues to be difficult in some cases. Extremely careful and detailed planning of all aspects of the vendor interface is required for a seamless integration. Much more time is required to perform integration work than is required for basic rollout of an EPICS + PLC slow controls system.

Consistent use of methodologies and tools speed deployment and produces a more consistent control system product. Building tools to produce control system software components takes up-front effort, but pays off quickly through efficiency gains and product quality improvement.

ACKNOWLEDGMENT

The author acknowledges all EPICS collaborators, who are too numerous to mention. Special recognition is given to Mark Rivers for his continued support of the asynDriver[9] package as well as Dirk Zimoch for his development and support of the streamDevice[10] package. Without these two packages, the e-Linac Control System at TRIUMF could not have succeeded. The author acknowledges Jason Abernathy (MSc) of the University of Victoria for his contribution of the viewscreen development.

REFERENCES

[1] Schneider Electric Canada website: http://www.schneider-electric.com/site/home/index.cfm/ca/
[2] Air Liquide Advance Technologies website: http://www.airliquideadvancedtechnologies.com/en/welcome.html
[3] Busch Vacuum Pumps and Systems Canada website: http://www.busch.ca
[4] BiRa Systems Inc. website: http://www.bira.com/products/bigen/epsc.php
[5] Galil Motion Control website: http://www.galilmc.com/
[6] Ampegon Transmission Systems website: http://www.ampegon.com/products/transmission-systems
[7] Siemens PLC website: http://w3.siemens.com/mcms/programmable-logic-controller/en/pages/default.aspx
[8] J. Richards, R.Nussbaumer, "ISAC EPICS on Linux: The March of the Penguins",ICALEPCS 2011, Grenoble, France
[9] EPICS asynDriver website: http://www.aps.anl.gov/epics/modules/soft/asyn/
[10] EPICS streamDevice website: http://epics.web.psi.ch/software/streamdevice/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

LANSCE CONTROL SYSTEM UPGRADE STATUS AND CHALLENGES*

Martin Pieck#, Dolores Baros, Eric Bjorklund, John Faucett, Jack Gioia, Jeff Hill, Pilar Marroquin, Jerome Paul, James Sedillo, Fred E. Shelley Jr., Heath Watkins
Los Alamos National Laboratory, Los Alamos, NM 87544, USA

Abstract

The Los Alamos Neutron Science Center (LANSCE) linear accelerator drives five user facilities: Isotope Production, Proton Radiography, Ultra-Cold Neutrons, Weapons Neutron Research, and Neutron Scattering. In 2011, we started an ambitious project to refurbish key elements of the LANSCE accelerator that have become obsolete or were near end-of-life. The control system went through an upgrade process that affected different areas of LANSCE. Many improvements have been made but funding challenges and LANSCE operational commitments have delayed project deliverables. In this paper, we will discuss our upgrade choices, what we have accomplished so far, what we have learned about upgrading the existing control system and what challenges we still face.

INTRODUCTION

The LANSCE accelerator looks back at almost 45 years of operations. It was one of the first accelerators to use computer technologies to control and monitor its beam line components. It started out with a custom in-house design called RICE (Remote Instrumentation and Control Equipment) which was installed in the early 1970's when the facility was built. Since then, the facility has seen partial upgrades and extensions utilizing CAMAC, VME, and PLCs while introducing EPICS (Experimental Physics and Industrial Control System) in the 1990's as a supervisory software control application.

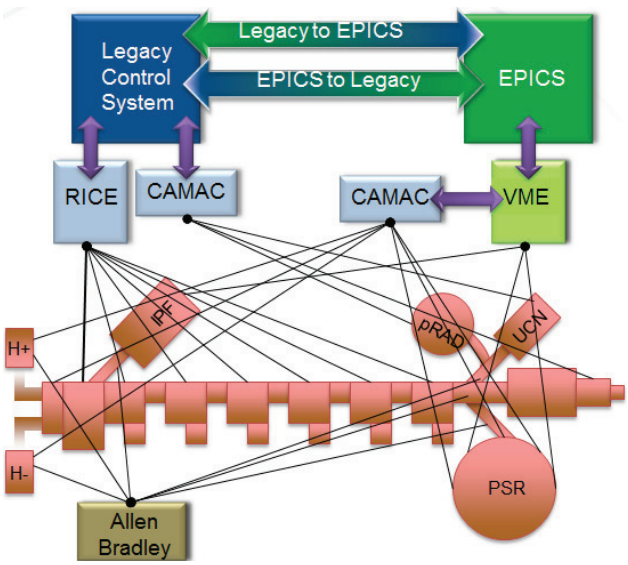


Figure 1: LANSCE Control System 1970-2010.

In 2011, LANSCE started a nine year rolling upgrade project which eventually will result in the complete replacement of the low-level RF system, the timing system, the timed data system, industrial control system, the beam-synchronized data acquisition system (including Beam Phase & Position Monitors (BPPM)), the fast protect reporting system and, wire scanner diagnostic equipment [1].

This upgrade project is primarily focused on ensuring reliable beam operations for a viable user program at the five experimental facilities. It will also benefit Los Alamos National Laboratory's future signature science facility called Matter-Radiation Interactions in Extremes (MaRIE) since, at its core, the 42-keV XFEL will be coupled with the existing LANSCE accelerator [2].

UPGRADE SCOPE

The monumental challenge of upgrading the control system is focused around the need to replace our VAX-based legacy control system which goes hand-in-hand with our RICE system. The VAX system has reached end of-life and the RICE systems, while a novel invention when it was designed in the late 1960's, is getting harder to maintain, and lacks the flexibility and performance of a modern distributed system with processing power near the front end.

RICE is a star configured control and data acquisition system that supports industrial control and beam-synchronized type of data acquisition. At its heart, the RICE Interface Unit can issue a parallel RICE module read request that provides a transverse snapshot of the accelerator. This implementation resembles the functionality of a timing system which provides trigger gates to distributed data acquisition equipment.

Given the complex functionality, replacing the RICE system is not an easy task and one starts to appreciate the engineer's ingenuity to design such a system about half a century ago.

Industrial Control

The first part of our RICE upgrade project addresses the Industrial Controls (slow control). We chose a Programmable Automated Controller (PAC) built by National Instruments (NI). The controller is called CompactRIO (cRIO) and is a reconfigurable control and data acquisition system which is supported by EPICS [3]. The NI cRIO–9024 embedded real-time controller features an industrial 800 MHz processor and contains, 512 MB DDR2 RAM, and 4GB of non-volatile storage. The removable cRIO controller sits in an NI cRIO-9118

*Work supported by LANL for the U.S. Department of Energy under contract W-7405-ENG-36; #pieck@lanl.gov ; LA-UR-15-27880

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

chassis with eight accessory slots, and a user programmable Xilinx Virtex-5 LX110 Field Programmable Gate Array (FPGA). The most noticeable advantages are reliability, high performance and hot swappable modules and a high performance FPGA. For our application, the cRIO is hosted in a commercial BiRIO rack-mount chassis which interfaces to commercial BiRIO interface boards that in turn provide interfaces to the device field wiring.

Figure 2: cRIO in BIRO chassis.

National Instruments provides a wide variety of modules that cover most of our needs. Beyond that third-party products complement the NI module product line. If that still doesn't meet the needs, a Module Development Kit (MDK) allows the development of custom modules to meet the unique needs of particular products and applications.

The cRIO is a fast and flexible solution at a reasonable price point. The typical response time is in the µsec range vs. msec in Programmable Logic Controllers (PLC). In addition, the cRIO provides more programming flexibility than a PLC. The cRIO FPGA and Real Time Controller (RTC) running VxWorks, are programmed in LabVIEW. The RTC embedded EPICS IOC (with a complete environment - all record types and fields are visible and available) makes this solution a standalone EPICS IOC which communicates with the RTC LabVIEW Real Time code via a shared library. Another advantage is that the independently running FPGA, interfacing directly with the cRIO I/O modules keeps running even though the EPICS IOC is rebooted. This can be helpful if interlock functionality needs to continue even when the IOC has to undergo maintenance. Last, but not least, the whole system maximizes its flexibility through the freedom on how an application is partitioned between the FPGA, LabVIEW RTC, and EPICS database code [4].

Beam-Synchronized Data Acquisition

The second part of the RICE system replacement effort focuses on the beam-synchronized data acquisition. The hardware solution chosen is a cPCI/VPX architecture in one crate allowing the communication between both sides via a PCI Express communication bridge. VPX is an ANSI standard (ANSI/VITA 46.0-2007) that provides VMEbus-based systems with support for switched fabrics over a new high speed connector [5].

Figure 3 provides the front view of our cPCI/VPX chassis. On the left we have three 250W, 3U high plug-in power supplies with an additional power supply expansion slot for redundancy. Located to the right of the power supplies are the six 3U VPX slots. The picture shows two test cards inserted, one of which is an Ethernet controller providing network connectivity to all VPX slots via the backplane. On the right side of the chassis are eight 6U cPCI slots.

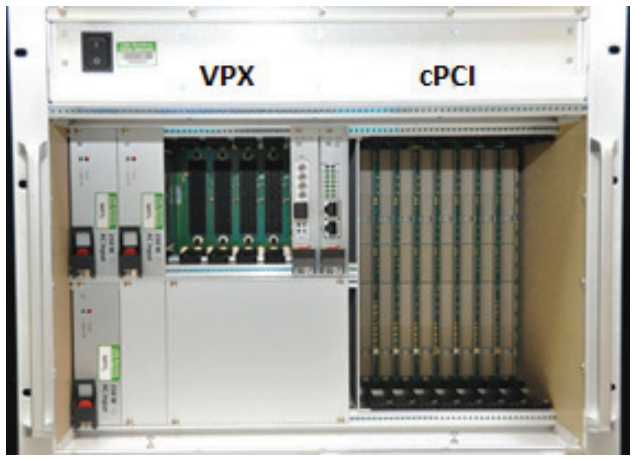


Figure 3: cPCI/VPX Chassis Front View.

This architecture will be used for the BPPM system. On the cPCI side the signals from 4-electrodes (top, left right, bottom) are captured along with the accelerator reference signal. The signals are then conditioned to 201.25 MHz RF waves and then transported to the VPX side where a high speed digitizer captures the data at 4-nanosecond time increments in order to analyse beam position and phase variations that occur throughout the 1 millisecond pulse cycle. Timing for synchronous measurements between BPPMs and beam specific information is provided by an event receiver (connected to the master timer via optical links) on the cPCI side that provides the timing and beam species information prior to submission to the EPICS database. A soft core processor, which resides in FPGA fabric, is used to store the results to the EPICS database for use by beam operators [6].

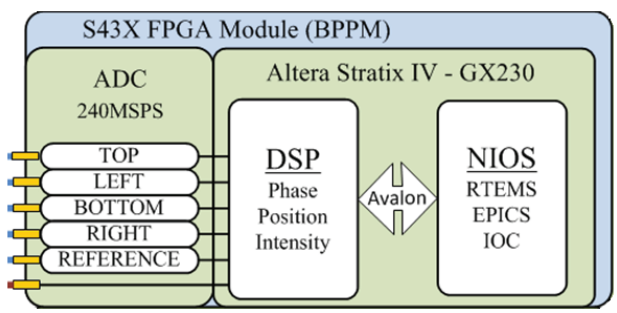


Figure 4: FMC Carrier Board with Altera Stratix IV FPGA and front end ADC.

The BPPM hardware solution on the VPX side shown in Figure 4 is very adaptable and will be used for our other beam-synchronized data acquisition application needs. It consists of a FPGA interfaced Mezzanine Card (FMC) [7], that hosts an Analog Digital Converter (ADC) and an Altera Stratix IV-GX230 [8]. The latter hosts an

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

embedded EPICS IOC running the RTEMS operating system on a NIOS-II softcore. The hardware/software solution has numerous advantages:

- Enables buying hardware COTS but still employs high performance hardware design
- FMC enables custom Digital Signal Processing (DSP) solutions in hardware without custom board design
- Ensures an independent and incremental upgrade path for AFE (ADC), DSP, or IOC
- Integrates ADC, digitizer module, signal processing EPICS IOC all into one board
- Highly distributed, therefore increased fault tolerance

Timing System

Essential to beam-synchronized beam operations is the timing system which undergoes a change from a centralized gate generating system to a distributed event-based system where timing gates are generated locally. The new Timing Pattern Generator (TPG) is a dual-redundant system. Each of the redundant TPG's has a VME-64x crate, a MVME-6100 processor, a set of Micro Research Finland (MRF) event-generator modules, and an AC zero-crossing detector and beam-enable logic module (implemented in a cRIO system). The cRIO FPGA-based beam enable logic has been used to implement specific features, such as enabling or disabling a beam from the operator consoles, single-shot mode, single-burst mode, continuous-burst mode, burst of bursts mode, and cycle stealing. The new TPG has performed successfully during the 2014/15 run cycle interfacing to the old (RICE) and new timing distribution system (fibre optics to local Event Receiver). While the TPG work is complete the distributed event-based generating devices (Timing IOCs), some of which are tightly coupled with our Low Level RF system, still need to be installed during future maintenance periods [1].

Wire Scanner

The cRIO platform is also used to upgrade our aging Wire Scanner control and data acquisition system that has been using RICE and CAMAC (Computer Automated Measurement And Control) technology. Beyond the cRIO Industrial Control configuration, the Wire Scanner cRIO chassis has an NI TPC-2206 Touch Panel Computer integrated into the chassis front panel to provide real-time visual display of wire-scanner operation and direct manual control of the wire-scanner actuator [9].

Furthermore, the project took advantage of the cRIO National Instruments MDK to design a specific, commercially unavailable, cRIO AFE module. The design has been built by and is available through National Instrument's Alliance Partner BiRIO. This Analog Front End (AFE) module is a dual channel, transimpedance amplifier with dual summed inputs and true DC coupling to collect the charge signals from the sense wires. It is designed to accommodate comparatively long macro pulses (>1ms) with high repetition rates (>120Hz) without the need to provide integrator reset signals. The basic AFE bandwidth is flat from true DC to 35 kHz with a well-defined first-order pole at 35 kHz. Numeric integration is utilized in the cRIO FPGA to provide pulse-to-pulse numeric integration of the AFE signal to compute the total charge collected in each macro pulse [10].

Other Controls and Diagnostics Equipment

Beyond what has been described so far we have other systems that need to be upgraded to allow us to retire RICE and the associated Legacy Control System. Among others, this includes the Harp and Emittance diagnostic equipment mostly located along our linac.

For the Harp, a cRIO based system is in production at our 1L target. This design will be leveraged to replace the remaining Harp systems. The current design has three AFE boards and one integrity board outside the cRIO but housed within a BiRIO chassis. Logic signals for beam synchronization and AFE control connect to the cRIO through a single NI-9401TTL logic module. Signals generated by the AFEs are digitized by three NI-9220 ADC modules. Finally, one NI-9475, 60V-source module is incorporated for integrity pulse generation that used is to check continuity of the Harp wires [11].

For the Emittance stations we are still working on our design but we are confident that the cRIO system will meet the requirements and become the platform of choice.

Another system in need of replacement is our CAMAC system that provides timing and data acquisition solutions. For the most part we have been very successful and found cRIO modules with similar functionality of the CAMAC module that needed to be replaced. Other CAMAC module functionalities are not that easily replaceable and we are diligently working on a path forward.

UPGRADE STATUS & CHALLENGES

At the beginning of the project we focused on engineering solutions that would minimize the number of hardware platforms we would need to introduce and still meet all our technical requirements. We believe that our cRIO and cPCI/VPX based platforms meet that objective. The controls upgrade project is only one part of a larger investment to upgrade the LANSCE accelerator. Funding levels have been flat for the project and did not consider the year to year funding needs to execute the project in the most effective fashion. This, combined with some unexpected funding adjustments for other non-controls project scope elements, added a great deal of uncertainty to whether we would be able to do everything we had planned for.

Therefore, after the design phase, we chose to focus the majority of our controls scope elements on the purchase of the equipment vs. purchase & installation. Hoping that if we have all the hardware in hand, we will find the funding to install it either through remaining project funds, one-time funding, and/or through our maintenance budget.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Table 1: Control System Upgrade Project by Numbers; (#) Number of WS Ctrl. & Data Acq. Chassis only.

	RICE to Industrial Controls	RICE to Synchronized Data Acq.	BPPMs	Wire Scanner	CAMAC to cRIO	HARP	Emittance
Total Number of Systems to be upgraded	66	66	34	53	38	32	3
Upgrades Installed / In Production	9	0	0	4	28	1	0
Equipment on Hand but Not Installed	27	31	7	8 (40)	2	0	1
Remaining Systems to be Purchased	30	35	27	41 (9)	8	31	2

This, however, has led to some unintended consequences. Not all of the control subsystems and/or related non-control subsystems designs are at the same maturity level. For example, while the majority of the wire scanner controls equipment (Table 1: #s in brackets) is ready for installation, the associated actuator has not been funded at the level that would allow us to install a completely new system. This in turn, made us explore the possibility of retrofitting our new Wire Scanner controls chassis to the old actuator still in the beam line. Furthermore, our RICE to cRIO (Industrial Control) upgrade has progressed with nine units installed while we don't have a RICE to cPCI/VPX (Synch. Data Acq.) unit in production. This is inefficient since we will need to get back to those systems that have been upgraded to the new cRIO subsystem in order to install the cPCI/VPX subsystem in order to retire a whole RICE unit. So far we still have all of our RICE units in production. However, 9 of them only run the beam-synchronized data acquisition.

Table 1 shows a numerical overview of our controls subsystems that need to be upgraded (row 1); subsystems hardware that have been upgraded (row 2); hardware that has been purchased for a pending upgrade (row 3); and subsystem hardware that still needs to be purchased such that a system can be upgraded (row 4).

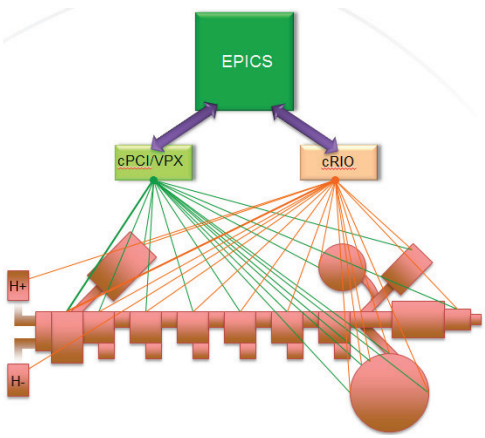


Figure 5: LANSCE Control System 2020.

Our progress has been hampered by the funding limitations and uncertainty. Furthermore, all installation work needs to be done during our 4 month maintenance period on top of all other maintenance that needs to be done without additional labor resources. Our ultimate goal is shown in Figure 5 (compared to Figure 1) with EPICS as a supervisory control system.

CONCLUSION

The cRIO and cPCI/VPX are flexible hardware and software solutions and therefore adaptable to a wide range of control and data acquisition problems. With most of the LANSCE Control System design work complete the project has transitioned to purchasing and installation. Significant work still lies ahead to complete this monumental control system upgrade task by 2020.

REFERENCES

[1] Bjorklund E., "Replacing the Engine in your car while you are still driving it", THHC2O03, these proceedings ICALEPCS 2015; Melbourne, Australia
[2] Pieck M., et al., "MaRIE – Instrumentation & Control System Design Status and Options", MOPGF162, these proceedings ICALEPCS 2015; Melbourne, Australia
[3] E. Björklund, A. Veeramani and T. Debelle, "Using EPICS Enabled Industrial Hardware for Upgrading Control Systems," ICALEPCS'09, Kobe, October 2009, WEP078, p.555(2009); www.JACoW.org
[4] Truchard J., et al.., "COTS Technology for High Energy Physics Instrumentation," BIW10, Santa Fe, New Mexico, USA (2010).
[5] https://en.wikipedia.org/wiki/VPX
[6] Watkins H., et al.., "Development of a High Speed Beam Position and Phase Monitoring System for the LANSCE LINAC,"WEPD09, IBIC2014, Monterey, California, USA (2014).
[7] http://dev.bira.com/products/module_datasheet.pdf
[8] Bittware, "S4-3U-VPX Commercial & Rugged Altera Stratix IV GX 3U VPX Board with Optional VITA 57 FMC for I/O" S43X datasheet, May 2014.
[9] Gilpatrick J.D., et al.., "WireScanner Beam, Profile Measurement for the LANSCE Facility," MOPPR081, IPAC'12, New Orleans, Louisiana, USA (2012).
[10] Gruchalla M., et al.., "LANSCE Wire Scanner AFE Analysis Design and Fabrication," TUPSM016, BIW10, Santa Fe, New Mexico, USA (2010).
[11] Sedillo J., et al.., "LANSCE 1L Harp Data Acquisition System Upgrade," TUPD11, IBIC2014, Monterey, California, USA (2014).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

MaRIE – INSTRUMENTATION & CONTROL SYSTEM DESIGN STATUS AND OPTIONS*

Martin Pieck#, Robert W. Garnett, Brian G. Smith, Fred E. Shelley Jr.,
Los Alamos National Laboratory, Los Alamos, NM 87544, USA

Abstract

Los Alamos National Laboratory has defined a new signature science facility, Matter-Radiation Interactions in Extremes (MaRIE) that builds on the existing capabilities of the Los Alamos Neutron Science Center (LANSCE). It will be the first multi-probe materials research center to combine high-energy, high-repetition-rate, coherent x-rays with electron and proton-beam charged-particle imaging to perform in-situ measurements of a sample in extreme environments. At its core, a 42-keV XFEL will be coupled with the LANSCE MW proton accelerator. A pre-conceptual design for MaRIE has been established. Technical risk reduction for the project includes an injector test-stand that is currently being designed. New accelerators are either planned, under construction, or currently in operation around the world, providing opportunities for the MaRIE project to leverage the instrumentation & controls (I&C) efforts of these facilities to minimize non-recurring engineering costs. This paper discusses possible MaRIE I&C system implementation choices and trade-offs, and also provides an overview of the proposed MaRIE facilities and the current design.

BACKGROUND

X-ray imaging is unique, both because of the penetrating power of x-rays in solid matter - as Wilhelm Rontgen discovered in 1895 - and because x-ray wavelengths are short enough to resolve the interatomic spacing in matter via diffraction - Max von Laue's discovery in 1912. Those properties allow scientists to push forward fundamental physical sciences and to find major applications in structural imaging, from new commercial drugs to jet turbine blades [1].

Los Alamos National Laboratory's proposed Matter-Radiation Interactions in Extremes (MaRIE) experimental facility is slated to introduce the world's highest energy hard x-ray free electron laser (XFEL). The MaRIE 42-keV XFEL, with bursts of x-ray pulses at up to gigahertz repetition rates for studying fast dynamic processes, will help accelerate discovery and design of the advanced materials needed to meet 21st-century challenges [2].

MaRIE FACILITY

The MaRIE facility will include a 12-GeV linac to provide a suite of measurements designed to investigate the performance limits of materials in extreme environments. One of MaRIE's most powerful tools will be the ability to multiplex an x-ray FEL, electron, and proton radiography onto a target material to study dynamic events as they develop. The existing LANSCE proton linac will be used to provide proton radiography (pRad) [3]. The MaRIE electron linac will be built in a new tunnel north (right side in the figure) of the existing LANSCE proton linac tunnel as shown in Fig. 1 [4].

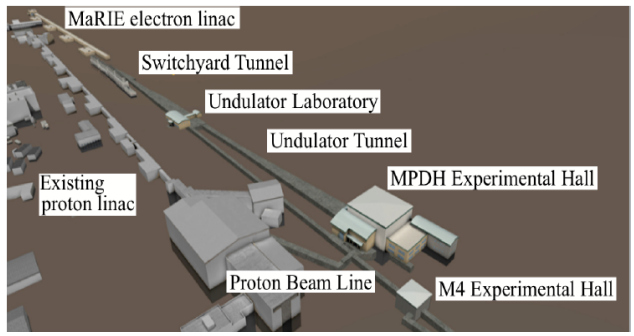


Figure 1: MaRIE Facility Layout.

OTHER XFELS

Besides the planned MaRIE XFEL facility in Los Alamos, New Mexico, USA, next-generation light sources also exist in Europe, Japan and elsewhere in the USA. X-ray facilities are being constructed at LCLS in California, SACLA in Japan, the European XFEL in Germany and the SwissFEL. The operating principles of these facilities are very similar. Electrons are first accelerated to high energies and then made to generate high-intensity x-ray laser light. LCLS and SACLA rely on conventional accelerator technologies. The European XFEL will use superconducting technology [5].

MaRIE BEAM REQUIREMENTS

The MaRIE electron beams consist of micro pulses for an XFEL undulator and micro pulses for electron radiography (eRad). A special feature of the MaRIE facility is the ability to provide unevenly spaced XFEL and eRad micro pulses distributed over a macro pulse of up to 100 µs. The macro pulse repetition rate is 60 Hz. Each XFEL micro pulse includes up to 0.2 nC of charge. Each 100-µs-long macro pulse can include up to 30 XFEL micro pulses. Each eRad micro pulse includes up to 2 nC of charge. Each 100 µs long macro pulse can include up to 10 eRad micro pulses. The spacing between micro pulses is determined by the experimental needs. The minimum spacing for each eRad micro pulse is 25 ns, while the minimum spacing for each XFEL micro pulse is 2.5 ns [4].

*Work supported by LANL for the U.S. Department of Energy under contract W-7405-ENG-36; #pieck@lanl.gov; LA-UR-15-27877

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

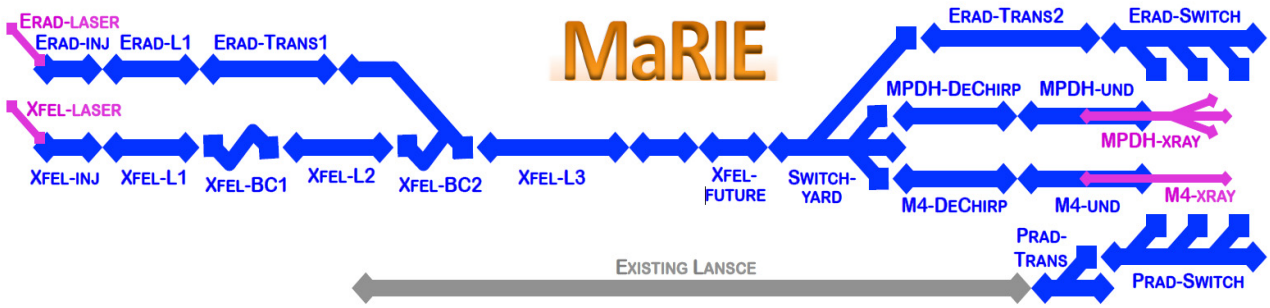


Figure 2: MaRIE Beamline Layout [6].

MaRIE BEAMLINE LAYOUT

The MaRIE XFEL and eRad micro pulses are produced and accelerated by separate injectors and initial injector linac sections. Both injector linac sections include an injector and L1 linac section. The XFEL side includes two bunch compressors and a short L2 linac section. The outputs of these parallel beamlines feed the L3 main linac as shown in Fig. 2. A switchyard at the end of the L3 linac splits the XFEL and eRad beams off to go through undulators or directly to the target [4].

Photo Injector Region

Current plans suggest a 1.3 GHz normal conducting photo injector (PI) for long pulse (100 μs) operation and a 60 MV/m gradient cathode for both the eRad and XFEL [7]. The photo injector design is similar to the PITZ design used at the Photo Injector Test facility in Zeuthen, also used at FLASH [8] and will be used for the European-XFEL [9]. Following each injector are two superconducting cryo modules (CM1 & CM2) operating at 1.3 GHz for the purpose of capturing the beam from the PI and introducing energy slew for bunch compression BC1. Cryo modules 3 & 4 will operate at 3.9 GHz for the purpose of linearizing the beam energy slew for bunch compressor (BC1). BC1 will provide x20 compression at about 400 MeV. An overview of the injector region is given in Fig. 3 [6].

Figure 3: MaRIE Injector Region Layout.

Linac

The MaRIE linac will use proven RF cavity designs. Four hundred sixty cavities will be of the 1.3 GHz TESLA type used in the FLASH [8], LCLS-II [10] and European XFEL [9] projects. The L2 linac includes 78 of these cavities and the L1 linacs each include 11 of these cavities. Like the International Linear Collider (ILC), the 460 TESLA cavities are run at an average cavity field of 31.5MV/m [11]. The 22 third-harmonic linearizer cavities will be of the 3.9-GHz type used in the FLASH linearizer [8]. Each L1 linac includes 7 of these cavities and the L2 linac includes 8 of these cavities. The average cavity gradient is 20 MV/m [4].

Undulator

The MaRIE facility is expected to have two matching undulator beam lines each leading to a different experimental facility. An underlator line is shown in Fig. 4. The first undulator line will go to the Multi-Probe Diagnostic Hall (MPDH). Each end-station in this hall will receive simultaneously eRad, pRad and XFEL beams. The second experimental facility will be the Making, Measuring, Modelling, Materials (M4) facility with potentially 3 end-stations. The undulator lines will produce greater than 2 x 10^{10} 42-keV photons in each 33-fs pulse from a bunch length of 10μm.

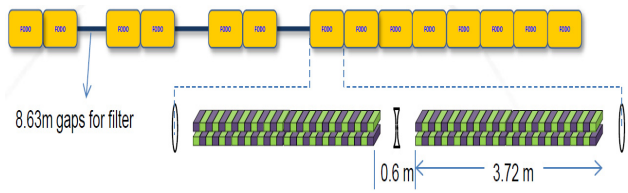


Figure 4: MaRIE Injector Region Layout.

The undulators planned for MaRIE are similar to the SwissFEL U15 undulator. Fourteen alternating focusing and defocusing quadruple lenses forming a FODO magnetic focusing lattice make up each of the undulator lines. Each 0.6 m break between undulator segments will house two gate valves, a beam position monitor, a phase shifter, a permanent magnet FODO quad and an adjustable alignment quad [12].

ACCELERATOR CONTROL SYSTEMS

Requirements

The demands on modern accelerator control systems are increasing from year to year. Today's operator and operations physicist require more and more information in real-time in order to minimize start-up time, maintain optimal beam parameters, predictively intervene to minimize beam down time and to recover quickly from off-normal events.

Given that an accelerator facility lifecycle consists of several different stages, the control system must be modular, scalable, and incrementally upgradeable.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Expansion of the control system to accommodate the installation of the accelerator and beamlines from early testing, through commissioning, and during the life of the facility should not impact the control system performance.

Today's Accelerator Control Systems

Today's accelerator control systems are a combination of a wide variety of commercial/industrial and custom made hardware and software solutions. Early accelerator controls deployed in-house custom solution as this was often the only choice. Over the years this has dramatically changed. Today more and more industrial and commercial based control system solutions are available and have become the first choice for control system engineers. The spectrum of these industrial and commercial products reach from uniquely dedicated hardware products with software configurable solutions for standard implementations to custom software solutions in highly adaptable hardware.

On a related note, over the past decade accelerator control system integration with facility related systems has become more common. Control systems for cryogenics, tunnel cooling and ventilation, etc. monitoring are needed and should be well-integrated because of their tight coupling to accelerator operation. It only recently has been felt there was a need for integrating the water distribution, building heating, etc. as well.

MaRIE CONTROL SYSTEMS

The scope of the MaRIE project likely requires a multilab collaboration with other facilities that have expertise in many of the systems required for MaRIE. Similarities between MaRIE and other XFEL facilities have been pointed out in this text. The expectation is that some of these systems could be delivered as turnkey subsystems (i.e. Cryo Plant System, Cryo Module Systems, High Power and/or Low level Radio Frequency System, Undulator System …) and later integrated with the overall facility control system. However, past experience has also shown that some of the subsystems and associated equipment were delivered by partner labs or industrial partners without their control system and were equipped in-house later with the control system product of choice. In the following we discuss the pros and cons (pros of a turnkey systems are in most cases closely tied to the cons of the in-house solution and vice versa) for a turnkey system.

Turnkey System vs In-House Solution

A turnkey system is a system that has been customized for a particular application. Turnkey systems include all the hardware and software, installation, start-up assistance, training, etc. for a subsystem with little or no involvement of the host facility (in this case MaRIE) until the system is accepted.

The pros of using a turnkey include:

- One responsible supplier will provide all the project management and become the single interface to the host facility. This frees the host facility from dealing with many individual contractors to achieve the same result.
- Suppliers most likely have already developed a control system solution for a particular subsystem which could save design / engineering costs.
- Turnkey system providers (subject matter experts) often have a better understanding of what is required to make a system work which in turn increases the cost certainty for the project.
- When working with one responsible authority, one would expect to have one warranty to secure the quality and craftsmanship of the subsystems to be delivered.

The cons of using a turnkey system include:

- Having a single responsible supplier usually means a higher management fee for this type of service which could be equivalent to hiring independent consultant(s) or permanent staff.
- Most likely the turnkey system needs to be integrated into one holistic control system which may require extensive integration work.
- In-house personal likely need extensive training due to the lack of being involved during the engineering design phase.
- Higher maintenance cost due to the possible wide variety of hardware and software solutions used across the host facility for different turnkey systems.
- Timely response to pending problems may be difficult due to lack of on-site subject matter experts.
- Reduced opportunity to develop in-house capabilities and knowledge base that could be beneficial for future projects at the host site.

Systems Engineering

A project of the scale of MaRIE will likely take delivery of many turnkey systems. To successfully integrate these systems a strong systems-engineering approach is needed as early as the beginning of the project.

A model should be developed to investigate and determine the required data flow, direction, performance, and need of synchronization between the individual systems.

Prime candidates for a turnkey system may be those that can operate in a stand-alone fashion or those that have limited interaction with the rest of the control system.

Before deciding to work with a turnkey system provider it is necessary to evaluate the technical expertise

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

available within our organization. For instance, if we have resources that could do the work, it may be better to take advantage of these in-house resources rather than outsource the work to a turnkey provider.

However, once a decision is made in favor of a turnkey system several things should be kept in mind in order to avoid any difficulties down the road.

- Be mindful about changing interface requirements in the future as the facility goes through its lifecycle stages.
- Insist on having access to all system documentation and software. Proprietary implementation may lead to the inability to make required changes in the future.
- Require the use of industry standards whenever practical which will make upgrading and interfacing easier in the future.
- Like the rest of the control system, turnkey systems should be designed with a modular upgrade path in mind.
- Test early, test often. Take advantage of prototypes, simulators, emulators, and any other way to let everyone involved get an early look at the system. Make sure tests prove that the supplier satisfies the requirements.

Integration

Given the range of demands (controls, data acquisition, data management, data analysis, …) on information technology in an accelerator facility one solution may not fit all needs. Coupled with individual preferences of the controls engineers it is more than likely that extensive integration work will need to be done. Given these facts integration should be planed for accordingly.

MaRIE PROJECT FUTURE

Over the last several years the MaRIE project has gained more and more momentum with the likelihood of gaining official US Department of Energy (DOE) project status soon. The first critical project milestone to be met will be approval of the project mission need, Critical Decision Zero (CD-0). In anticipation of an approved CD-0, the MARIE project team has developed a project schedule and cost estimate based on the facility layout presented in this paper. Furthermore, independent reviews have validated our scientific and cost/schedule bases.

To further reduce project risk a detailed technology maturation plan has been developed for MaRIE that includes an injector test stand based on an existing Advanced Free Electron Laser test facility at the LANSCE user facility.

CONCLUSION

MaRIE is a promising project that will require a multilab collaboration. In this environment, the control system's ability to be upgradable, scalable, extendable, and allow the integration of many turnkey systems will be essential to the success of the project. Implementing turnkey solutions is not necessarily always the best choice for a project like MaRIE. To determine whether a turnkey solution is a viable option, there are a number of important considerations to take into account which have been discussed in this text.

REFERENCES

[1] P. H. Bucksbaum, and N. Berrah, "Brighter and Faster" Physics Today, July 2015, p. 26 (2015); http://www.physicstoday.org

[2] C. W. Barns, "Roadmap to MARIE March 2015", LA-UR-15-22270, http://www.lanl.gov/science-innovation/science-facilities/marie/_assets/docs/roadmap-march-2015.pdf

[3] K. E. Kippen, et al., "AOT & LANSCE Focus: Proton Radiography Facility", LA-UR-13-24376, http://www.osti.gov/scitech/biblio/1083846, 2013.

[4] J. Bradley, et al., "An overview of the MaRIE X-FEL and Electron Radiography LINAC RF System", https://jacowfs.jlab.org/conf/y15/ipac15/prepress/WEPWI001.PDF, 2015.

[5] No Author, "The European XFEL in international comparison", (2015) http://www.xfel.eu/overview/in_comparison/.

[6] K. Bishofberger, "MaRIE Beamline Layout", (2015)

[7] S. Russel, et al., "MaRIE XFEL Pre-Conceptual Reference Design Injector", LA-UR-15-21963, http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-15-21963

[8] http://flash.desy.de/.

[9] http://www.xfel.eu/.

[10] J. N. Galayda, "LCLS-II Project", in Proc. IPAC'14, Dresden, Germany, 2014.

[11] ILC Global Design Effort, ILC Technical Design Report, ISBN 978-3-935702-77-5 (2013), http://www.linearcollider.org/ILC/Publications/Technical-Design-Report.

[12] D. Nguyen, et al., "MaRIE Undulator & XFEL Systems", LA-UR-15-22102, http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-15-22102.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

STATUS OF THE LOCAL MONITOR AND CONTROL SYSTEM OF SKA DISHES

S. Riggi*, U. Becciani, A. Costa, A. Ingallinera, F. Schillirò, C. Trigilio, INAF-OACT, Catania, Italy
G. Nicotra, C. Nocita, INAF-IRA, Bologna, Italy
V. Baldini, R. Cirami, A. Marassi, INAF-OATS, Trieste, Italy

Abstract

The Square Kilometer Array (SKA) project aims at building the world's largest radio observatory to observe the radio sky with unprecedented sensitivity and collecting area. In the SKA1 phase of the project, two dish arrays are to be built, one in South Africa (SKA1-Mid) and the other in Western Australia (SKA1-Survey). Each antenna will be provided with a local monitor and control system, enabling remote operations to engineers and to the Telescope Manager system. In this paper we present the current status of the software system being designed to monitor and control the dish subsystem. An overview of the dish instrumentation is reported, along with details concerning the software architecture, functional interfaces, prototyping and the evaluated technologies.

THE SKA DISH ARRAY

The Square Kilometer Array (SKA) project [1] is being designed to carry out radio observations of the sky in the wide frequency range from 50 MHz to 20 GHz with high sensitivity and collecting area to throw light in key areas of astrophysics and cosmology [2].

To this aim three different arrays were planned to be built for SKA Phase I and deployed at two different sites: the SKA1-Mid array at the South Africa's Karoo region hosting 190 dish antennas incorporating the Meerkat precursor telescopes and observing in the 0.35-13.8 GHz domain, the SKA1-Survey array at the Western Australia's Murchison region (MRO) site hosting 60 dishes equipped with phased array feeds (PAFs), incorporating the ASKAP precursor telescopes and observing between 0.65 and 1.67 GHz, and the SKA1-Low at the MRO site hosting 250000 low-frequency antennas.

The SKA design foreseen for Phase 1 was recently updated after a re-baselining process [3] carried out by the SKA Office to constrain the project costs to the established cost ceiling. The major outcomes of such review impacting dish array were a reduction in the SKA1-Mid array to 70% of its original size (190 dishes), equipped with prioritized band 2 (0.95-1.76 GHz), 5 (4.6-13.8 GHz) and 1 (0.35-1.05 GHz) and the deferral of SKA1-Survey array. Antennas (64) from the Meerkat precursor will be integrated in the SKA1-Mid array. The achieved array layout will have a central core component and additional spiral components targeted to deliver baseline lengths of 150 km. In the rest of the paper we will focus on SKA1-Mid antennas only.

* sriggi@oact.inaf.it

DISH INSTRUMENTATION

The general design for a SKA1-Mid antenna, shown schematically in Fig. 1, currently foresees a 15-m offset Gregorian dish with a feed-down configuration equipped with wide-band single pixel feeds (SPFs) for Band 1-2-5. The feed packages are mounted on a fan indexer at the focal position, allowing for changing among the available frequency bands. Only one band will be available for observation at any given time. A RFI-shielded cabinet is present in the antenna pedestal to house digital electronics and hardware for antenna movement and monitoring and control purposes, including the computing equipment supporting the Local Monitoring and Control system. No active cooling will be available for equipments inside the cabinet, only ventilation. Four sub-elements are identified in the SKA-Mid1 dish element: the Dish Structure (DS), the Single Pixel Feed (SPF), the Receiver (Rx) and the Local Monitoring and Control (LMC). Additional details on the first three are reported in the following subsections, while the LMC design is discussed in final sections.

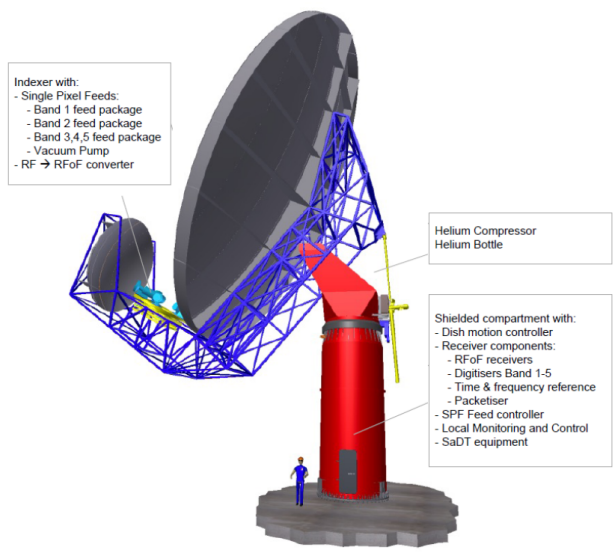


Figure 1: Schematic overview of SKA Dish design and instrumentation.

Dish Structure

The Dish Structure (DS) sub-element is responsible for the design of antenna, including mechanical structure and support, optics, reflectors, indexer as well as for the power distribution to be supplied to all sub-elements and safety systems (i.e. fire/intrusion sensors, limit switches, ...). DS

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

will supply the servo systems for antenna positioning in both azimuth and elevation comprising servomotors and encoders for precise position reading, connected to the Antenna Control Unit (ACU), placed inside the pedestal compartment. Three different antenna prototypes have been built within the DS consortium and tested against performance requirements derived from scientific goals. A down-selection among the antenna candidates is expected within few months.

Single-Pixel Feed

The Single Pixel Feed (SPF) sub-element will provide the feed packages of the antenna that receives the astronomical radio signals and relative control equipment. Each feed package will include orthomode transducers (OMTs) and low noise amplifiers (LNAs), prototyped for the SKA SPF bands, along with a Gifford McMahon (GM) cryogenic cooler to cool the LNAs, a cryostat assembly sharing a common helium supply system with a single helium compressor located at the antenna yoke, with helium supply lines routed to the indexer. A rotary vane vacuum pump is located on the indexer with vacuum lines to all connected cryostats. A single controller placed in the pedestal interfaces with the SPF feed packages, helium and vacuum systems using a low speed serial protocol over fibre, and allows external monitoring and control operations performed by LMC through a controller application software implemented on an ARM architecture.

Receiver

The Receiver (Rx) sub-element provides the hardware equipment to digitize the RF signal (e.g. the science data) received from each SPF band after a RF-to-optical conversion performed at the indexer. Digitization occurs inside the pedestal enclosure. A number of components are present. The digitiser packetizes and transmits the signal over a high-speed Ethernet link to the central signal processor. A Master Clock timer unit receives time and frequency reference inputs externally (from SKA SaDT element) and generates timing and frequency references where needed, including the control of the calibration noise source that is transmitted over fibre to the feed packages. As for SPF, a central controller is present and acts as a single point of monitoring and control to the LMC sub-element.

THE DISH LMC SYSTEM

The SKA M&C Hierarchy

The monitoring and control of the SKA constitutes a big challenge given the large and heterogeneous number of instrumentation to be remotely managed, including the integration of SKA Meerkat and ASKAP precursors. The overall number of monitoring points is roughly estimated of the order of 10^5 for SKA Phase 1 and larger than 10^6 for SKA2 [4], each antenna contributing with few hundreds parameters for SKA1-Mid and few thousands for SKA1-Survey antennas and PAF receivers. In this context the monitoring and control data flow provided by each SKA dish is expected to be of

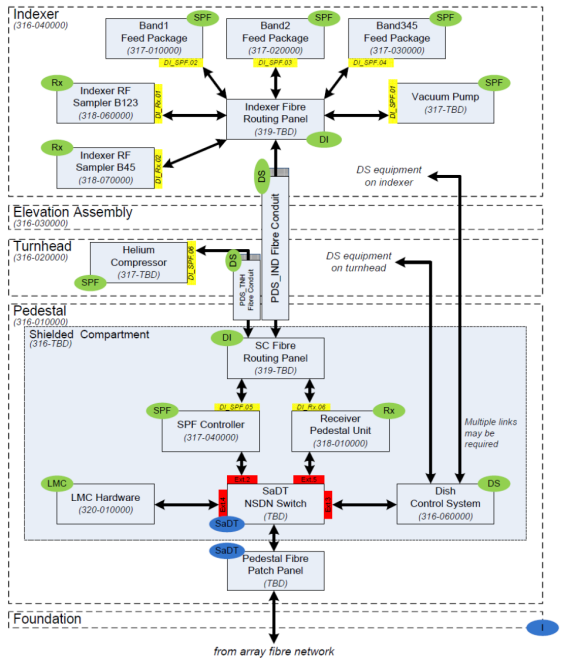


Figure 2: Schema of monitoring and control interfaces among SKA dish sub-elements.

the order of 100-200 kbps for SKA1-Mid and SKA1-Survey antennas and 1 Gbps for SKA1-Survey receiver.

To deal with complexity and ease system scalability, a hierarchical architecture of the M&C system was considered. The Telescope Manager (TM) element at the top of the hierarchy orchestrates the scientific observations, centrally coordinating all the involved telescopes and providing control information to them. Further details on TM design and organization will be given at this conference [5]. In order to perform the telescope monitoring and control TM communicates with the Local Monitoring and Control (LMCs) elements of the other SKA subsystems (one per each SKA subsystem, i.e. Dish, Central Signal Processor (CSP), Science Data Processor (DSP), ...). These LMCs perform the direct control and monitoring tasks on their subsystems, fulfilling the following main responsibilities:

- maintain direct and local connectivity to controlled components, specially in case of TM downtimes
- receive and execute control commands (i.e. setup, configuration, calibration, scheduling, life-cycle, ...) from TM involving dish components
- collect and aggregate logging and monitoring information (state/status, moni parameters, ...), events and alarms from dish components and report them to TM, after a proper mapping to a SKA common model
- perform diagnostic operations, fast control (<100 ms rate) and safety actions (autonomously or when commanded) on dish components
- perform life-cycle management operations (i.e. remote software/firmware update, power up/down)

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- providing drill-down and tunnelling capabilities (i.e. remote access to devices or engineering interfaces, ...) to TM and operators

The M&C interfaces among dish sub-elements are represented in the diagram of Fig. 2. LMC performs M&C operations exclusively via the sub-element controllers, through a local Ethernet link, enabled by a network switch provided by the SKA Signal and Data Transport (SaDT) element, responsible also for the external connectivity of the antenna (e.g. transport of science and M&C data).

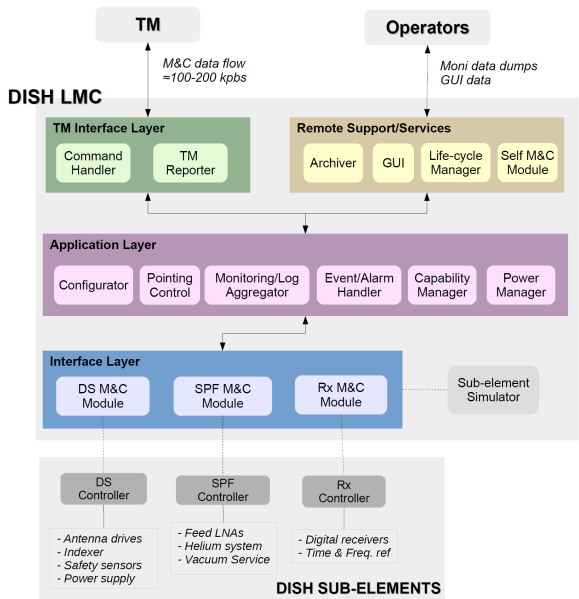


Figure 3: Logical view of Dish LMC software architecture arising from the preliminary design phase.

Preliminary Dish LMC Design

The high-level components of Dish LMC were identified during the preliminary design phase. A logical schema of the LMC architecture for SKA1-Mid antennas is reported in Fig. 3. The LMC software system can be regarded as a collection of modular packages, each providing a set of functionalities (as per SKA LMC requirements), organized in a three-level hierarchical structure. The "Interface" layer comprises a set of interface modules for the DS, SPF and Rx controllers, implementing communication access, control commands and retrieval of monitoring information. At this level a mapping from internal hardware state/status parameters to a global SKA control model (standardized for all SKA LMCs) is also performed. A suite of sub-element simulators is also planned to be developed for early testing of internal interface modules.

The "Application" layer implements built-in high-level functionalities and collective operations on sub-elements. For instance, configuration tasks, such as the configuration of the antenna for observation in a given frequency band or the setup of TM interface reporting, are performed by a *Configurator* component. Aggregation of all monitoring and log information collected from sub-elements (including pointing meta-data) and from LMC components (self M&C module) is carried out by a *Monitoring/Log Aggregator* component, which is responsible to maintain a rolled-up view of monitoring data inside the antenna and report it to TM. Similarly, events and alarms, particularly those related to safety or emergency situations (i.e. intrusion, emergency stop, ...), are handled by an *Event/Alarm Handler* component, which is responsible to execute recovery actions on failing components, filtering/reporting relevant events to TM (i.e. target lock, stow position, power-cut, ...).

Further, LMCs in SKA are expected to report a global capability information to TM, expressing the ability to deliver certain functionalities, i.e. for the dish the ability to operate the telescope in a band given the internal (health) status of all its constituents. Such functionality is provided by the *Capability Manager* component in the LMC architecture.

Pointing operations, performed by the *Pointing Control* component, requires support for receiving and executing pointing commands from TM interface, including the computing of local pointing correction to be applied before commanding the DS controller servo system.

On top of the hierarchy, a "TM Interface" layer is responsible to execute high-level commands from TM (defined in a standardized TM-LMC interface model), making use of service functionalities implemented in the middle layer. This layer was introduced since TM should not be aware of LMC internal implementation nor should access directly dish sub-element components. Moreover, it is desiderable to decouple as much as possible TM interface from LMC internal design, so that a change on the interface side does not severely affect LMC.

An archiving component, comprising a data archive plus archiving services, is also present. It is infact expected that LMC should not permanently store monitoring and log data locally. These data will be delivered to TM which will make them available globally. Only a limited-duration circular queue data archive, covering a 12 h time-window at maximum, is therefore foreseen and LMC shall provide services for creating, managing and downloading the archive.

Finally a set of high-level services are to be delivered to both LMC and engineering operators, such as a monitoring and control GUI, tunnelling services to access sub-element engineering GUIs and utilities to perform life-cycle operations (i.e. software/firmware update).

Implementation Guidelines

Object-oriented design will be extensively adopted for the implementation of the LMC control system, C++ and python being the likely adopted programming languages. LMC components are mostly network-distributed "online" components, designed to be configurable at run-time, exchanging data/events/logs each other over a software channel with different patterns, exposing commands callable by other components. "Offline" components, in turn, provide a utility or support function in the context of online components.

All the required communication and M&C features (alarm,

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

logging, archiving systems, GUI tools, error and exception handling, security, ...) will be provided by a M&C framework and its underlying communication middleware. To minimize efforts and risks LMC will therefore rely on an existing off-the-shelf M&C framework rather than implementing base functionalities from scratch.

Key Technologies

A considerable effort is being made within the SKA Telescope Manager consortium to promote a standardization of adopted technologies, best practise, conventions (i.e. common model for definition of states and modes, ...) and re-utilization of experiences and modules throughout the entire SKA control system. As a consequence the choice of a suitable COTS M&C framework, affecting the development of all SKA elements, resulted from a collaborative effort of representative members from all elements' LMCs and TM groups, as well as from the support of selected experts of the framework candidates and reviews of external accelerator/observatory facilities. A set of about 150 requirements (including performance requirements whenever available for LMCs) has been weighted against all the examined frameworks to drive the best choice for SKA needs, among which:

- Strong alignment with architectural concepts and required LMC functionalities;
- Reliability and scalability to SKA system size;
- Support for industrial standards and custom module development;
- Maturity and long-term support for both framework and middleware;
- Support for hierarchy creation among components, component configuration and command implementation;
- Availability of experiences both within LMCs and SKA precursors;
- Integration and re-use of precursors;
- User-base and strong community support

The following candidates were taken under consideration for evaluation against technical requirements: EPICS v3 [6] and v4 [7], TANGO Control System [8], Alma Common Software (ACS) [9], Meerkat CAM [10]. At last TANGO got the highest scores and was finally selected for prototyping stage after a careful evaluation of pros and cons considering also modern M&C trends (see for instance [11]).

The protocol for communication between LMC and sub-element controllers is still to be defined. A natural choice would be extending TANGO layer down to low-level, but other alternatives for the middleware and data serialization have been evaluated against LMC requirements: ØMQ [12] with MessagePack [13] or Google Protocol Buffer [14] as serializer, ZeroC ICE [15], KaTCP [16]. Sample prototype modules have been developed and benchmark tests carried out on different network links. ØMQ and ICE received higher overall rates in the scoring process. KaTCP and ØMQ were found easier to be integrated with the TANGO layer. TANGO, ICE and ZeroMQ middlewares latency scaling curves were found comparable and allow to fullfil the most stringent performance requirement. KaTCP significantly deviated from the other middlewares, particularly above few MB transferred. The reason of such discrepancy requires further investigation.

Other technologies are currently under investigation, particularly those related to GUI development and continuous integration. Outcomes of the evaluation will be reported in a future project update.

SUMMARY

The SKA project has recently entered pre-construction phase II. LMC is expected to report a detailed design review of the dish control system. This stage will include definition of commands, monitoring points, state/status for each dish sub-element, consolidation of internal and external interfaces, refinement of the architecture and downselection of technologies to be employed. Development of some of the software components is also foreseen.

ACKNOWLEDGMENT

The author thanks the System Engineering (SE) team of Dish Consortium for their efforts and support in the dish interface modelling.

REFERENCES

[1] http://www.skatelescope.org

[2] "Science with the Square Kilometre Array", New Astronomy Reviews 48 (2004) 979-1606.

[3] A. McPherson, "Report and Options for Re-Baselining OF SKA-1", SKA-TEL-SKO-0000229 (2015).

[4] Y. Wadadekar et al., "Design Concept Description", WP2-005.065.020-TD-002 (2011).

[5] L.R. Brederode et al., "SKA Telescope Manager Project Status Report", MOA3O01, *These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

[6] www.aps.anl.gov/epics

[7] http://epics-pvdata.sourceforge.net

[8] www.tango-controls.org

[9] www.eso.org/~almamgr/AlmaAcs

[10] L. van den Heever, "Meerkat Control and Monitoring - Design Concepts and Status", MOCOAAB06, ICALEPCS'13, San Francisco, CA, USA (2013).

[11] A. Götz et al, 'TANGO - CAN ZMQ REPLACE CORBA¿, TUCOCB07, ICALEPCS'13, San Francisco, CA, USA (2013).

[12] http://zeromq.org

[13] http://msgpack.org

[14] http://developers.google.com/protocol-buffers

[15] http://zeroc.com

[16] http://pythonhosted.org/katcp

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

STATUS OF THE EPICS-BASED CONTROL AND INTERLOCK SYSTEM OF THE BELLE II PXD

M. Ritzert*, Heidelberg University, Germany†

Abstract

The Belle II e+e- collider experiment at KEK (Tsukuba, Japan) will include a new pixelated detector (PXD) based on DEPFET technology, providing the two innermost layers around the beampipe. This detector requires a complex control and readout infrastructure consisting of several on-sensor ASICs and remote FPGA boards. We present the architecture and EPICS-based implementation of the control, alarm, and interlock systems and their interconnectivity to other legacy/heterogeneous subsystems. The interface to the NSM2-based Belle II run-control to orchestrate the PXD startup sequence is also presented. An installation of CSS is used to implement the user interface. The alarm system uses CSS/BEAST, and is designed to robustly minimize spurious alarms. The interlock system consists of two main parts: a hardware-based system that triggers on adverse environmental (temperature, humidity, radiation) conditions, and a software-based system. Strict monitoring including the use of heartbeats ensures permanent protection and fast reaction times. Especially the power supply system is permanently monitored for malfunctions, and all user inputs are verified before they are sent to the hardware. The control system is embedded into a larger slow-control landscape that also incorporates archiving, logging, and reporting in a uniform workflow for the ease of daily operation.

CONTROL SYSTEM

The PXD control system is built on the control framework EPICS[1], which is widely used in high energy physics systems. The Eclipse-based suite Control System Studio (CSS)[2] is being used as the operator interface framework.

The EPICS system will be deployed in a dedicated physical network segment only accessible via a gateway using the Channel Access (CA) protocol [1] for load reduction on the EPICS Input/Output Controllers (IOCs)[3] and access logging. Access control lists (ACLs) enforced within the IOCs will be used to implement various access levels (read only, user, administrator).

In order to achieve a consistent view on operating conditions during the experimental run and for analysis after a failure, a major amount of identified key values have to be recorded and archived centrally in a database. This is to ensure that in normal as well as abnormal situations a consistent and synchronized view on the data and whole system is possible.

The system will be deployed on x86_64 servers running Scientific Linux as the operating system. To allow for fast, repeatable installation and controlled updates, the required software is packaged as RPMs.

The following list details major devices of the system and their means of control via EPICS:

- The data handling engine (DHE) [2, 3] receives the global trigger and timing signals and distributes them to the sensors. The internal FPGAs connect to the outside world via the IPbus protocol. IPbus is based on UDP/IP and can be implemented purely in HDL. An IOC implementing the IPbus protocol in the DHE-specific way has been written.
- The actual DEPFET detectors are controlled and read out via a number of ASICs that are accessible via JTAG from the DHE. The IOC for the DHE and the firmware running on the DHE include code to access the JTAG chain. Actual commands are again transmitted to the DHE via IPbus.
- The power supplies communicate with the control PC via the CHROMOSOME protocol, a fault-tolerant middleware [4] that also includes a heartbeat to quickly detect a lost connection on both sides. Another custom IOC has been written to interface these devices.
- The online selector nodes (ONSEN) receive tracking information from other subdetectors via the data concentrator (DATCON) and high level trigger (HLT) and uses it to filter out events not in proximity to a projected particle crossing in the PXD. The ONSEN devices are controlled with an IOC running on the embedded Power PC in the FPGA via a memory mapped interface.
- The ATCA and µTCA crates housing the ONSEN, DHH and DATCON systems are accessible via IPMI.
- Belle II run control communicates via the NSM2 protocol developed at KEK [5, 6]. A bidirectional gateway also developed at KEK is used to interface with the EPICS world.
- The environmental conditions inside the detector volume are monitored via Bragg grated fibres read out with an optical interrogator from Micron Optics. An IOC that reads data from the device at high frequency (several 100 Hz) is under development. The data will be made available to EPICS either as a single data points down-sampled from the raw device rate, or put through an FFT transformation to obtain a frequency spectrum, which is then available as a waveform record. The latter mode is especially useful to analyze the minute movements of the detector.
- The actual slow control of the IBBelle CO_2 cooling plant is implemented in a PLC located within the plant itself. The Modbus protocol is used to monitor its

* michael.ritzert@ziti.uni-heidelberg.de
† For the DEPFET collaboration.

[1] http://www.aps.anl.gov/epics/
[2] http://controlsystemstudio.org
[3] http://www.aps.anl.gov/epics/base/R3-14/12-docs/AppDevGuide.pdf

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

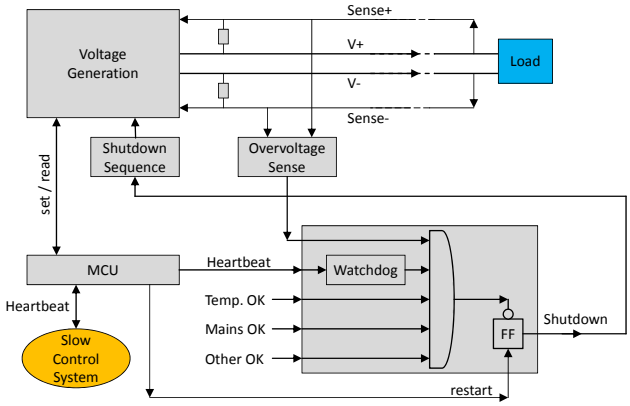


Figure 1: Protection system of the power supplies. All items in gray are inside the power supply.

operation, set the desired operating point, and provide expert access.

A common naming scheme for all PVs ensures that users can quickly decode the meaning of a PV name in any context.

ALARMS AND INTERLOCKS

The Best Ever Alarm System Toolkit (BEAST) [7] integrated into a custom build of CSS is used to visualize the alarms for the operators.

Two alarm levels, major and minor, are defined as "(part of the) detector in-operational, or immediate danger to the detector" and "take action to avoid a major alarm" respectively. A major alarm will typically cancel the current run. Many major alarms also trigger an interlock.

Several soft-interlocks have been applied in order to ensure immediate action during detected potentially dangerous conditions or malfunctions, or to prevent these situations from arising in the first place. Software interlocks are accompanied by hard-wired interlocks that are in any case the last-resort to prevent hardware damages.

Power Supplies

As the only supplier of power into the detector volume, positive control of the power supply (PS) system at all times is of utmost priority. The PS unit is controlled by an internal microcontroller (MCU). To allow for immediate, automatic reaction in case of problems, a sophisticated interlock system is implemented, as shown in Fig. 1. It can command an automatic shutdown of all voltages the power supply provides to the detector. A number of conditions are monitored to compute the shutdown signal. Besides the over-voltage protection and monitors for the mains input voltage and internal temperature of the power supply, a watchdog IC for analog regulators triggers a shutdown when the MCU fails to send a proper heartbeat signal for any reason. An analog principle is used between the MCU and the control IOC. Here, a periodic heartbeat signal is part of the CHROMOSOME protocol. Together, both heartbeats ensure that the output of the power supply can only be enabled when the MCU is operational, and the slow control system is connected. In the IOC, a single PV to shut down all connected power supplies is provided. It is accessible as an "emergency shutdown" button from all power supply- related operator screens, or from other IOCs that implement additional interlock functionalities.

Table 1: Alarm Conditions in the Power Supply System

Enabled	Set V	Actual V	Mode	State
no	1.8 V	0.0 V	CV	OK
no	1.8 V	1.8 V	CV	OV
yes	1.8 V	1.5 V	CV	UV
yes	1.8 V	1.5 V	CC	OC
yes	1.0 V → 1.8 V	1.5 V	CV	OK
yes	1.8 V	2.0 V	CV	OV
yes	1.8 V	1.8 V	CV	OK

Any emergency shutdown triggers an orderly shutdown sequence that operates independently from the MCU by means of adjustable RC delays. It ensures that the various voltages for the system are brought down in a safe order. In the case of mains failure, a large capacitor provides enough power to complete the shutdown sequence.

Input Validation

All data requests (especially voltage and current set points) towards the power supplies are validated before being sent to the devices. Beyond hard limits set in the corresponding EPICS records, dynamic limits depending on other settings are considered. Typically, this includes a maximum allowed difference between a pair of voltages, or current limits based on the state of the system.

The verification is done purely inside the EPICS database. An example set of PVs implementing the functionality is shown in Fig. 2. In this figure the requested value in (#3) is validated by a dedicated calcout record (#4), which filters malicious values before they reach the ao record (#5) of the device driver. An alarm is triggered and stored in a bo record (#7) if the requested and set values do not match. Note that the behavior is different to just using DRVH and DRVL in the ao record (#5), as invalid values are not clipped to the maximum/minimum allowed, but completely ignored. Additional logic in the record (#2) resets the channel to idle (i.e. 0 V) when the emergency shutdown of the PS has been triggered as recorded in record (#1). A similar but slightly more complex logic is used to introduce dependencies between different values.

Access control lists are used to ensure that the actual set record (#5) is not accessible from outside the IOC, so that the verification cannot be bypassed. Access to the PVs defining the conditions is limited to experts' accounts.

Readback Monitoring

The provided voltages of the PS are monitored by both the PS itself, and by the external SC system. The EPICS database structure used to implement the monitoring is shown in Fig. 3 and explained in the following: First, the

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

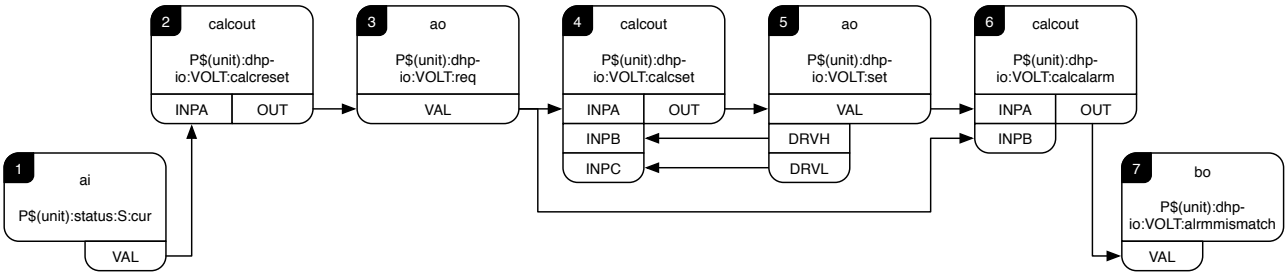


Figure 2: The EPICS database structure implemented to validate input values.

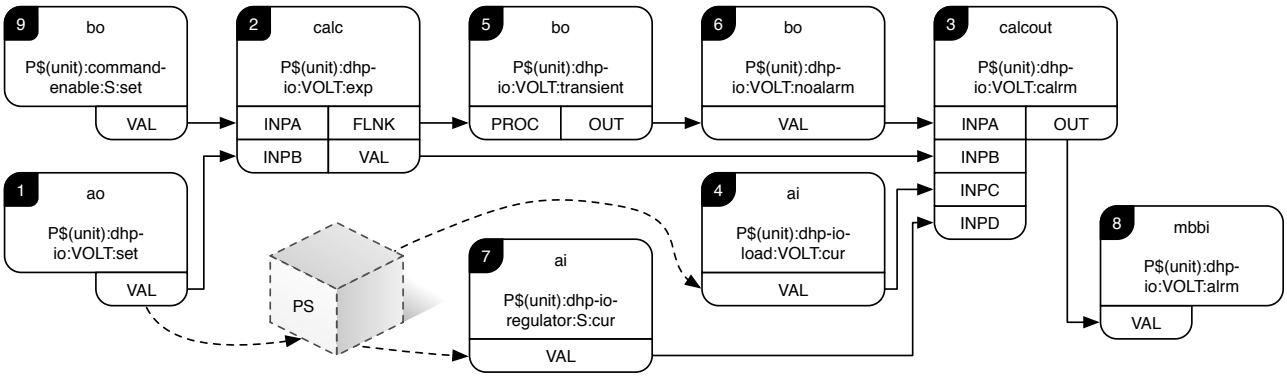


Figure 3: The EPICS database structure implemented to detect error states in the power supply system.

expected voltage at the load (record #2) is computed. It corresponds to the voltage requested by the user (#1), if the PS is enabled, or to 0V otherwise (#9). The (#3) record compares it to the actual voltage (#4), which is read back from the PS device. To avoid spurious alarms during voltage ramping steps, the alarm is suppressed for one second in case of the expected voltage changes. To that end, the bo records (#5) with OMSL closed_loop and (#6) with HIGH 1.0 are used to form a PV that is "on" for one second after a change in record (#2). The signal of this record is then considered in the logic for raising an alarm in case of a voltage mismatch. Also considered is the current regulator state (constant current vs. constant voltage) of the channel available in the record (#7). The effect of the implemented logic for a channel that is supposed to provide a fixed voltage is shown in Table 1. The detected states are OK (no alarm), OV (over-voltage), UV (under-voltage), and OC (over-current). All comparisons take the precision of the voltage readback into account by accepting readback data as matching within a sufficiently wide range around the desired value. Finally, the state is pushed into an mbbi record (#8), where mnemonic names are assigned to the states. This is the record whose state is shown in the GUI.

Radiation Monitoring

The PXD, as the innermost detector of Belle II, is most at risk to receive dangerous doses of radiation. Therefore a set of several radiation monitors are positioned at locations of expected high doses that have been identified from simulations and subsequently used to monitor the actual dose rate at all times. The monitors are controlled via a PLC that also computes interlock conditions on fast, extremely high rates, and slower but still higher than normal rates. In case of dangerous conditions, an immediate dump of the beams is requested from the SuperKEKB accelerator via a dedicated hard-wired line of communication.

The Modbus protocol is used to communicate with the PLC from the EPICS network. It is used to configure the interlock conditions, and to monitor the rates for display and archiving purposes.

CONFIGURATION DATABASE

The system configuration required to start a run is stored in a configuration database. An XML-like tree structure of configuration variables is efficiently stored in an SQL (currently PostgreSQL is used) database using a wandering tree algorithm to implement revision control system like functionalities. This way several files with multiple revisions per file can be stored in the database, and each entry can be queried for its history (change author, date and commit text). On run start, the desired run type category is used to identify the desired configuration.

The configuration data are made available to the system by means of an IOC that reads the configuration from the database and exports the configured PVs in the EPICS network.

The storage itself is tamper proof: After a commit, a revision cannot be modified in any way, it can only be superseded by a new revision. This is enforced by access controls on the database level and ensures that the active configuration at any point in time can positively be identified only from the configuration id in use at that time.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

RUN CONTROL

The run control of the PXD is implemented in a hierarchical way. The desired run state is set either by the user for PXD-only test runs, or received from the Belle II master run control. It is forwarded to all PXD subsystems, which can decide to participate in the automatic run control, or be left out of it for the current run. A new state is only considered as reached, when all dependent subsystems confirm it. The actions during the state transitions are implemented using the sequencer module for EPICS[4].

The most complex part controls the ASICs on the detector modules and the PS. About 100 steps are required to bring up the digital power for the ASICs, configure them, and finally enable the analog power to the detector. Sanity checks throughout the sequence ensure that the sequence completes successfully. The state machine code is simplified by means of the C preprocessor. After converting it with snc, the code is compiled in C++ mode.

On a larger scale, the Belle II detector overall implements its run control and power supply control using the Network Shared Memory system version 2 (NSM2). A bidirectional gateway between NSM2 and EPICS is used to establish the communication.

LOGGING INTEGRATION

A C++ logging library that can interface with JMS2RDB, a connector layer between the Java Message Service and Relational Databases, which is also used for log messages from CSS components, has been written. It forwards log messages to an ActiveMQ server via the STOMP protocol[5] that is converted by ActiveMQ to be understood by JMS2RDB. IOCs can use this library to implement their own logging. This allows for all messages from the entire system to be combined in the "Message History" view in CSS.

Besides the STOMP implementation, the library features a thread-safe and fast design, and automatic generation of a backtrace in case the application crashes. For the application thread, posting a log message is always $O(1)$, because the message is only put in a queue to be processed by another thread. The implementation is integrated into EPICS in the sense that a DBD file is provided that, when loaded, provides commands on the IOC level that allow the dynamic configuration of logging destinations, changes of the log levels per subsystem or destination, etc. Adjusting log levels via PVs is planned for the next release.

IMPLEMENTATION STATUS AND OUTLOOK

The development of the SC system for the PXD is progressing from the stage where all components are controlled independently from their respective developers to a more integrated mode of operation. Tests and measurements with several systems in use together are now common and working smoothly.

The next milestone is a test beam with all system components early 2016, when all components of the SC system are expected to be ready. During the commissioning phase of the accelerator, a partial PXD system, installed in the Belle II detector on the beam line, and the final detector in the assembly room have to be supported at the same time. A full setup of the slow control system will be used for both purposes. The hardware used in these setups will be combined for the final installation, so that two parts of each component are available to build a redundant system using pacemaker/drbd that offers the high-availability required in HEP experiment controls.

[4] http://www-csr.bessy.de/control/SoftDist/sequencer/

[5] http://stomp.github.io/stomp-specification-1.2.html

ACKNOWLEDGEMENTS

This work has been supported by the German Federal Ministry of Education and Research (BMBF).

The main author would like to thank Thorsten Röder for his valuable contributions to this work.

REFERENCES

[1] J. O. Hill, "Channel access: A software bus for the LAACS," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 293, pp. 352–355, Aug. 1990.

[2] D. Levit, I. Konorov, B. Zhuravlev, S. Paul, T. Gessler, S. Lange, D. Munchow, B. Spruck, W. Kühn, J. Zhao, and Z. Liu, *Data acquisition system of the DEPFET detector for the Belle II experiment*. IEEE, 2013.

[3] D. Levit, I. Konorov, D. Greenwald, and S. Paul, "FPGA Based Data Read-Out System of the Belle II Pixel Detector," *IEEE Transactions on Nuclear Science*, vol. 62, no. 3, pp. 1033–1039, 2015.

[4] C. Buckl, M. Geisinger, D. Gulati, F. J. Ruiz-Bertol, and A. Knoll, "CHROMOSOME: a run-time environment for plug & play-capable embedded real-time systems," *SIGBED Review*, vol. 11, Nov. 2014.

[5] S. Yamada, R. Itoh, K. Nakamura, M. Nakao, S. Y. Suzuki, T. Konno, T. Higuchi, Z. Liu, and J. Zhao, "Data Acquisition System for the Belle II Experiment," *IEEE Transactions on Nuclear Science*, vol. 62, no. 3, pp. 1175–1180, 2015.

[6] T. Konno, R. Itoh, M. Nakao, S. Y. Suzuki, and S. Yamada, "The Slow Control and Data Quality Monitoring System for the Belle II Experiment," *IEEE Transactions on Nuclear Science*, vol. 62, pp. 897–902, June 2015.

[7] K. Kasemir, X. Chen, and E. Danilova, "The best ever alarm system toolkit," *ICALEPCS09*, 2009.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ACTIVE MAGNETIC BEARINGS SYSTEM UPGRADE FOR LHC CRYOGENIC COLD COMPRESSOR, RADIATIONS MITIGATION PROJECT (R2E)

P. Arpaia, University of Napoli Federico II, Naples, Italy; CERN, Geneva, Switzerland
M. Girone, University of Sannio, Benevento, Italy; CERN, Geneva, Switzerland
M. Hubatka, MECOS AG, Winterthur, Switzerland
M. Pezzetti, CERN, Geneva, Switzerland

Abstract

During the normal operations of the Large Hadron Collider (LHC), the high hadrons flux level induced several Single Event Errors (failure caused by a particle passing through a sensitive device) to the standard electronics installed underground. Such events perturbed LHC normal operation. As a consequence, a mitigation plan to minimise radiation-induced failures and optimise LHC operation was started: Radiation to Electronics (R2E) mitigation project. The full paper will deal with the mitigation problem for LHC/Point 8 equipment and the main improvements for the equipment in LHC/Point 4, with special focus on the controllers for the Active Magnetic Bearings (AMB) used in the IHI-LINDE cold compressors. A proven approach based on frequency response analysis to assess the cold compressor mechanical quality will be presented. The hardware and software design, implemented to increase the global reliability of the system, will be highlighted. A corresponding experiment protocol was developed at CERN in collaboration with the Swiss Company MECOS and the Italian Universities of Sannio and Napoli Federico II. Preliminary experimental results showing the performance of the proposed approach on a case study for the cold compressor stage 1 in Point 4 will be finally reported.

INTRODUCTION

The level of the flux of hadrons with energy in the multi MeV range expected from the collisions at the interaction Points 4 and 8, which are not in the center on the cavern, unlike Point 1,2 or 5, (Fig. 1) will induce Single Event Errors (SEEs) in the standard electronics present in much of the control equipment. Furthermore, a risk of SEEs induced by thermal neutrons cannot be excluded. In the long-term, such events could perturb the LHC operation, possibly leading to critical situations for the machine elements and subsequent important downtime [1].

In particular this paper is focused on the impact on the electronics of MECOS Active Magnetic Bearing controllers. The high-performance bearing are applied in the LHC cryogenic cold compressor equipment in order to satisfy the need of covering vibration, stability and robustness in nominal operation and transition phases. In fact, magnetic bearing-equipped machines are suitable for unlimited, reliable and safe operation even in the presence of "large" residual unbalance levels [2]. Those high performances requires deeper user understanding of the magnetic bearing technology and a commissioned system cannot be modify so easily. The R2E Mitigation Project foresees shielding or relocating the equipment sensitive to radiation, which is presently installed in these critical areas, into safer areas. The majority of these mitigation activities were performed, in three phases, between (date) in Point 4 and between (date) in point 8. This document reports on these mitigation activities and their associated improvements. It presents the strategy applied, the enhancements on the modification for the new version of AMB used in the IHI-LINDE cold compressors which were needed to be replaced within a longer distance (from 20 m to 45 m).

STRATEGY APPLIED

The first LHC run took place from 2010 to 2013, on 14th February 2013 the Long Shutdown 1 (LS1) started. During the LS1 preventive maintenance, upgrade and necessary repairs were performed.

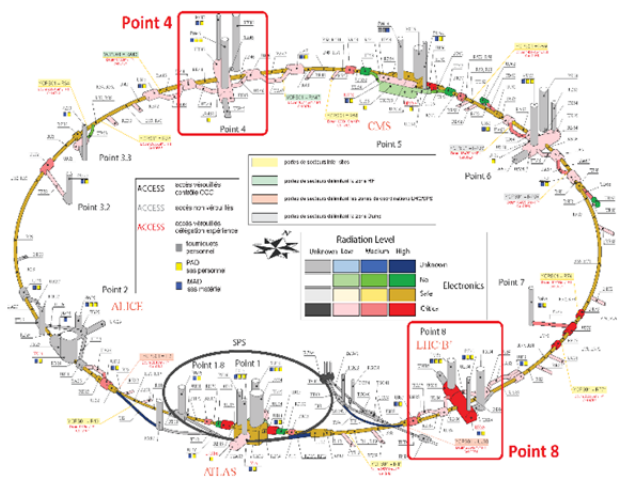


Figure 1: LHC critical areas considered by the R2E Mitigation Project.

A project called Radiation to Electronics (R2E) has been created in 2009 to mitigate the risk of failures due to single events upset (SEU) of the equipment installed in the tunnel and shielded areas. In 2011-2012, due to beam parameters increasing, SEUs had a significant

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

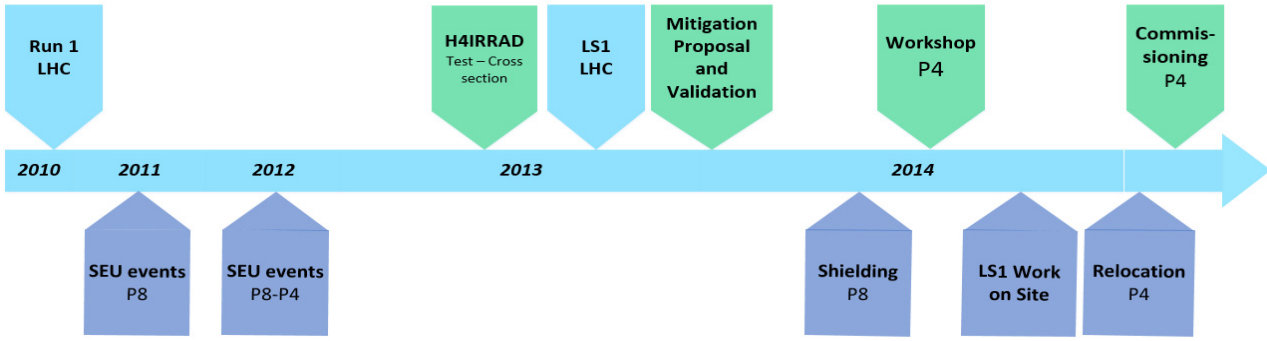


Figure 2: Actions strategy timeline.

impact in P8 and P4. These events caused several equipment failures. On October 2012, in order to determine the failure cross-section due to radiation of specific bulk equipment, a new test area was setup in the H4-beamline of the North Area: H4IRRAD. In this area several tests were conducted and they proved that industrial or commercial equipment are not qualified for radiation and they had to be removed from exposed areas. As a consequence the sensitive equipment located in these areas was relocated to safer areas and protected by new shielding walls (for P8). These mitigation activities were performed mainly during the LS1. Even though the commissioning had very tight constraints it was successfully accomplished.

AMB R&D

In this section, the enhancements on the controlling system for the Active Magnetic Bearings (AMB) used in the IHI-LINDE cold compressors are presented. All the hardware and software improvements were first tested in the CERN control and electricity laboratory, and after several validation tests were implemented in the cold compressor systems in LHC/Point 4. The hardware and the software improvements will be first discussed. At the end the proposed fault detection protocol for the cold compressors will be presented.

Hardware Improvements

Before the installation in the LHC cavern in P4, several tests and measurements on IHI-LINDE spare cold compressors systems (CCS) were performed together with MECOS Company.

Due to the electrical cabinets' relocation, as previously described for R2E project, longer cables (from 20 to 45 m) to connect the cold compressors and the magnetic bearings controllers were needed. Thanks to this project the cabling know-how has been transferred to CERN and the new cables were produced in situ and improved.

In Fig.3 an active magnetic bearing system architecture is schematized and it is possible to notice that two different type of cables are needed for an AMB control system: a cable for position sensors (*blue arrow*) and another cable (*red arrow*) for the levitation magnets (actuators).

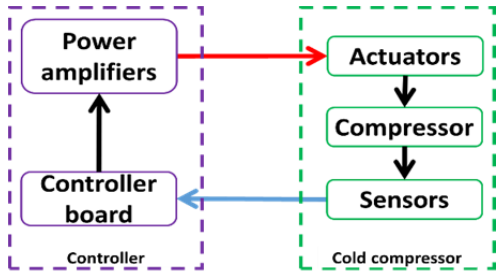
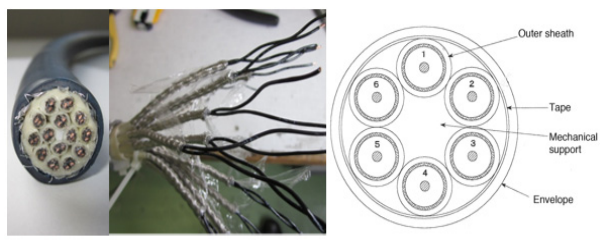


Figure 3: Active magnetic bearing system architecture.

Major attentions were put on sensors cables, because all the processor calculations are based on the position sensors measurements, and high noise levels could lead to system instabilities. In Fig.4.b a section of the existing sensors cable is showed.



a) b)

Figure 4: a) New sensors cables; b) Old sensors cables section.

A different configuration was adopted, and in particular, the following cables were chosen (Fig.4.a): IGUS Chainflex CF12 TPE-14 x 2 x 0.5. In these cables the wirings are twisted per pairs and they have a double shield: one shield per pair and a global shield for the entire cable.

In Fig.5.a and 5.b, position sensor test measurements, performed in CERN control and electricity laboratory, are showed. The measurements were carried out with the old controller (MBC150) and with the old and the new cable respectively, both 20 m long.

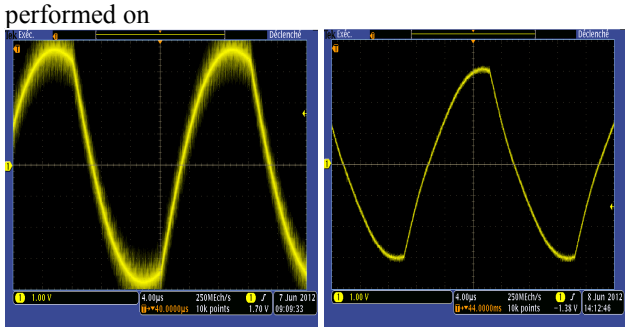
It is possible to notice that the overall noise level, with the new cable, is decreased. After these encouraging results, several measurements were performed on the new

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

cables and in particular several lengths were tested: from 25 up to 60 m.

The power supply cables for actuators were also changed (CERN NG18-9 x 2 x 1.0) but no tests were performed on



a) b)

Figure 5: a) Position measurement with the old cable: b) with the new cable.

them because, as mentioned, they are not as critical as the sensors cables for compressors controls.

Together with the new cables, new connectors (Amphenol MS4102E), for power supply and sensors, were installed on both the cables extremities and on the new MECOS controller to improve the connections robustness.

Several hardware modifications were also implemented inside the AMB controller (MBC–170A). The internal power supply was removed in order to enhance the system reliability. Instead of 4 different power supplies, one per unit, CERN adopted a redundant system: the 4 controllers were supplied by two independent electrical power systems circuits.

In more, the previous MBC generation needed two different software configurations for the four stages: one type could be used only for stage 1-2 and another type for stage 3-4. CERN, together with MECOS, decided to implement the same configurations for all the controllers, thus helping reduce the down-time of the LHC machine by simplifying the maintenance procedure. Moreover it improves the handling of the spare parts by reducing the number of controllers to be kept in stock.

Software Improvements

Together with the hardware enhancements, some software modifications were implemented in the supervision and control system for the cold compressors system in P4.

In case of cold compressors failures, on electrical/electronic or mechanical equipment, the entire 1.8 K cold box, and consequently the entire LHC machine is stopped. To restart the system an operator was required to go to the caver and reset the system manually in the electrical cabinet. In order to reduce the operation time a remote controller RESET was implemented on MBC–170A. Thanks to that the operators in P4 cryogenics control room can easily restart the entire cold box.

Together with the remote RESET, a monitoring interface, integrated in the LHC supervision and control panels, was implemented. In particular the axial rotor position and the radial bearings unbalance levels are monitored. The operators, monitoring the abovementioned trends, can avoid potential failures in a reactive manner.

Another enhancement, aimed to facilitate the control and the supervision system is the implementation of a customized analog output (4-20 mA) to monitor the compressor rotational speed. The maximum current (20 mA) is set according to the maximum compressor speed: 240, 500, 833, and 833 Hz respectively for compressor stages from 1 to 4.

In conclusion, a different control strategy on cold compressor stage 1 (CCS1) was adopted. All the cold compressors installed in the 1.8 K cold boxes are controlled with a SISO type controllers (Fig. 6.a [2]).

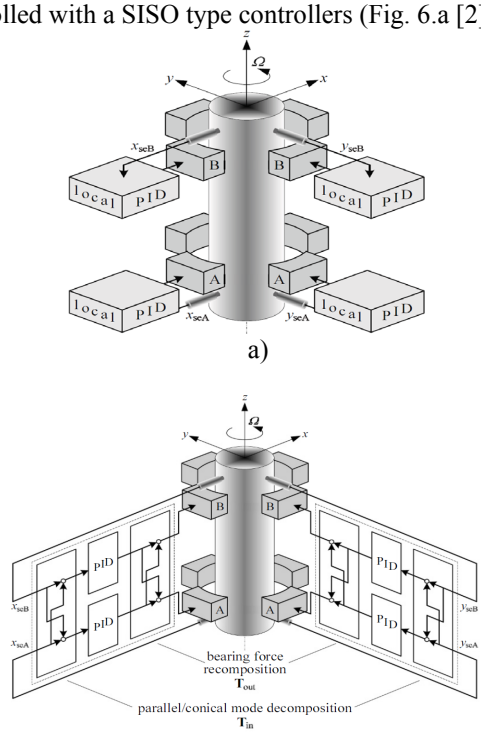


Figure 6: a) AMB controller architecture SISO; b) Parallel/conical modes control.

Due to the CCS1 intrinsic physical properties (higher gyroscopic effects due to impeller dimensions, first bending modes close to nominal operational frequency), during the controller parametrization, some difficulties in tuning the different SISO controllers independently were found. For this reason a MIMO strategy was applied (Fig. 6.b [2]). Such a controller, controls the *parallel* (centre of gravity motion on x and y planes) and *conical* (centre of gravity tilting motion on α and β angles) modes.

Cold Compressors Fault Detection

To evaluate the cold compressor mechanical quality a proven technique based on frequency response function analysis (FRF) has been used. A corresponding

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

experiment protocol was developed at CERN in collaboration with the Swiss Company MECOS and the Italian Universities of Sannio and Napoli Federico II. The proposed method will allow, during the LHC ordinary maintenance stops, to evaluate the mechanical quality of the cold compressors through a proactive fault detection.

Thanks to the nature of the active magnetic bearings, the system FRF $H = {}^{Y}/_{X}$ can be measured without additional sensors or measurement devices (Fig.7).

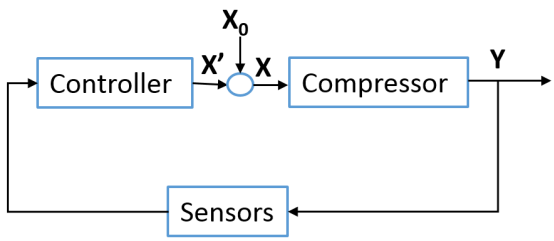


Figure 7: Measurement system architecture.

In particular, the system outputs Y are the rotor position measurements, while the inputs X are the sum of the controller signals (X') and an excitation signal X_0. In Fig.8 a FRF measurement for CCS1, performed in the lab, with the impeller in a vacuum case and in standstill conditions, is reported.

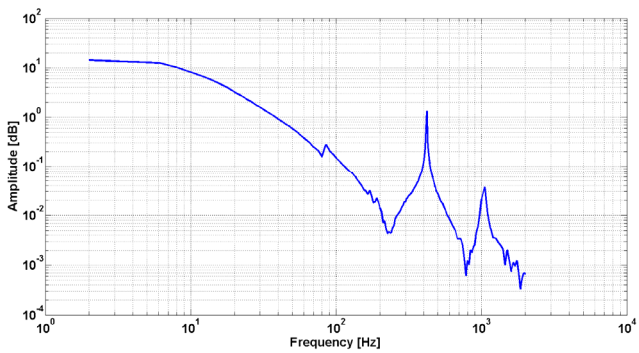


Figure 8: Cold compressor system 1 frequency response function.

During the P4 commissioning phase, FRF measurements were performed on all the cold box stages for both the installations: QURCA and QURCB. In particular, several test campaigns were conducted: a first measurements campaign was conducted for fixed rotational speeds (15, 22, 35 and 45 Hz for CCS1 to CCS4 respectively) at 300 K. Consequently, other measurements were conducted with the same rotational speeds but with the helium temperature was equal to 30 K. Finally a last test campaign was conducted on QURCB speeding up the compressors up to their nominal speed: 240, 500, 833, and 833 Hz respectively for compressors stages from 1 to 4.

During the first two measurements campaign on QURCA, a strange behaviour on CCS1 was detected. A special focus was addressed to the first resonance frequency peaks because from 300 to 30 K, they moved to lower frequencies (Fig.9).

Since the rotor is considered as rigid body, for lower temperatures, and especially for such a temperature gradient, the peaks were expected to move to higher frequencies. In more the CCS1 FRF presented some unexpected peaks at low frequencies, as it is possible to notice in Fig.9.Based on that, the compressor was replaced and sent back to the manufacturer.

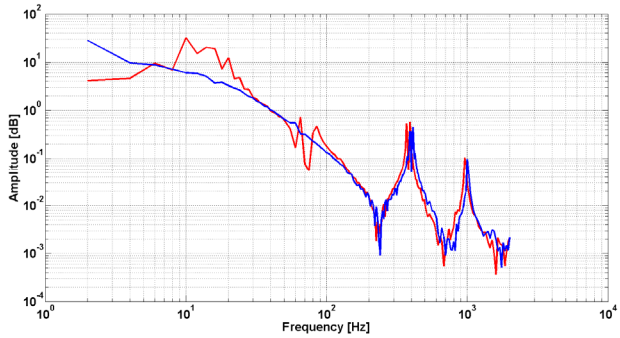


Figure 9: CCS1 installed in P4: frequency response function at 300 K–*blue*– and 30 K –*red*–.

The IHI-LINDE confirmed that some mechanical small problems were present. The same test campaign at 300 and 30 K was then performed on the new compressor (Fig.10). As it is possible to notice the behaviour is completely different from the previous compressor.

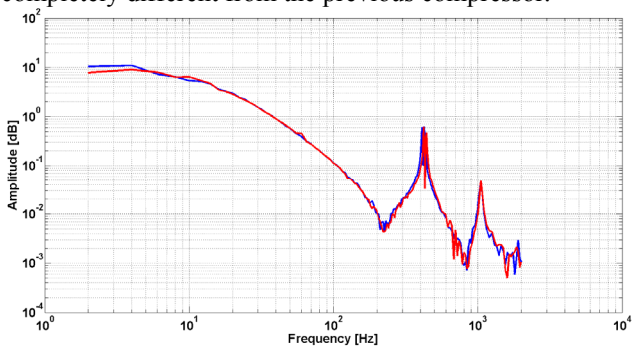


Figure 10: New CCS1 installed in P4: frequency response function at 300 K–*blue*– and 30 K –*red*–.

Thanks to this technique, in a preventive way, a mechanical compressor fault was detected and consequently a potential breakdown was avoided.

CONCLUSION

The work described has enhanced a deep knowledge of the cold compressor process control equipment. Moreover a new improved preventive maintenance plan has been set up based on the cold compressors mechanical quality evaluated by on-line measurement (done through AMB) compared with old one.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REFERENCES

[1] L. Perrot, “R2E strategy and activities during LS1”, CERN Chamonix, 2012.

[2] Gerhard Schweitzer, Eric H. Maslen, “Magnetic Bearings Theory, Design and Application to Rotating Machinery”, Springer, Berlin, 2009.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

BRINGING QUALITY IN THE CONTROLS SOFTWARE DELIVERY PROCESS

Z. Reszela, G. Cuni, C. M. Falcón Torres, D. Fernandez-Carreiras, G. Jover-Mañas, C. Pascual-Izarra, R. Pastor Ortiz, M. Rosanes Siscart, S. Rubio-Manrique, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain

Abstract

The Alba Controls Section (ACS) develops and operates a diverse variety of controls software which is shared within international communities of users and developers. This includes: generic frameworks like Sardana [1] and Taurus [2], numerous Tango [3] device servers and applications where, among others, we can find PyAlarm [4] and Panic [5], and specific experiment procedures and hardware controllers. A study has commenced on how to improve the delivery process of our software from the hands of developers to laboratories, by making this process more reliable, predictable and risk-controlled. Automated unit and acceptance tests combined with continuous integration, have been introduced, providing valuable and fast feedback to the developers. In order to renew and automate our legacy packaging and deployment system we have evaluated modern alternatives. The above practices were brought together into a design of the continuous delivery pipelines which were validated on a set of diverse software. This paper presents this study, its results and a proposal of the cost-effective implementation.

INTRODUCTION

The ACS designed, constructed and maintain the control systems required to run the facility. The core software components of the Alba control systems, like for example Tango – the middleware framework for building distributed control systems, Sardana – the scientific SCADA oriented to the experiment control and Taurus – the GUI library, are fruits of a collaborative effort of many institutes including Alba [6]. Peripheral components, like Tango device servers, Sardana controllers and macros or Taurus GUIs were either developed in-house or reused from the public repositories. Most of them are developed in Python.

Previously, the software development processes were mainly organized around one-person projects that made it difficult to conduct design questioning discussions and limited the knowledge flow. The software testing was neither formalized nor tried to be automated. This made the project transfers between the developers more difficult, for example on a developer leave. Newcomers, without the complete knowledge of the project, could not feel confident when introducing a change in the code without a way of testing it at the unit or at the system level. In some extreme cases this lead to development downtime periods, to abandoned projects or simply to buggy releases. Subversion (SVN) was the standard version control system (VCS) which did not encourage working branch-per-feature mode. Untraceable commit histories were not helping to enter into the project dynamics. The software packaging and deployment were done manually what is neither interesting nor motivating for the engineers. All the above problems and difficulties were not helping in reducing the long lead time – from the scientist request to the successful use of the software in the experiment.

DEVELOPERS COLLABORATION

Almost two years ago the ACS decided to introduce changes in the software development organization. The in-house projects were transformed from the individual to the group-based efforts. Furthermore, in the case of the two core and initially ACS's internal projects Sardana and Taurus (started at Alba in the previous decade) community-driven development and organization models were introduced. This had an impact on the following parts of the software development process.

Code Design

Internally, developers were organized in groups of 4-6 members – the *Scrum* teams [7]. A mixture of senior and junior developers were selected to build each team. The knowledge transfer activities became a second plan and continuous process. Information about the projects, previously restricted to the privileged project owners, quickly equalized among the team. Agile design practices were introduced: avoidance of upfront designs and plans and promotion of iterative and incremental developments. The Scrum activities brought many interesting design discussions that helped to achieve a better quality of the released products.

In parallel, the Sardana and Taurus project development and decision-taking was opened to a community composed mainly by synchrotrons similar to Alba (DESY in Germany, MaxIV in Sweden Solaris in Poland and ESRF in France), as well as by other institutions, companies and individuals (mostly within the Tango collaboration) who are basing their own developments in them. All these entities actively participate in the community activities. The community model requires remote collaboration tools. Sardana and Taurus are hosted on the Souceforge [8] platform and use a number of provided tools (e.g. issue tracker, mailing lists, wikis, etc.) in the code design processes. Discussions about the critical improvements and modifications are organized and formalized around public processes called Sardana Enhancement Proposal (SEP) and Taurus Enhancement

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Proposal (TEP) inspired in similar workflows from the Python [9] and Debian [10] projects.

Code Control

Since many developers started contributing to the projects the SVN became a limitation in many ways. Other systems were evaluated having the following reasons in mind: easier branching and merging (necessary in scattered organizations like ours) and easier tools and workflows for the code validation. Git won this competition bringing many other benefits such as its distributed architecture, better performance and the possibility of much cleaner history of commits with less effort. Hence Sardana and Taurus projects were migrated from SVN to Git, and the ACS started using Git for the newly created internal projects.

Sardana and Taurus branching rules were formalized according to *gitflow* [11,12] (Fig. 1). Two main branches and the core for the project are: the *develop* branch – where all the developments take place or get integrated into and the *master* branch – which always represents the latest production-ready state of the project. Each commit to the master branch gets tagged following the *semantic version system* (semver) [13] consisting on three dot-separated fields: major, minor and patch (Fig 1.). Increment in the major field indicates backwards incompatible changes in the API. The minor field increments when new functionality is added in a backwards compatible manner. And finally the patch field increments with the backwards compatible bug fixes. Other supporting branches are: the *feature* branches – where development of the SEPs, TEPs or feature requests take place, the *release* branch – the intermediate branch between the develop and the master states as well as the *hotfix* branches where the critical bugs in the master branch are fixed.

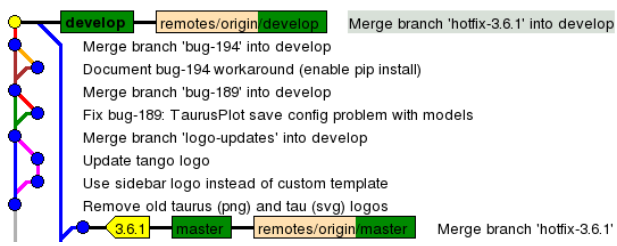


Figure 1: Extract from the Taurus git history demonstrating use of gitflow and semver rules.

Code Review

Together with collaborative software development, systematic code review practices were introduced. They rely on examination and validation of the contribution, usually done before checking it into the repository. In the case of the internal projects, lightweight peer reviews had proven to be a great way of improving the quality of code. Apart from that they help to equalize the technical knowledge within the team and reduce the information silos across the developers. The Sardana and Taurus projects apply more formal code reviews. All the code contributions are evaluated on the public forum but only the integration managers (representatives of each of the community institutes) are allowed to push the code into the canonical repository. The only tools used in this process are just a few git commands and the developers mailing list. Based on the current experience we can affirm that the quality of these projects has improved thanks to the code reviews. At the same time we can observe that the limited time of the integration managers is a bottleneck for patch integration.

TESTING

No testing strategy existed for the software projects developed and maintained by the ACS. SEP5 [11] established the common testing strategy for Sardana and Taurus. The following best practices are based on its results and more than a year experience with software testing.

Tests should be written before developers start work on the features that they test. Together, these tests form an executable specification of the behaviour of the system, and when they pass, they demonstrate that the functionality required by the users has been implemented correctly. The automated test suite should be run by the continuous integration (CI) service every time a change is made to the application – which means that the suite also serves as a set of regression tests. This strategy applies ideally to new projects, where with prior selection of the testing technology and the CI platform, developers could start writing and applying automatic tests in the process right from the beginning. However mid or legacy projects, like for example Sardana and Taurus, require a certain variation of the approach. The best is to start automating the most common, important, and high-value use cases of the application. Based on this selection, “happy path” tests covering these high-value scenarios should be automated. The rest of the scenarios should initially be tested manually. They should be automated only when one discovers that the same function is tested manually more than a couple of times.

Jenkins [14] was selected as the general CI system for the ACS projects mainly because of its big community of users, a broad and continuously growing set of plugins, and simplicity in setting them up and running.

The main purpose of the CI service is to test the software on each commit, providing fast feedback to the developers. Our use of Jenkins was extended to the unique registry of all the software maintained by the group. Hence even the projects which are not actively developed by the group or the external projects have their corresponding Jenkins jobs. These jobs are not triggered at each commit but their role is to automate the build processes and the integration tests with our control system.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONTINUOUS DOCUMENTATION

Sardana and Taurus projects use Sphinx [15] to create documentation. Sardana and Taurus documentation was originally hosted on the Alba's internet servers. This setup required manual builds and deployments on every update. Mainly due to the work overheads the documentation update frequency gradually decreased to twice per year, concurring with the biannual releases.

The documentation build processes of both projects were adapted to make them independent of specific libraries e.g. PyTango (Python language binding to Tango). This was achieved via a custom Python modules *mock* generator. Finally, both projects migrated thir documentation to the Read The Docs (RTD) [16] platform (Fig. 2), bringing the following benefits:

- Documentation gets built on every commit, early validating its correctness and notifying developers about any errors.
- Maintenance of the servers and the necessary software is outsourced to RTD.
- Several versions of the documentation e.g. *stable* or *latest* are available in a unique place.
- The documentation is available in different formats e.g. html, pdf, epub and is easily searchable.

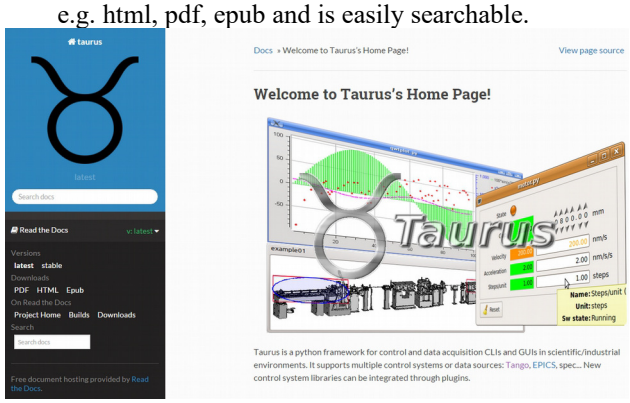


Figure 2: Taurus latest documentation available on RTD.

CONFIGURATION MANAGEMENT

The ACS manages all the software under maintenance with the bliss system [17]. The bliss system, developed by the ESRF, is a rpm-based packaging and Software Configuration Management (SCM) tool. It comprises two applications: the blissbuilder and the blissinstaller, and centralizes information about the packages and the hosts in a mysql database. Its main advantages are: the offline access to hosts' configurations and an intuitive to "non-packaging experts" graphical way of defining and creating packages. While bliss has served very well over its many years of use, it shows limitations when it comes to automatic package creation and deployment. First, all the package definition is spread in the bliss database tables and it is not possible to maintain it in the project code repository. This limits the creation of the project rpm package to the bliss system users only. The automatic package creation is not fully customizable, and requires modification of the package metadata in the database. However the biggest limitation of bliss comes with the configuration management and the automatic deployments (bliss does not provide a way to configure a group of hosts neither supports the Windows platform). While it seems possible to implement all the missing features in the bliss system (it is written in Python) or develop the complementary scripts, we decided to evaluate alternative public and widely used products.

SCM Tools

Many SCM tools exist, with the most popular choices being Puppet, Chef, Ansible [18] and Salt [19], and all of them are successfully used in many different organizations. In the process of comparison we had in mind the following aspects: precedence was given to free and open source projects, ideally developed in Python – the widest spread programming language in the group. Apart from that the ideal candidate was expected to support both Linux and Windows platforms in the most seamless way possible. Definition of the hosts configuration, and the possibility of applying them to different groups of hosts was also considered as an asset. Finally, the simplicity and the smoother learning curve for the ACS were also considered.

A closer look was given to the two Python based candidates: Salt and Ansible. They give a possibility to use the repository integration modules like the ones for apt, zypper or yum and allow operating system agnostic definitions of the hosts configurations. Salt architecture is based on a single master and distributed minions which exchange messages when necessary. However the use of minions is optional, and Salt could fallback to execute commands via ssh if necessary (a mixture of both solutions in the same system is possible). Ansible and Salt offer very similar features. Salt could eventually bring benefits in the future thanks to its higher scalability and the agent based architecture. In its favour speaks that the Alba IT Systems Section uses Salt for SCM of the High Performance Computing Centre of the Alba Synchrotron.

Packaging

Migration from the bliss system to Salt would eventually require an alternative way of creating software packages. This seems quite simple in case of the Python based projects since the standard packaging modules allow creation of rpm and deb packages for the Linux platform and exe and msi installers for the Windows platform. Sardana and Taurus already use distutils [20], and as a proof of concept creation of the rpm packages was successfully tested. In case of the Windows platform the msi format proved to be a better choice than the exe, since it allows the unattended installations necessary by the SCM. The migration from bliss would also require setting up package repositories where the packages would be uploaded and from where the SCM tool would pull the packages for installation.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

CONTINUOUS DELIVERY

Agile software development together with the continuous delivery aims to transform a concept into working software as fast as possible. Continuous delivery is based on fully automated, reliable, repeatable and constantly improving software delivery pipelines. Each change in the project code should trigger the pipeline execution and in case of success deliver a deployment-ready product. The negative result of an intermediate pipeline stage must break the pipeline execution and be immediately reported to developers who should stop the current work and fix the breaking change. Each project actively developed by the ACS could have its continuous delivery pipeline, initially comprising three stages: commit, acceptance and user acceptance. These stages should be executed sequentially only if the previous stage ended successfully. Each stage could be divided into parallel run jobs if necessary.

Based on the experimental implementation of the pipelines for Taurus (Fig. 3) and Sardana the following tools and setups are recommended. Jenkins works as the pipeline orchestrator where each job represents one pipeline stage. Jobs are interconnected and trigger the downstream jobs while the pipeline advances.

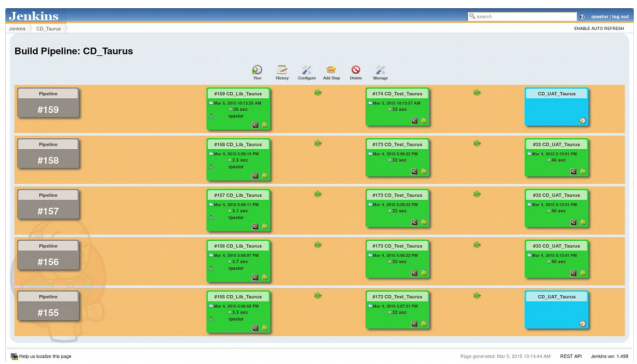


Figure 3: A proof-of-concept continuous delivery pipeline of Taurus project.

The commit stage is triggered on each change in the VCS. The commit stage should run the unit tests of the project and execute the static code analysis and in case of success build the software packages. Finally, all the packages get uploaded to the repository. It is very important that from now on, all the subsequent stages always use the same package created in the commit stage.

The acceptance test stage should take place in an environment as similar to production as possible. This stage should start from the package deployment to the acceptance test environment using the SCM, as it would be deployed to production. If the software works in production on various platforms e.g. Windows and Linux, the acceptance tests should also be performed on all of them. Finally the automated acceptance tests should be executed. The acceptance tests may require a specific configuration (also maintained under the VCS) which should be applied to the acceptance test environment before the tests execution. Preparation and maintenance of the acceptance test environments may be a tedious and error prone job. Execution of the acceptance tests in the Docker [21] containers showed to be a great solution to these problems. Docker is a platform for developing, shipping, and running applications using the container virtualization technology. The idea behind it is to maintain the Docker images for each of the acceptance test environments and spin them up on demand of the pipeline execution. This solution provides lightweight, reliable and isolated testing environments occupying the resources only when needed. Docker integrates well with Jenkins via plugins and one of them allows to seamlessly use Docker containers as the Jenkins slave nodes.

Successful acceptance stage should notify developers that the package is ready for the user acceptance tests. These tests should be executed manually following a well-defined testing scenarios and it is very important to do that in an environment as similar to production as possible. While the tests must be executed manually the preparation of the testing environment should be automated thanks to the SCM. Based on the user acceptance test results a decision is taken if a package is production ready or not. Of course executing the user acceptance tests at each commit could become expensive, so this stage should be done on demand.

NEXT STEPS

Sardana and Taurus projects could already apply the continuous delivery strategy to their biannual releases. Ideally their pipelines should be accessible by the whole community of developers, both in and outside of ALBA. This may be solved by using cloud providers for the continuous delivery tools, but it has not been investigated yet. While the decision of the eventual migration to the new SCM is blocked by still very shallow knowledge about the software packaging, it also depends on the upgrade of the general platform of the Alba control system. Online code review platforms may bring new quality to the current review processes, making them more accessible to the developers and reducing the workload on the integration managers.

ACKNOWLEDGEMENT

We would like to thank the rest of the Alba Controls Section and especially to Roberto Homs, Daniel Roldan, Jordi Andreu, Fulvio Becheri and Sergi Blanch, and the Taurus and Sardana community members for their active work in this project. We would also like to thank the IT Systems Section of Alba and especially to Sergi Puso, Marc Rodriguez, Mario Diaz and Antoni Pérez as well as the MIS Section and especially to Daniel Salvat for their valuable feedback.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REFERENCES

[1] T. Coutinho et al. "Sardana, the Software for Building SCADAs in Scientific Environments", ICALEPCS2011, Grenoble, WEAAUST01.

[2] C. Pascual-Izarra et al. "Effortless Creation of Control & Data Acquisition Graphical User Interfaces with Taurus", ICALEPCS 2015, Melbourne, THHC3O03.

[3] Tango website: http://www.tango-controls.org

[4] S.Rubio-Manrique et al., "Extending Alarm Handling in Tango.", ICALEPCS 2011, Grenoble, MOMMU001.

[5] S.Rubio-Manrique et al., "PANIC, a suite for visualization, logging and notification of incidents.", PCaPAC 2014, Karlsruhe, FCO206.

[6] Alba website: http://www.albasynchrotron.es

[7] G. Cuni et al. "Introducing the SCRUM Framework as Part of the Product Development Strategy for the ALBA Control System", ICALEPCS 2015, Melbourne, MOD3O04.

[8] Sourceforge website: https://www.sourceforge.net

[9] Python Enhancement Proposals: https://www.python.org/dev/peps

[10] Python Enhancement Proposals: http://dep.debian.net/deps/dep0

[11] Sardana Enhancement Proposals: http://sf.net/p/sardana/wiki/SEP

[12] Gitflow branching model website: http://nvie.com/posts/a-successful-git-branching-model

[13] Semantic versioning system website: http://http://semver.org

[14] Jenkins website: http://jenkins-ci.org

[15] Sphinx website: http://sphinx-doc.org

[16] Read The Docs website: http://readthedocs.org

[17] The ESRF Beamline Control Unit website: http://www.esrf.eu/Instrumentation/software/beamline-control/BLISS

[18] Ansible website: http://www.ansible.com

[19] Salt website: http://saltstack.com

[20] Distutils documentation website: https://docs.python.org/2/library/distutils.html

[21] Docker website: http://www.docker.com

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

LASER –DRIVEN HADRON THERAPY PROJECT

F. Scarlat, Valahia University of Targoviste, Targoviste, Romania
F. Scarlat, A. Scarisoreanu, National Institute for Laser, Plasma and Radiation Physics-INFLPR, Bucharest-Magurele, Romania
N. Verga, University of Medicine and Pharmacy, Carol Davila, Bucharest, Romania
Fl. Scarlat, Bit Solutions, Bucharest, Romania

Abstract

The laser beam (10 PW, 15 fs, 150 J, 10^{23} W/cm^2) generated by APOLLON Laser System, now under construction on Magurele Platform near Bucharest may also be applied in radiotherapy. Starting from this potential application, location of malign tumors in patient may be situated, e.g., superficial (≤5 cm), semi-deep (5-10 cm) and profound (>10-40 cm). This paper presents the main physical parameters of a research project for a therapy based on hadrons controlled by laser, for the treatment of superficial and semi-deep tumors. Energies required for pin-pointing the depth of such tumors are 50-117 MeV for protons and 100–216 MeV/u for carbon ions. Hadron beams with such energies can be generated by the mechanism Radiation Pressure Acceleration (RPA). Besides, the control systems to provide the daily absorbed dose from the direct and indirect ionizing radiation at the level of the malign tumor of 2 Gy in 1 or 2 minutes with expanded uncertainty of 3 % are presented.

INTRODUCTION

APOLLON Laser System of 10 PW (150 J, 10^{23} W/cm^2, 15 fs) [1], which is under construction on Magurele Platform close to Bucharest, could employ the relativistic laser beams ($I_0\lambda_0^2>10^{18}$ Wcm^{-2}μm^2) as an optional application for researches and experiments to accomplish a final project of laser-driven hadron therapy. According to IAEA-TRS 398 standards, the hadrons must have the kinetic energies of 50 to 250 MeV for protons (P) and 100 to 450 MeV for carbon ions (C) [2].

The hadron beams have the in-depth absorbed dose distributions characterized by small relative doses in the entrance area up to the proximity of the practical path end when the dose is rapidly increasing along a very narrow area in the form of a peak called Bragg Peak (Fig. 1). Function of the hadron energy, this peak is occurring at any depth starting from 2.7 cm (50 MeV-P, 100 MeV-C) up to the depth of 38 cm (250 MeV-P, 450 MeV-C), which corresponds to the 1st stage tumors.

In case of tumors of other stages (2nd-4th) the extension of Bragg–Spread Out Bragg Peak (SOBP) is used, with the residual path centred (focused) in the middle of the tumor. At present, the hadron radiotherapy (RT) employs: conventional accelerators of radiofrequency (RF) of NC type, isochronous cyclotron (IBA Protons P: 235 MeV; RF 106 MHz; SC isochronous cyclotron (Varian P; 150 MeV, 72.8 MHz), SC Syncyclotron (Mevion P: 250, 20t), and Synchrotron: slow-cycling (Siemens C:85-430 MeV/u) and proton linacs [3].

At present there are R&D dedicated to obtaining a compact unit. In view of that, a FFAG is in process to be finalized, a cyclinac (cyclotron + high frequency linac) and a Dielectric Wall Accelerator [4].

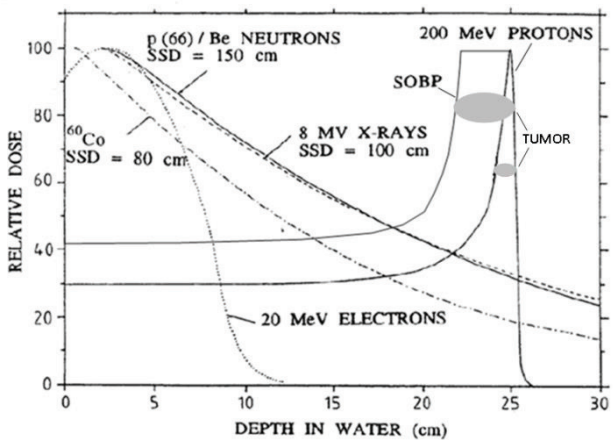


Figure 1: Depth dose distributions in water for X & γ rays, neutron and proton.

Besides, the gradient of the acceleration, limited to 50-100 MV/m in RF linac, may be removed by substituting the RF wave with the laser EM wave that allows an acceleration gradient of 50-100 GV/m [5]. This paper presents the physical parameters of a project employing a laser accelerator for RT of tumors located up to 10 cm in depth using RPA mechanism.

RPA ACCELERATION METHOD

The mechanism called Radiation Pressure Acceleration (RPA) is based on the transfer of energy and the flux of the moment between the linear polarized (LP) laser pulse or circular polarized (CP) laser pulse and the particles inside a solid target. When the target is thin, the acceleration mechanism is called "light sail" (LS) RPA, and when the target is thick (as the case in this paper), the mechanism is called "hole boring" (HB) RPA. The RPA theory is described, for example, in [6].

The HB RPA mechanism developed by Robinson & co. in relativistic regime CP gives the expression for the momentum balance of the plasma surface [7]

$$(2I_0)\left(1/\gamma_f^2\right)\left(1+\beta_f\right)^2 = 2\gamma_f^2 m_i n_i v_f^2 = (2A/Z)\gamma_f^2 m_p n_{e0} v_f^2 \quad (1)$$

where $I_0=(1/2)\varepsilon_0 c E_o^2$ is the laser radiation intensity ($I_0\lambda_0^2$[W/cm^{-2}μm^2]$=1.37\cdot10^{18}\cdot\alpha\cdot a_o^2$ cu α=½ for LP/CP laser pulse), $\varepsilon_o=8.854\cdot10^{-12}$ [As/Vm] is the permittivity of

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

free space, c the speed light, v_f the speed of the laser front or HB velocity, $\beta_f=v_f/c$, $\gamma_f=(1-\beta_f)^{-1/2}$, m_i the ion mass, n_i the ion number density, $A=m_i/m_p$, $m_p=1836m_e$ is the proton mass, m_e the electron mass, Z the ionic charge state and $n_{eo}=n_iZ$ the electron density. Also, E_o [TV/m]$=2.7\cdot10^{-9}I_o^{1/2}$ [W/cm^2]$=3.21a_0/\lambda_0$ [μm] is the peak amplitude of the transverse electric field of LP laser pulse where λ_0 is the wavelength [8]. In the relation of the monochrome radiation I_0 above, the parameter a_0–figure of merit-is the dimensionless amplitude of the transverse electric field E_0 of a LP laser pulse

$$a_0 \equiv eE_0/m_e\omega_0 = 0.85\cdot10^{-9}I_0^{1/2}[W^{1/2}/cm]\lambda_0[\mu m] \quad (2)$$

and $a_0/\sqrt{2}$ for the CP laser pulse, with e the electric charge and $\omega_0=2\pi c/\lambda_0$ the frequency of laser wave. Since parameter a_o, is the ratio between the EM wave energy and the electron energy at rest, it indicates the laser operation regimes. For the acceleration of the electron ($m_{0,e}=0.511$ MeV/c^2), the operation regime becomes relativistic when $a_0\geq1$. The acceleration of the proton ($m_{0,p}=938.27$ MeV/c^2) at relativistic energies ($a_o=1836$) requires an intensity $I_o\lambda_0^2=4.62\cdot10^{24}$ [Wcm^{-2}μm^2] [9].

The second important parameter related to the density of a target ($\rho=m_in_i$, $\rho(H^+)=1$ g·cm^{-3}, $\rho(C)=2.27$ g·cm^{-3}), introduced in [6], as a figure of merit Ξ, noted by b_o in this work, is the dimensionless amplitude of the peak intensity

$$b_0 \equiv I_0/\rho c^3 = (Z/A)\left(m_e n_c/2m_p n_{e,0}\right)a_0^2 \quad (3)$$

where $n_c=m_e\varepsilon_o\omega_p^2/e^2=2I_0/\alpha\cdot m_ec^3a_0^2$ is the critical density defined as the electron density at which the plasma frequency $\omega_p=5.64\cdot10^4$ (n_c[cm^{-3}])$^{1/2}$ becomes equal with the laser frequency $\omega_0=2\pi c/\lambda_0$. The overdense plasma acts like mirror when $\lambda>\lambda_0$ or ($n_e<n_c$, $\omega_0<\omega_p$). The parameter b_0 determines the value of the laser intensity I_0, required to obtain the kinetic energy (P:250 MeV and C:450 MeV/u) and intensity of the hadron beams (10^{10}pps) for therapy. Experimental the laser pulse intensity is measurable by determining the laser pulse energy $\mathcal{E}_0$, the pulse duration τ_{FWHM} at full-width at half maximum and the spot radius. Also, it is possible to determine the peak power P_p, the peak electric field E_p, HB velocity β_f, the accelerated ion kinetic energy T_i, the conversion efficiency of laser energy to the ion energy χ, and the total number of ions per bunch that can be accelerated N_i.

PROJECT PHYSICAL PARAMETERS

Employing the HB-RPA mechanism it was possible to determine the system parameters of the laser-driven hadron therapy research project at lab level.

After having commissioned the APPOLON 10 PW laser system, the experiments with the beams generated by the laser are started. The preliminary main parameters are presented in Tables 1 & 2 [9].

Table 1. Main Parameters for the Proton Therapy

Quantity	Simbol [UM]	Minim value/ Maxim value
1. Proton beam		
Kinetic energy	Tp [MeV]	50/116.70
Magnetic rigidity	BR[Tm]	1.02/1.61
2. Laser beam		
Amplitude parameter	a_0	347/575
Electric field	E_0 [PV/m]	0.866/1.434
Laser intensity	I_0 [W/cm^2]	$1\cdot10^{23}$/$2.75\cdot10^{23}$
Pulse width	τ [fs]	144/158
Peak power	P_0 [PW]	7.1/8.4
Beam area	A [μm^2]	7.1/3.14
Pulse energy	$\mathcal{E}$ [kJ]	1.02/1.36
3. Proton target		
Intensity parameter	b_0	0.03688/0.10193
Electric field	E_{x0} [PV/m]	0.737/1.22
Conversion efficiency	%	27.75/16.93
Acceleration time	τ_{acc} [fs]	1.4/1.32
Target thickness	d [μm]	6.98/11.50

Table 2. Main Parameters for the Carbon Ion Therapy

Quantity	Simbol [UM]	Minim value/ Maxim value
1. Carbon ion beam		
Kinetic energy/nucleon	$T_{u,I}$ [MeV]	100/215.83
Magnetic rigidity/nucleon	BR[Tm]	1.476/2.235
2. Laser beam		
Amplitude parameter	a_0	347/572
Electric field	E_0 [PV/m]	1.964/3.254
Laser intensity	I_0 [W/cm^2]	$5.17\cdot10^{23}$/$1.42\cdot10^{24}$
Pulse width	τ [fs]	207/221
Peak power	P_0 [PW]	16.23/44.6
Beam area	A [μm^2]	3.14/3.14
Pulse energy	$\mathcal{E}$ [kJ]	3.4/9.8
Ion numbers	N_i	$6.86\cdot10^{12}$/$1.16\cdot10^{13}$
3. Carbon ion target		
Intensity parameter	b_0	0.08428/0.23097

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Electric field	$E_{x,0}$ [PV/m]	1.57/2.35
Conversion efficiency	%	36.73/49
Acceleration time	τ_{acc} [fs]	11.32/7.54
Target thickness	d [μm]	13.90/21.38

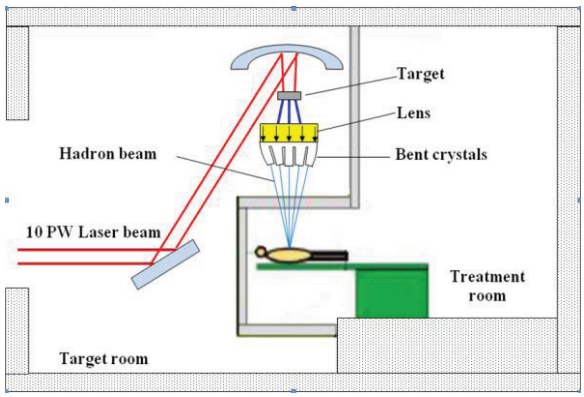


Figure 2: Proposal of an optical-laser gantry system which using mirrors and the phenomem of chaneling in bent crystals as steering, bending and focusing device for hadron beams.

The reference point for this project is represented by work [10] which is a synthesis of 2 versions on the use of laser accelerators. With the first version, a conventional accelerator is substituted with a laser accelerator and the target is located at the target chamber entrance. With the second version (Fig. 2) the laser beam is guided by mirrors up to the target located in the Gantry system, above the treatment table. We consider that the second version, the one which is to use the channelling phenomenon in bent crystals for to make the hadron beam bending and focusing towards the tumor, is the best.

As an example, see the channelling properties for P (50-250 MeV) and C (100 MeV/u-450 MeV/u) in the case of employing crystals of type silicon /germanium /tungsten, Tsiganov radius is of 16.3/7.8/2.1 [cm] respectively the equivalent magnetic fields are of 6-15/13-34/48-115 [Vs/m^2] for protons and 105-250/227-523/885-1942 [Vs/m^2] for carbon ions [11].

In case of this project, the first version is not considered because of the unavailability of a gantry system to be adjusted (modified). The second version cannot be applied either because it take much time to get the research results of the channelling phenomenon in bent crystals. Therefore we decided on a third version presented in Fig. 3 [12].

This version offers the same device for making, bending and focusing the hadron beams (P & C). The physical parameters of the hadron beams will serve to elaborate the opto-electronic design.

This assembly will be a transportable module that is to be component – by – component inserted and then assembled, or it may be placed as a unit in the target chamber of the 10 PW laser. It will be used to finalize the structure, geometry and characterization of the target. The hadron RT successful application depends on.

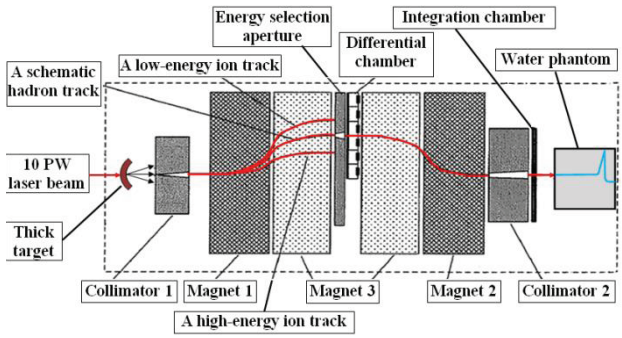


Figure 3: A schematic diagram showing the hadron generation and beam formation in fixed geometry [12].

BEAM MONITORING

Absorbed Dose

The absorbed dose to water when irradiated by hadron beams of quality Q (= Q_P or Q_C) is given by relation [2]

$$D_{w,Q} = M_Q N_{D,w,Qo} k_{Q,Qo} \ , \qquad (4)$$

where M_{corr} is ionization chamber reading in [C] corrected for influence quantities, $N_{D,w,Qo}$ [Gy/C] the absorbed dose to water calibration factor of ionization chamber in a beam of quality Qo(=^{60}Co), and $k_{Q,Qo}$ the beam quality correction factor to account for the use of the calibration factor in a different beam quality Q.

$$k_{Q,Qo} = \frac{(s_{w,air})_Q (W_{air}/e)_Q p_Q}{(s_{w,air})_{Qo} (W_{air}/e)_{Qo} p_{Qo}}. \qquad (5)$$

In our case, $k_{Q,Qo}$ for proton or carbon ion beams, is given by the relation (5) where $s_{w,air}$ is the water to air mass collision stopping power ratio, (W_{air}/e) is the mean energy required to produce an ion pair in dry air and p is a correction factor accounting the perturbation by the presence of the ion chamber in the phantom.

The factors specific to hadron beams Q (Qp&Qc) have the following values: $(s_{w,air})_{Qp}/(w_{air}/e)_{Qp}/p_{Qp}$=function of energy/34.50/1.0 and for $(s_{w,air})_{Qc}/(w_{air}/e)_{Qc}/p_{Qc}$=1.330/34.23/1.0. The values in case of the calibration at the ^{60}Co γ radiation beam quality (Q_0=Q_{Co}) for a PTW 31010 Markus type plan parallel ionization chamber are $(s_{w,air})_{Qo}/(w_{air}/e)_{Qo}/p_{Qo}$=1.133/33.97/1.003 [-/JC^{-1}/-] [13].

Combined Uncertainty

IAEA TRS 398 recommends the use of ionization chambers for depth values of $z \geq 0.5$ g/cm^2 for protons and $z \geq 2$ g/cm^2 for carbon ions. The uncertainty of the charge M_Q can be assessed by statistical analysis of a series of observations. The uncertainty of M_Q is of type A.

The uncertainties of $N_{D,w}$ and $k_{Q,Qo}$ are of type B. The combined uncertainty, u_C, of absorbed dose $D_{w,Q}$ in the quadratic addition of type A and B uncertainties is [14]:

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

$$u_c(D_{w,Q}) = \sqrt{u_A^2(M_Q) + u_B^2(N_{D,w,Qo}) + u_B^2(k_Q)} \quad (6)$$

Assuming no correlation between the components, the expression of the relative combined standard uncertainty yields.

$$\frac{u_c^2(D_{w,Q})}{D_{w,Q}^2} = \left(\frac{1}{M_Q}\right)^2 u^2(M_Q) + \left(\frac{1}{N_{D,w,Qo}}\right)^2 u^2(N_{D,w,Qo}) + \left(\frac{1}{k_{Q,Qo}}\right)^2 u^2(k_{Q,Qo}) \quad (7)$$

Standard uncertainties in D_w(TRS 398, ICRU 78) are $u(N_{D,w})$=0.6 in SSDL for $k_{Q,Qo}$ calculated of 2-2.3 for protons and of 3-3.4 for carbon ions [15].

Aspects of a Control System for Particle Therapy

The therapy employing the ion beams shows some important parameters that need to be subjected to a control. The energy of the laser pulse, $\mathscr{E}_0$, transferred to the ion beam, $\mathscr{E}_i=N_iT_i$ with the energy conversion efficiency χ given by HB-RPA mechanism ($\chi\mathscr{E}_0=\mathscr{E}_i$), successively amplified with the pulse repetition frequency f_R, determines the beam average power P_{rms}, and then amplified by the irradiation time t_S, with the number of irradiation sessions n_S, it determines the total energy absorbed into the tumor, $\mathscr{E}_T=m_TD_T$. Based on the energy balance relation, it is possible to determine the number of ions per bunch $N_i=m_TD_T/n_Sf_Rt_S\mathscr{E}_i$ required to apply the dose D_T(=2/70Gy) in n_S(=1/35) irradiation session. In point of the legal aspects related to the control system of such parameters, they are certified by the National Commission for Nuclear Activities Control (CNCAN, Romania). Also are met the ALARA criteria. [16].

CONCLUSIONS

In this paper, the method employed is based on HB-RPA mechanism aimed to generate hadrons by means of thick targets.

By means of the mechanism was selected the parameters of the laser accelerator supplying hadron beams for therapy of malign tumours located up to 10 cm.

They constitute input main parameters for the opto-electronic project of the gantry system

The gantry system is a device capable to be installed inside the 10 PW laser target chamber. It was selected both for studying target geometry and for selecting the beam energy.

REFERENCES

[1] N.V. Zamfir, "Extreme Light Infrastructure – Nuclear Physics ELI-NP", Experimental Programme Workshop at ELI-NP, Bucharest , Romania, (2012).

[2] IAEA TRS 398, "*Absorbed dose determination in external beam radiotherapy. An international code of practice for dosimetry based on standards of absorbed dose to water"*, Tehnical Report no 398.

[3] U. Amaldi et al.,"Accelerators for hadrontherapy: From Lawrence Cyclotrons to Linacs", Nucl. Instr. and Meth.in Phys. Res. A, 620 (2-3), 577 (2010).

[4] K.W.D. Ledingham, P.R. Bolton, N. Shikazono,C - M. Ma, "Towards Laser Driven Hadron Cancer Radiotherapy: Review of Progress", arxiv.org/abs/ 1405.2657, 12 May 2014.

[5] E. Esarei, P. Sprangle, J. Krall, A. Ting, IEEE Transactions on Plasma Science, 24 (2), 252 (1996).

[6] A. Macchi, M. Borghesi, M. Passoni, Rev. Mod. Phys.85, 751 (2013).

[7] A.P.L. Robinson, P. Gibon, M. Zepf, S. Kar, R.G. Evans and C. Bellei, Plasma Phys. Control. Fusion 51, 024004 (2009).

[8] P. Sprangle, E. Esarey and J. Krall, Bull. Am. Phys. Soc. 40, 1861 (1995).

[9] F. Scarlat, A. Scarisoreanu, N. Verga, Fl. Scarlat, C. Vancea, Journal of Intense pulsed Lasers and Applications in Advances Physics, 4 (4) 55 (2014).

[10] S. V. Bulanov and V. S. Khoroshkov, Plasma Phys. Rep.28, 453, 2002.

[11] R.A. Carrigan Jr., "*On the possible applications of the steering of charged particles by bent single crystals including the possibility of separated charm particle beams*",FERMILAB-Pub_80/45-EXP 7850.

[12] C.-M. Ma, I. Veltchev, E. Fourkal, J.S. Li, W. Luo, J. Fan, T. Lin and A. Pollack, Laser Phys.,16 (4) 1 (2006).

[13] F. Scarlat, N. Verga, A. Scarisoreanu, E. Badita, M. Dumitrascu, E. Stancu, C. Vancea, Fl. Scarlat, Journal of Intense Pulsed Lasers and Applications in Advanced Physics, 3 (2) 15 (2013).

[14] F. Scarlat, A. Scarisoreanu, R. Minea, E. Badita, E. Sima, M. Dumitrascu, E. Stancu, C. Vancea, "Secondary Standard Dosimetry Laboratory at INFLPR", Optoelectronics and Advanced Materials – Rapid Communications, Vol.7, No.7-8, p.618-624, August 2013.

[15] IAEA TEDOC 1585 "*Measurements Uncertanity. A Practical Guide for Secondary Standard Dosimetry Laboratories*", (Vienna 2008).

[16] O. Sotolongo-Grau, D. Rodríguez-Pérez, J. A. Santos-Miranda, O. Sotolongo-Costa, J. C. Antoranz, Math. Med. Biol., 26 (4) 297 (2009).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A UNIFIED APPROACH TO THE DESIGN OF ORBIT FEEDBACK WITH FAST AND SLOW CORRECTORS

S. Gayadeen, M. T. Heron, G. Rehm, Diamond Light Source, Didcot, UK

Abstract

A unified control design is proposed to simultaneously determine the inputs to both the fast and slow arrays of correctors. By determining the interaction of the spatial subspaces of each array of correctors, spatial modes which require both fast and slow correctors can be identified. For these modes, a mid-ranging control technique is proposed to systematically allocate control action for each corrector. The mid-ranging control technique exploits the different dynamic characteristics of the correctors to ensure that the two arrays of actuators work together and avoid saturation of the fast correctors. Simulation results for the Diamond Storage Ring are presented.

INTRODUCTION

In this paper, an approach to electron orbit controller design is presented where there are two arrays of actuators with different dynamics. The aim is to design a controller which uses both corrector arrays to meet the electron beam stability requirements so that: the time taken for computation of the control action is not greater than that of the single array system, the tuning for performance is intuitive and the specific dynamics and constraints of each array are addressed. In order to meet these goals, a method of exploiting the knowledge of the spatial responses of the arrays of correctors is used to determine the interaction between the controllable subspaces. As a result the problem can be decomposed into a series of single-input, single-output (SISO), two-input, single-output (TISO) and two-input, two-output (TITO) problems. As a consequence, the online computation is minimised by selecting appropriate control directions when subspaces of the two arrays overlap. In particular, if the control directions of the two arrays are orthogonal, then the problem can be interpreted as a decoupled structure (i.e. two SISO structures) and SISO design techniques can be applied. If however, the control directions align, the problem has a TISO structure, and mid-ranging control is proposed. The term mid-ranging control typically refers to the class of control problems where two actuators are manipulated to control one measured variable. Furthermore there is the condition that one input should return to its midpoint or some setpoint. The inputs usually differ in their dynamic effect on the output and in the relative cost of manipulating each one, with the fast input normally being more costly to use than the slow input [1]. Therefore mid-ranging control schemes seek to manipulate both inputs upon an upset but then gradually reset or mid-range the fast input to its desired setpoint. Mid-ranging control therefore is suitable for the fast and strong corrector problem. Though it may be possible to manipulate each actuator separately, for electron beam control it is desirable to simultaneously manipulate both inputs, as the strong correctors are bandwidth limited. Additionally, an important characteristic of mid-ranging applications is that input constraints on the faster input are avoided. There are several approaches to designing mid-ranging controllers [1], but in [2] an IMC structure is used, which is adopted in this paper since it is consistent with the design for a single array of actuators at Diamond Light Source [3].

MULTI-ARRAY CONTROLLER DESIGN

Subspace Interaction

For a storage ring electron orbit control system with two arrays of actuators where N_f and N_s are the number of actuators in each array, the dynamics of each array $g_{s,f}(z)$ are given by a first order plus delay transfer function determined by the open loop bandwidth $a_{s,f}$ and delay $\tau_{s,f}$ [4], and without loss of generality, it is assumed that:

1. $g_s(1) = 1$ and $g_f(1) = 1$ i.e. the DC gains of the dynamics can be taken as unity.
2. the dynamics of $g_f(z)$ are faster than $g_s(z)$.
3. $N_s \leq N_f$.
4. all the actuators of a given array have the same dynamics.

The position measured at M BPMs is described by

$$Y(z) = g_s(z)R_sU_s(z) + g_f(z)R_fU_f(z) + D(z) \qquad (1)$$

where $U_s(z)$ and $U_f(z)$ represent the inputs applied to the two distinct arrays of actuators and $D(z)$ represent the disturbances acting on the electron beam. The response matrices for each array is represented by R_s and R_f where $N_s = \text{rank}(R_s)$ and $N_f = \text{rank}(R_f)$. The response matrices can each be expressed in terms of reduced singular value decompositions so that,

$$R_s = \Phi_s\Sigma_s\Psi_s^{\mathrm{T}}, \qquad R_f = \Phi_f\Sigma_f\Psi_f^{\mathrm{T}} \qquad (2)$$

where $\Phi_s \in \mathbb{R}^{M\times N_s}$, $\Sigma_s \in \mathbb{R}^{N_s\times N_s}$, $\Psi_s \in \mathbb{R}^{N_s\times N_s}$, $\Phi_f \in \mathbb{R}^{M\times N_f}$, $\Sigma_f \in \mathbb{R}^{N_f\times N_f}$ and $\Psi_f \in \mathbb{R}^{N_f\times N_f}$. The columns of Ψ_s and Ψ_f in Eq. 2 represent the controllable subspaces of the response matrices and even though R_s and R_f are independent, there may be some overlap between the controllable subspaces and the relationship between the subspaces of R_s and R_f can be found by comparing the subspaces of Φ_s and Φ_f which are both orthogonal and can be determined from Algorithm 12.4.3 in [5] such that

$$\Phi_f^{\mathrm{T}}\Phi_s = A \qquad (3)$$

and the singular value decomposition of A is given by

$$A = \Phi_A\Sigma_A\Psi_A^{\mathrm{T}} \qquad (4)$$

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

with left matrix, $\Phi_A \in \mathbb{R}^{N_f \times N_s}$, right matrix $\Psi_A \in \mathbb{R}^{N_s \times N_s}$ and $\Sigma_A \in \mathbb{R}^{N_s \times N_s}$ such that

$$\Sigma_A = \text{diag}\{\cos(\theta_{A_1}), \ldots, \cos(\theta_{A_{N_s}})\} \tag{5}$$

where θ_{A_i} is the angle between the principal vectors spanning Φ_f and Φ_s which span the controllable subspaces of R_s and R_f.

Internal Model Control Multi-array Design

An IMC structure for such a multi-array system results in a closed loop response given by

$$\begin{aligned} Y(z^{-1}) = \Big(I - g_s(z^{-1})R_s\tilde{Q}_s(z^{-1}) \\ - g_f(z^{-1})R_f\tilde{Q}_f(z^{-1})\Big)\, D(z) \end{aligned} \tag{6}$$

(for a zero setpoint) which can be expressed as,

$$Y(z) = \left(I - [\Phi_s \;\; \Phi_f] \begin{bmatrix} g_s(z)\Sigma_s\Psi_s^{\mathrm{T}}\tilde{Q}_s(z) \\ g_f(z)\Sigma_f\Psi_f^{\mathrm{T}}\tilde{Q}_f(z) \end{bmatrix}\right) D(z) \tag{7}$$

where

$$\begin{aligned} \tilde{Q}_s(z) &= \Psi_s\Sigma_s^{-1}Q_s(z)\tilde{\Phi}_s \\ \tilde{Q}_f(z) &= \Psi_f\Sigma_f^{-1}Q_f(z)\tilde{\Phi}_f \end{aligned} \tag{8}$$

for some diagonal $Q_s(z)$ and $Q_f(z)$ and for $\tilde{\Phi}_s \in \mathbb{R}^{N_s \times M}$ and $\tilde{\Phi}_f \in \mathbb{R}^{N_f \times M}$. Therefore Eq. 7 can be expressed as

$$Y(z) = \left(I - [\Phi_s \;\; \Phi_f] \begin{bmatrix} g_s(z)Q_s(z)\tilde{\Phi}_s \\ g_f(z)Q_f(z)\tilde{\Phi}_f \end{bmatrix}\right) D(z^{-1}) \tag{9}$$

and for steady state

$$Y_{ss} = \left(I - [\Phi_s \;\; \Phi_f] \begin{bmatrix} \tilde{\Phi}_s \\ \tilde{\Phi}_f \end{bmatrix}\right) D_{ss} \tag{10}$$

since the diagonal transfer function matrices $Q_f(z)$ and $Q_s(z)$ are chosen to have unity DC gain. Pre-multiplying both sides by $[\Phi_s \;\; \Phi_f]^{\mathrm{T}}$, projects the response into modal space so that

$$\begin{bmatrix} \Phi_s^{\mathrm{T}} \\ \Phi_f^{\mathrm{T}} \end{bmatrix} Y_{ss} = \begin{bmatrix} \Phi_s^{\mathrm{T}} \\ \Phi_f^{\mathrm{T}} \end{bmatrix} D_{ss} - \begin{bmatrix} \Phi_s^{\mathrm{T}} \\ \Phi_f^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \Phi_s & \Phi_f \end{bmatrix} \begin{bmatrix} \tilde{\Phi}_s \\ \tilde{\Phi}_f \end{bmatrix} D_{ss} \tag{11}$$

and

$$\begin{bmatrix} \tilde{\Phi}_s \\ \tilde{\Phi}_f \end{bmatrix} = K \begin{bmatrix} \Phi_s^{\mathrm{T}} \\ \Phi_f^{\mathrm{T}} \end{bmatrix} \tag{12}$$

is defined such that K is a decoupling matrix, with $K_{11} \in \mathbb{R}^{N_s \times N_s}$, $K_{12} \in \mathbb{R}^{N_s \times N_f}$, $K_{21} \in \mathbb{R}^{N_f \times N_s}$ and $K_{22} \in \mathbb{R}^{N_f \times N_f}$. From Eq. 11 and Eq. 12,

$$\bar{Y}_{ss} = \begin{bmatrix} \Phi_s^{\mathrm{T}} \\ \Phi_f^{\mathrm{T}} \end{bmatrix} Y_{ss}, \qquad \bar{D}_{ss} = \begin{bmatrix} \Phi_s^{\mathrm{T}} \\ \Phi_f^{\mathrm{T}} \end{bmatrix} D_{ss} \tag{13}$$

and

$$\bar{Y}_{ss} = \left(I - \begin{bmatrix} I & \Phi_s^{\mathrm{T}}\Phi_f \\ \Phi_f^T\Phi_s & I \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}\right) \bar{H}_{ss}. \tag{14}$$

From the relationship in Eq. 3, then

$$\begin{bmatrix} I & \Phi_s^T\Phi_f \\ \Phi_f^T\Phi_s & I \end{bmatrix} = \begin{bmatrix} I & A \\ A^{\mathrm{T}} & I \end{bmatrix} \tag{15}$$

and given Eq. 4

$$\begin{aligned} &\begin{bmatrix} I & \Phi_s^T\Phi_f \\ \Phi_f^T\Phi_s & I \end{bmatrix} = \\ &\begin{bmatrix} \Phi_A & \\ & \Psi_A \end{bmatrix} \left[\begin{array}{cc|c} I & 0 & \Sigma_A \\ 0 & I & 0 \\ \hline \Sigma_A & 0 & I \end{array}\right] \begin{bmatrix} \Phi_A & \\ & \Psi_A \end{bmatrix}^{\mathrm{T}}. \end{aligned} \tag{16}$$

The ideal design would be to make K equal to the inverse of the matrix in Eq. 16 and because it has a block diagonal structure, this gives

$$K = \begin{bmatrix} \Phi_A & \\ & \Psi_A \end{bmatrix} \tilde{K} \begin{bmatrix} \Phi_A & \\ & \Psi_A \end{bmatrix}^{\mathrm{T}} \tag{17}$$

where

$$\tilde{K} = \begin{bmatrix} \tilde{\Sigma}_A & 0 & \tilde{\Sigma}_A \\ 0 & I & 0 \\ \tilde{\Sigma}_A & 0 & \tilde{\Sigma}_A \end{bmatrix} \odot \begin{bmatrix} I & 0 & -\Sigma_A \\ 0 & I & 0 \\ -\Sigma_A & 0 & I \end{bmatrix} \tag{18}$$

and $\tilde{\Sigma}_A \in \mathbb{R}^{N_s \times N_s}$ is a diagonal matrix with elements

$$\left[\tilde{\Sigma}_A\right]_{ii} = \frac{1}{1 - \cos^2\theta_{A_i}} \tag{19}$$

and $\odot$ denotes element-wise multiplication. Difficulties with this choice arise when $\cos(\theta_{A_i}) \approx 1$ which occurs when the angle between the subspaces is small. Instead, heuristic choices for the elements of $\tilde{K}$ can be made, which are described below:

1. Define the following:

$$\begin{aligned} \tilde{K}_{11} &= \begin{bmatrix} \tilde{\Sigma}_A & 0 \\ 0 & I \end{bmatrix}, & \tilde{K}_{12} &= \begin{bmatrix} -\tilde{\Sigma}_A\Sigma_A \\ 0 \end{bmatrix} \\ \tilde{K}_{21} &= \begin{bmatrix} -\tilde{\Sigma}_A\Sigma_A & 0 \end{bmatrix}, & \tilde{K}_{22} &= \tilde{\Sigma}_A \end{aligned} \tag{20}$$

2. When $\cos(\theta_{A_i}) = 0$, the directions $\Phi_s(:,i)$ and $\Phi_f(:,i)$ are orthogonal so that the system is considered as a decoupled two-input, two-output system i.e. two SISO structures. The relationships in Eq. 20 give this automatically where the corresponding elements of $\tilde{K}_{11}$ and $\tilde{K}_{22}$ are set to 1 and $\tilde{K}_{12}$ and $\tilde{K}_{21}$ are set to 0 so that the two directions are controlled independently.
3. When $0 < \cos(\theta_{A_i}) < 1$, Eq. 20 automatically splits the effort between the two actuators.
4. When $\cos(\theta_{A_i})$ is close to 1, the directions $\Phi_s(:,i)$ and $\Phi_f(:,i)$ almost line up and choosing to control in one direction is appropriate. This is achieved by setting the corresponding elements of either $\tilde{K}_{11}$ or $\tilde{K}_{22}$ to 1 and the other to 0. In this case, the corresponding elements of $\tilde{K}_{12}$ and $\tilde{K}_{21}$ are set to 0.
5. When $\cos(\theta_{A_i}) = 1$, the directions $\Phi_s(:,i)$ and $\Phi_f(:,i)$ align, giving a TISO structure. In this case, mid-ranging control is appropriate due to the actuator characteristics and the corresponding diagonal elements of $\tilde{K}_{11}$ and $\tilde{K}_{22}$ are transfer functions designed using a mid-ranging control technique.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Table 1: Angles Between Controllable Subspaces of R_s and R_f

	$\cos(\theta_{A_i})$	
$i = 1$	0.9978	0.9982
$i = 2$	0.9965	0.9896
$i = 3$	0.9503	0.8814
$i = 4$	0.8533	0.4833
$i = 5$	0.4750	0
$i = 6$	0.2290	0
$i = 7$	0	0

SIMULATION STUDY

In this section a simulation study is presented for the storage ring, where two arrays of actuators with different dynamics are considered. For this study, one array is considered to contain 'slow' corrector magnets which are bandwidth limited to 10 Hz ($a_s = 2\pi \times 10$) and the second array uses 'fast' corrector magnets which have a larger bandwidth of 700 Hz ($a_f = 2\pi \times 700$) but are amplitude limited. In each case the delay is taken as $\tau_{s,f} = 700$ μs. In this case, number of slow correctors are $N_s = 7$ and the number of fast correctors are $N_f = 165$. The frequency responses for the two arrays are shown in Fig. 1.

Choice of Control Directions

The cosine of the angles between the first 7 columns of Φ_s and Φ_f are determined by Eq. 4 and are listed in Table 1. The following strategy is used for control:

- When $\cos(\theta_{A_i}) \approx 1$, the control directions almost line up, so mid-ranging control is used. From Table 1, this is the case for $i = \{1, 2, 3, 4\}$ horizontally and $i = \{1, 2, 3\}$ vertically.
- When $\cos(\theta_{A_i})$ is small, the control effort is split using $\tilde{K}$ in Eq. 20. From Table 1, this is the case for $i = \{5, 6\}$ horizontally and $i = 4$ vertically.
- When $\cos(\theta_{A_i}) = 0$, the directions are decoupled and SISO structures are used. For $i = 7$ horizontally and $i = \{5, 6, 7\}$ vertically both the fast and slow actuators are used.

Mid-ranging Controller Design

For the TISO problem, the output is expressed as

$$Y(z) = \Big(1 - g_s(z)q_{mr_s}(z) - g_f(z)q_{mr_f}(z)\Big) D(z)) + \Big(g_s(z)q_{p_s}(z)) + g_f(z)q_{p_f}(z)\Big) U_r(z) \quad (21)$$

where $g_s(z)$ and $g_f(z)$ are the slow and fast process models respectively and $q_{mr_s}(z)$ and $q_{mr_f}(z)$ are the associated IMC controllers. The mid-ranging objective is to use both inputs to control Y and return the fast input to its setpoint U_r. Pre-filters $q_{p_s}(z)$ and $q_{p_f}(z)$ are included to obtain a desired response from U_r to Y [2]. Therefore, the corresponding elements of $\tilde{K}_{11}$ and $\tilde{K}_{22}$ in Eq. 20 are $q_{mr_s}(z)/q_{s_i}(z)$ and $q_{mr_f}(z)/q_{f_i}(z)$ respectively. The control signals to the two actuators are

$$\begin{aligned} U_s(z) &= q_{mr_s}(z)D(z) + q_{p_s}(z)U_r(z) \\ U_f(z) &= q_{mr_f}(z)D(z) + q_{p_f}(z)U_r(z). \end{aligned} \quad (22)$$

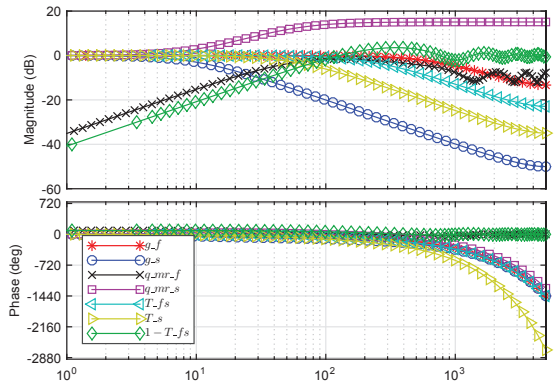


Figure 1: Frequency responses of fast ('∗' red) and slow actuators ('o' blue), fast ('x' black) and slow ('□' magenta) IMC controllers, fast and slow together complementary sensitivity ('◁' cyan), slow complementary sensitivity ('▷' yellow) and closed loop sensitivity ('◇' green) for the mid-ranging design.

The mid-ranging design specifies not only the complementary sensitivity with both actuators, $T_{fs}(z)$, but also the complementary sensitivity corresponding to the control action with the slow actuator alone, $T_s(z)$ which are defined as follows,

$$\begin{aligned} T_{fs}(z) &= g_s(z)q_{mr_s}(z) + g_f(z)q_{mr_f}(z) \\ T_s(z) &= g_s(z)q_{mr_s}(z). \end{aligned} \quad (23)$$

To achieve the control objectives, $T_{fs}(z)$ and $T_s(z)$ must be unity at steady state and the decoupling between the setpoint on the faster input, U_r and Y is achieved through the use of pre-filters which must satisfy the condition,

$$g_s(z)q_{p_s}(z) + g_f(z)q_{p_f}(z) = 0. \quad (24)$$

Because $g_f(z)$ and $g_s(z)$ both include delay terms, $T_s(z)$ is chosen as

$$T_s(z) = T_s^-(z)T_s^+(z) \quad (25)$$

where $T_s^+(z)$ includes the delays of both $g_f(z)$ and $g_s(z)$ so that $T_s(z)/g_f(z)$ and $T_s(z)/g_s(z)$ are both causal and stable. So in this case,

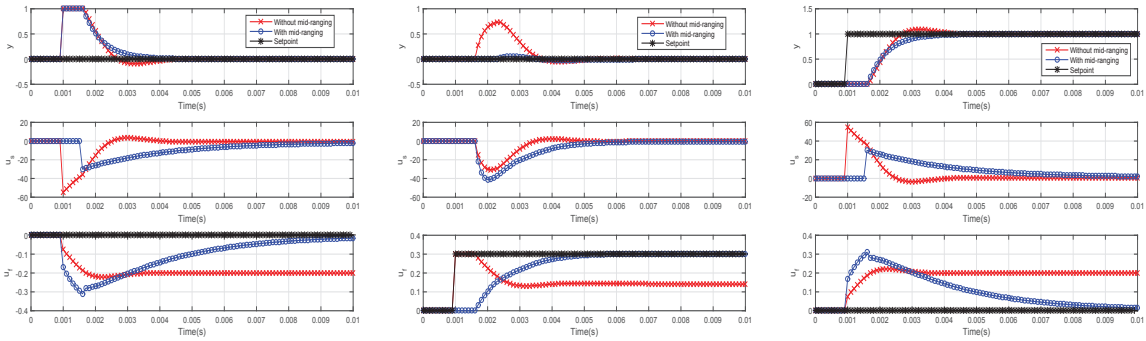
$$T_s(z) = z^{-(d_s+d_f)} \frac{1 - \lambda_s}{1 - \lambda_s z^{-1}} \quad (26)$$

and the controller for the slow array is given by

$$q_{mr_s}(z) = T_s(z^{-1})/g_s(z)^{-1} \quad (27)$$

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9



(a) Step in disturbance. (b) Step in fast actuator setpoint. (c) Step in output setpoint.

Figure 2: Responses of the output y and the slow u_s and fast u_f actuator inputs to various step changes when both actuators are used without mid-ranging control ('x' red) and with mid-ranging control ('o' blue). The reference for the output and fast actuator input are also shown ('$*$' black).

Likewise the fast and slow together sensitivity can be written as

$$T_{fs}(z) = T_{fs}^{-}(z)T_{fs}^{+}(z) \tag{28}$$

where $T_{fs}^{+}(z)$ includes the non-minimum phase components of $g_f(z)$ only so that $T_{fs}(z)/g_f(z)$ is causal and stable. The controller for the fast array is determined by

$$q_f(z) = \left[T_{fs}(z) - T_s(z)\right]/g_f(z) \tag{29}$$

Design of the pre-filters are described in [2] however a simple choice is

$$\begin{aligned} q_{p_f}(z) &= \tilde{q}_p(z)g_s^{+}(z) \\ q_{p_s}(z) &= -\tilde{q}_p(z)\frac{g_f(z)}{g_s^{-}(z)} \end{aligned} \tag{30}$$

where $\tilde{q}_p(z)g_s^{+}(z)|_{ss} = 1$ for some filter $\tilde{q}_p(z)$. The chosen complementary sensitivities are shown in Fig. 1 along with the closed loop sensitivities. The slow actuator has a control sensitivity that is low bandwidth only, while the fast actuator has a control sensitivity that is mid-frequency only and goes to zero at steady state; giving the mid-ranging effect. IMC is advantageous because it gives the control sensitivities of the fast and slow actuators directly as q_{mr_f} and q_{mr_s}.

In Fig. 2, the performance of the TISO system with and without mid-ranging is compared for a step change in disturbance, fast setpoint and output setpoint. Firstly, the effect of a step change in the disturbance is shown in Fig. 2a. The mid-ranging system firstly manipulates the fast actuator and moves the slow actuator to ensure that the fast actuator does not saturate by returning it to the setpoint $u_r = 0$. The non-mid-ranging system allows the fast input to settle to a steady state close to the saturation limit of ± 0.5 which on the next upset, the fast actuator may easily because saturated. The responses to a step change in the fast actuator setpoint are shown in Fig. 2b and as before, the fast actuator achieves the requested setpoint change using the mid-ranging approach. Also, the effect in the output is decoupled through the pre-filters so that changes to the fast actuator setpoint are minimised on the output. Fig. 2c shows the responses to a change in the output setpoint. Both approaches achieve the required setpoint change, however the system without mid-ranging uses more control effort from the slower actuator. Mid-ranging control is therefore used to obtain a desired response from $Y(z)$ to $D(z)$ and from $U_r(z)$ to $U_f(z)$ and a decoupled response between $U_f(z)$ to $Y(z)$.

CONCLUSION

For a storage ring with two arrays of actuators used for electron orbit control, the overlap of the controllable subspaces of each array can be used to reduce the problem into either SISO, TISO or decoupled TITO blocks. The TISO blocks represent the modes of the system where both fast and slow actuators are required for control. For such cases, mid-ranging control is appropriate given the distinct characteristics of the two arrays of actuators and provides a control strategy that uses both actuators in such a way as to meet performance specifications without causing the faster actuator to saturate.

REFERENCES

[1] B. Allison and S. Ogawa, "Design and tuning of valve position controllers with industrial applications," *Transactions of the Institute of Measurement and Control*, vol. 25, pp. 3–16, 2003.

[2] S. Gayadeen and W. P. Heath, "An Internal Model Control approach to mid-ranging control," in *IFAC International Symposium on Advanced Control of Chemical Processes*, (Istanbul, July), pp. 542–547, 2009.

[3] J. Rowland, M. G. Abbott, J. A. Dobbing, M. T. Heron, I. Martin, G. Rehm, and I. Uzun, "Status of the Diamond fast orbit feedback system," in *International Conference on Accelerator and Large Experimental Physics Control Systems*, (Knoxville, Tennessee), pp. 535–537, 2007.

[4] S. Gayadeen, *Synchrotron Electron Beam Control*. PhD thesis, Department of Engineering Science, University of Oxford, UK, 2014.

[5] G. H. Golub and C. F. Van Loan, *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

CONTROL SYSTEM CHALLENGES FROM AN UPGRADE TO THE DIAMOND LIGHT SOURCE STORAGE RING

M. T. Heron, A. Rose, Presented on behalf of the DDBA control and instrumentation team,
Diamond Light Source, Oxfordshire, UK.

Abstract

In 2016 Diamond Light Source will replace one double bend achromatic cell of the storage ring (SR) with two double bend achromatic cells in the same longitudinal space. This will create an additional straight section for an insertion device (ID), thereby converting a bending magnet source point into an ID source point. Installation of the two new cells and recommissioning of the SR will take place in an eight week shutdown, with an expectation that post-shutdown operation will resume with a level of performance comparable to that pre-installation. The additional components in the two new cells necessitate a substantial reworking of the interface layer of control system, together with changes to all applications which are dependent on the physics parameters of the storage ring. This paper will describe how the control system aspects of this project will be managed.

INTRODUCTION

Diamond Light Source is part way through a project to replace one of the standard Double Bend Achromat (DBA) cells of the 561 metre circumference electron storage ring with a new design consisting two DBA cells and a straight section for an insertion device[1]. This arrangement of the two DBA cells has been termed the "Double-Double Bend Achromat" (DDBA) cell. The new insertion device associated with the DDBA cell will service a new photon beamline which otherwise would have had a bending magnet as its source.

The project management aspects of the DDBA project, from a control and instrumentation point of view, are of relevance to future projects at Diamond and other light sources. In the case of Diamond there are presently accelerator physics studies exploring whether a second DDBA can been installed in 2018, together with a conceptual design study considering how the whole storage ring could be rebuilt with a more modern lattice design to reduce the natural emittance. For other light sources, a number of facilities have plans to rebuild their storage rings with more modern lattice designs, again to reduce the natural emittance. As operational facilities, all these projects face the same challenge as the DDBA project in how to make such an invasive change to an operational facility while minimising the impact on user time and maintaining operational performance.

DDBA PROJECT

The existing DBA cell consists of two dipoles and ten quadrupole and seven sextupole magnets to bend and focus the electron beam, which are mounted on three girders. With the DDBA design, each half of the DDBA cell is mounted on one girder including five quadrupoles, five sextupoles and the two dipole magnets. Between the two girders is a new straight section that is used to house the new insertion device, as shown in Fig 1.

To fit a DDBA cell into the longitudinal space of a standard DBA cell necessitates higher field gradient magnets, and so requires smaller apertures. It also necessitates a greater number of magnets, including the addition of trim windings on the dipoles, to enable the existing dipole supply to be used for the main ampere turns and to be able to match the integrated field to that of the standard dipoles. While there are adequate numbers of corrector magnets for orbit correction on the sextupoles, a portion are located over copper vessels, which would significantly impact their dynamic performance in the global Fast Orbit Feedback (FOFB). This is addressed through additional separate correctors over stainless steel portions of the vacuum vessel. The ratings of the new quadrupoles are similar to the old, but the new sextupoles have lower impedance, so in both cases the existing power supplies can be used, albeit with a lower operating voltage for the sextupoles. There is one additional beam position monitor, hence one additional Libera Electron. For the vacuum vessel the use of lumped NEG pumps means fewer ion pumps, but a comparable number of vacuum gauges and additional vacuum valves. For vessel thermal protection there are comparable additional thermal monitors, thermal interlocks and water flows but in different locations. These differences in instrumentation are summarised in Table 1.

ELECTRICAL AND INSTRUMENTATION

From Table 1, the numbers of each component type in the DDBA cell are not significantly different from a standard DBA cell. However, for the majority of electrical circuits it was concluded that it would be better (quicker and operationally more reliable) if new cables were installed to connect the girder back to the instrumentation located outside the ring in the Control Instrumentation Area (CIA). This largely came about because one of the three cable penetrations would not be available after the changes, meaning that a third of the cables would need rerouting, and it was not evident there was sufficient cable length to reassign cables to their new locations on the DDBA cell. Despite the small overall change in numbers of components to control, the new instrumentation necessitated additional racks, which exceed the capacity of the CIA and so necessitate it being extended.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

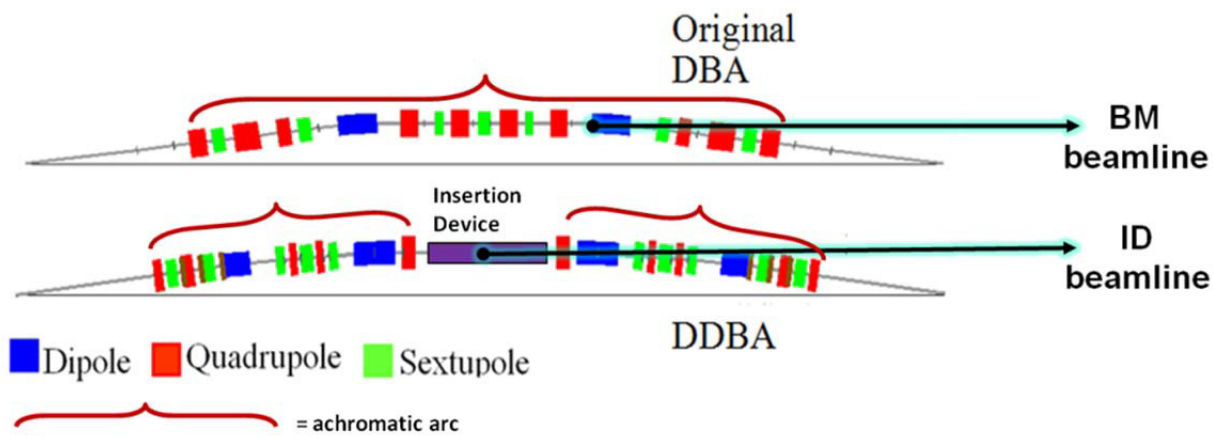


Figure 1: Changes from a DBA to DDBA cell.

For reasons outlined under the section on Project Management, components on the new girders will be pre-cabled to termination boxes and the rack-to-girder cables terminated with plugs to have a single interface point for each girder, thereby facilitating the installation, termination and testing of cables before the installation shutdown.

Table 1: DBA and DDBA, Number of Components

Components	DBA Cell	DDBA Cell
Dipoles (Main, Trim)	2,0	4,4
Quadrupoles	10	10
Sextupoles	7	10
Corrector Magnets on Sextupoles (H, V, Skew Quad)	7,7,4	10,10,4
Separate Corrector Magnets (H, V)	0	2,2
Beam Position Monitors	7	8
Vacuum Ion Pumps	18	7
Vacuum Gauges	7	11
Vacuum Valves	4	7
Vessel Temperatures	42	49
Vessel Temperature Interlocks	12	14
Vessel Water Flow Interlocks	16	14

CONTROL SYSTEM DEVELOPMENTS

At Diamond the EPICS based control system[2] is partitioned by geographical location and by technical system. Hence there are separate IOCs for each of the technical subsystems associated with a cell. IOCs for MPS, Corrector Power Supplies, Diagnostics, Quadrupoles and Sextupoles and Vacuum all require significant amounts of development. For each of these, versions of these IOCs are also being built with soft records to enable simulation of actual Process Variables (PVs) without the hardware, thereby enabling development and testing of overall summary views and global applications, such as sequencing start-up and shutdown of systems.

Protection of equipment is accomplished by standard solutions based on PLCs. For vacuum protection an additional vacuum valve control crate will be added to control the new valves and manage the interlocks to protect them and the ion pumps. For vessel protection a new PLC solution has been designed, where all temperatures that provide interlocks are hardwired back to the PLCs located in the CIA, while temperatures used for monitoring only are connected using remote analogue input modules. This differentiation comes from evidence of long term radiation damage to electronics located in the vault.

Diamond uses a number of the high-level applications, including Matlab Middle Layer, LOCO, and various bespoke applications for routine operation. In addition there is extensive use made of Matlab scripts for various physics studies. The scripts are large, uncontrolled and maintained by the user, so responsibility of updating them sits with the user. For the other applications, these are being updated for the new lattice configuration during late 2015 and early 2016. To enable a degree of early testing a Virtual Accelerator (VA) has already been built with the new lattice to present the same PV interface as the real machine, but on a different port. This will be used to test the applications prior to the machine becoming available. However, final testing will only be possible as part of commissioning with beam.

Routine operation of Diamond runs seven global real-time feedback and feedforward processes[3] to control beam position, vertical beam size and tune. All of these use model base information and, with the exception of the FOFB, run relatively slowly, i.e. sub one second sample rate, and communicate over EPICS PVs. They will be updated and tested, as much as possible, against the VA and soft IOC version of the new IOCs. Nevertheless, final testing will only be possible in installation shutdown and as part of commissioning with beam. For FOFB, the additional BPM was added to the low latency FOFB

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Figure 2: Temporary cable installation.

network in early 2015, and the addition PSUs will be added in early 2016, but again final testing will only be possible in installation shutdown and as part of commissioning with beam. The TMBF should have limited lattice dependence but will need recommissioning.

PROJECT MANAGEMENT

The DDBA project plan is divided into two sections: Phase 1 covers work up until the installation shutdown and Phase 2 the installation shutdown. The critical path of the former was determined by the lead times for the design and manufacturer of magnets and vessels. This provides adequate time to build new control racks and implement the control system software changes. However in the initial planning of Phase 2, the installation shutdown, the critical path of fourteen weeks was dominated by the time to remove and re-install the electrical cables and recommissioning systems. A shutdown of this length was deemed operationally unacceptable, so options to shorten the shutdown were explored. The chosen solution was to install and terminate the new cables before the shutdown and to have a single termination panel for all cables on each of the new girders. This would allow girders cabling and rack-to-girder cables to be pretested and pluggable, to minimise shutdown cabling time. However, to install the new cables necessitated clearing the cable trenches of the old cables and this required the existing DBA girders be re-cabled back to the instrumentation racks on temporary cables, see Fig. 2. This process of switching to temporary cable could only happen during shutdowns and so was managed as part of Phase 1 of the project planning taking place during 2014 to 2016.

STATUS AS OF AUTUMN 2015.

As of autumn 2015 the major contracts are well underway for magnets and vessels, with girders already delivered and a new temperature stabilised alignment room constructed. The enabling work to install and switch to temporary cable took place in the first half of 2015 with no impact on operations. Then the original cables were removed, and in the summer 2015 shutdown the dipole circuit was rerouted. Designs for new instrumentation systems are being reviewed, and new instrumentation racks are being constructed. The additional BPM was installed and added to the FOFB network. The extension to the CIA was completed in early 2015.

The VA has been built, and work is now commencing on updating the physics applications. This is in line with the Phase 1 plan. The planned installation and commissioning shutdown is set for August and September 2016. The Phase 2 plan has been detailed, and it is clear that installation and subsequent recommissioning of the storage ring will be challenging. Given the established user community it remains critical that routine operation is re-established as planned, and in addition that the user community are briefed on the expected level of operational performance (beam properties and reliability) on start-up, to help manage their expectations.

CONCLUSION

In terms of control systems, replacing one cell of storage ring is not as challenging as replacing the whole ring. However, it still poses a risk to an established operational facility, both in terms of the time off-the-air and re-establishing routine operations at a level of performance and reliability expected by an established user community. From the DDBA project it has become evident that the mechanical engineering aspects lend themselves to being prepared off-line, and it is the electrical and control aspects that naturally dominate the installation and recommissioning time. However, we have shown that through upfront enabling work, the time and risks associated with control system related tasks during the upgrade can be minimised.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REFERENCES

[1] R.P. Walker et al., "The Double-Double Bend Achromat (DDBA) Lattice Modification for the Diamond Storage Ring", IPAC'14, Dresden.
[2] M.T. Heron, et al., "The Diamond Light Source Control System", EPAC06, Edinburgh.
[3] M.T. Heron, et al., Feed-forward and Feedback Schemes Applied to The Diamond light Source Storage Ring, IPAC14, Dresden.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ROBUST STABILITY ANALYSIS OF ORBIT FEEDBACK CONTROLLERS

S. Gayadeen, M. T. Heron, G. Rehm, Diamond Light Source, Didcot, UK

Abstract

Closed loop stability of electron orbit feedback controllers is affected by mismatches between the accelerator model and the real machine. In this paper, the small gain theorem is used to express analytical criteria for closed loop stability in the presence of spatial uncertainty. It is also demonstrated how the structure of the uncertainty models affects the conservativeness of the robust stability results. The robust stability criteria are applied to the Diamond electron orbit controller and bounds on the allowable size of spatial uncertainties which guarantee closed loop stability is determined.

INTRODUCTION

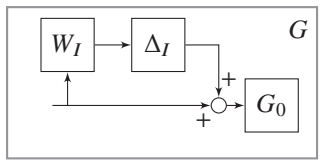
For electron orbit control, the nominal or "golden" response matrix is used to design the controller gains. The control system is referred to as "robust" if it is insensitive to differences between the actual system and the model of the system which is used to design the controller [1]. These differences are referred to as model-plant mismatch or simply uncertainty. The general approach for robust control design is to find a representation of the model uncertainty and then check for robust stability i.e. determine whether the system remains stable for all "real" processes within the uncertainty set. The following notation is adopted: Π is a set of possible perturbed plant models, $G_0(z)$ is the nominal plant model which belongs to the set Π and $G(z)$ is the perturbed model (representing the real plant) which also belongs to the set Π. A norm-bounded uncertainty description is used to describe the uncertainty i.e. the set Π is generated by allowing H_∞ norm bounded perturbations in the nominal plant i.e. Δ is a normalised perturbation with H_∞ norm ≤ 1 and the H_∞-norm of a discrete operator $M(z^{-1})$ is defined as

$$\|M\|_\infty := \max_{\omega\in[-\pi,\pi]} \sigma_{max}(G(e^{j\omega})) \qquad (1)$$

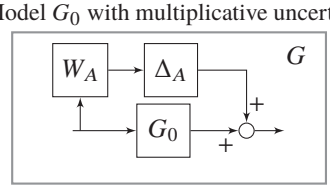
where M is linear, time-invariant and $\sigma_{max}(\cdot)$ indicates the largest singular value of the matrix.

Uncertainty can be classed as either parametric or unmodelled dynamics. Parametric uncertainty is where the model structure is known but some parameters are unknown or uncertain, whereas unmodelled dynamics refers to the case where dynamics have been neglected or missing. In this paper, "lumped" uncertainty is considered which represents one of several sources of parametric uncertainty and/or unmodelled dynamics. This type of uncertainty is usually represented as multiplicative uncertainty [1] as shown in Fig. 1a. In the case of multiplicative uncertainty,

$$\Pi_I : G(z) = G_0(z)(1 + W_I(z)\Delta_I(z)); \quad \|\Delta\|_\infty \leq 1 \forall\omega \quad (2)$$



(a) Model G_0 with multiplicative uncertainty.



(b) Model G_0 with additive uncertainty.

Figure 1: Uncertainty representations of real process G.

where the subscript I indicates "input" uncertainty. In the case of additive uncertainty,

$$\Pi_A : G(z) = G_0(z) + W_A(z)\Delta_A(z); \quad \|\Delta\|_\infty \leq 1 \forall\omega \quad (3)$$

which is shown in Fig. 1b. In each case, a weight $W(z)$ is introduced in order to normalise the perturbation to be less than 1 in magnitude at each frequency and to obtain the weight:

- for multiplicative uncertainty:

$$|W_I(e^{j\omega})| \geq \max\left|\frac{G(e^{j\omega}) - G_0(e^{j\omega})}{G_0(e^{j\omega})}\right| \quad \forall\omega \qquad (4)$$

- for additive uncertainty:

$$|W_A(e^{j\omega})| \geq \max|G(e^{j\omega}) - G_0(e^{j\omega})| \quad \forall\omega. \qquad (5)$$

The procedure for robust stability test is as follows:

1. The closed loop system is represented in the structure shown in Fig. 2 where Δ represents the uncertainty in the system and M is the transfer function matrix describing the closed loop "as seen by" Δ.

2. The **small gain theorem** for systems represented as Fig. 2, is applied. The theorem takes the form [1]:

   Theorem 1 (Robust stability) *Assuming that the nominal system $M(e^{j\omega})$ is stable and that the perturbations Δ are stable, then the interconnected $M - \Delta$ system in Fig. 2 is stable for all perturbations Δ, satisfying $\|\Delta\|_\infty \leq 1$ (i.e. the system is robustly stable) if and only if*

$$\|M\|_\infty < 1. \qquad (6)$$

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

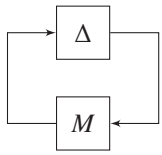


Figure 2: M-Δ structure for robust stability analysis.

In this paper, the robust stability test is applied to the closed loop for the Diamond electron orbit feedback system in the presence of uncertainty in the response matrix. Firstly, it is demonstrated how to apply the robust stability test to the system using the Singular Value Decomposition (SVD) of the response matrix and secondly the Fourier decomposition of the response matrix is used for the robust stability test.

ROBUST STABILITY ANALYSIS: SVD METHOD

Additive Uncertainty in the Response Matrix

Consider an additive uncertainty on the response matrix where the uncertainty is calculated from the difference between the golden and measured response matrices, such that

$$R = R_0 + W_R \Delta_R \tag{7}$$

and the weight W_R is introduced so that $\|\Delta_R\|_\infty \leq 1$ and found using Eq. 5. The perturbed closed loop system using an Internal Model Control (IMC) structure is represented in Fig. 3 where

$$\begin{aligned} U(z) &= -q(z)W_R\Psi D\Sigma^{-1}\left[g(z)\Phi^{\mathrm{T}}V(z)\right. \\ &\quad \left.+g(z)\Phi^{\mathrm{T}}RW_R^{-1}U(z) - g(z)\Sigma\Psi^{\mathrm{T}}W^{-1}U(z)\right] \\ U(z) &= -g(z)q(z)W_R\Psi D\Sigma^{-1}\Phi^{\mathrm{T}}V(z). \end{aligned} \tag{8}$$

Therefore the system "seen by" the uncertainty is the transfer matrix

$$M_R = -q(z)g(z)W_R\Psi D\Sigma^{-1}\Phi^{\mathrm{T}} \tag{9}$$

and for robust stability

$$\gamma_R = \|M_R\|_\infty \leq 1, \quad \forall\omega. \tag{10}$$

Multiplicative Uncertainty in Singular Values Σ_0

A multiplicative uncertainty, Δ_Σ in the singular values can be represented as

$$\begin{aligned} \Sigma &= \Sigma_0\,(\Delta_\Sigma + I) \\ \Delta_\Sigma &= \Sigma_0^{-1}\Sigma - I \end{aligned} \tag{11}$$

where Σ_0 is derived from the SVD of the golden response matrix and Σ is derived from the SVD of the measured response matrix. In this case, the uncertainty is diagonal i.e.

$$\Delta_\Sigma = \mathrm{diag}\{\delta_\Sigma\}, \quad |\delta_{\sigma_i}| \leq 1 \quad \forall i. \tag{12}$$

The closed loop system with Δ_Σ is shown in Fig. 4 where

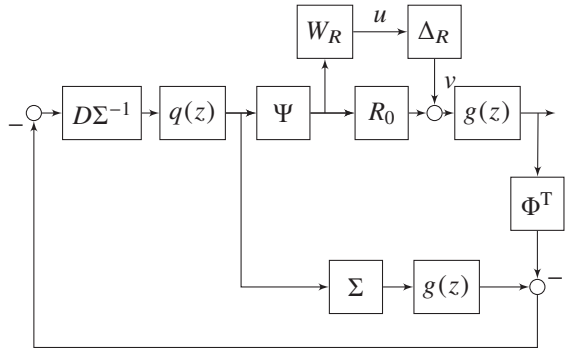


Figure 3: Additive uncertainty in R_0.

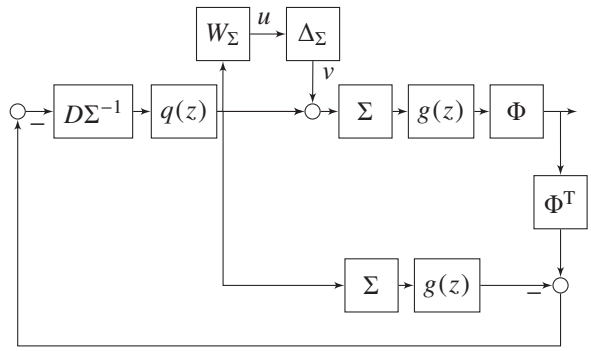


Figure 4: Multiplicative uncertainty in Σ_0.

$$\begin{aligned} U(z) = -W_\Sigma q(z) D\Sigma^{-1}\left[g(z)V(z) + g(z)\Sigma W_\Sigma^{-1}U(z) - \right. \\ \left. +g(z)\Sigma W_\Sigma^{-1}U(z)\right] \end{aligned} \tag{13}$$

so that

$$M_\Sigma = -q(z)g(z)W_\Sigma D\Sigma^{-1}. \tag{14}$$

Using the small gain theorem, for robust stability,

$$\gamma_\Sigma = \|M_\Sigma\|_\infty \leq 1 \quad \forall\omega. \tag{15}$$

Multiplicative Uncertainty in Singular Vectors Φ_0 and Ψ_0

If the uncertainty is considered to be in the left singular vectors, such that

$$\begin{aligned} \Phi &= (\Delta_\Phi + I)\,\Phi_0 \\ \Delta_\Phi &= \Phi\Phi_0^{\mathrm{T}} - I \end{aligned} \tag{16}$$

then given the structure in Fig. 5, the transfer matrix of the linear plant is

$$M_\Phi = -q(z)g(z)W_\Phi\Phi_0 D\Phi^{\mathrm{T}} \tag{17}$$

and using the small gain theorem

$$\gamma_\Phi = \|M_\Phi\|_\infty \leq 1 \quad \forall\omega \tag{18}$$

for robust stability.

Likewise, perturbed right singular vectors can be represented as

$$\begin{aligned} \Psi &= \Psi_0\,(\Delta_\Psi + I) \\ \Delta_\Psi &= \Psi_0\Psi^{\mathrm{T}} - I \end{aligned} \tag{19}$$

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

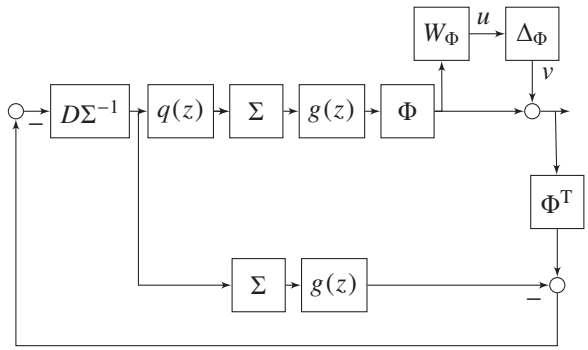


Figure 5: Multiplicative uncertainty in Φ_0.

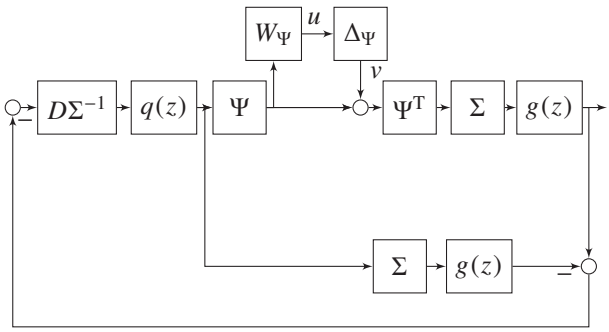


Figure 6: Multiplicative uncertainty in Ψ_0.

and given the structure in Fig. 6

$$M_\Psi = -q(z)g(z)W_\Psi\Psi_0 D\Psi^T \tag{20}$$

and robust stability

$$\gamma_\Psi = \|M_\Psi\|_\infty \leq 1 \quad \forall\omega. \tag{21}$$

Robust Stability Test Results

Fig. 7 shows the value of γ_R, γ_Σ, γ_Φ and γ_Ψ up to 5 kHz, where for each test, the bound is less than 1 and therefore the closed loop system is stable. The results can also be used to give a margin of stability. For additive uncertainty in the response matrix, the peak value over the range of frequencies is $\gamma_{R_{max}} = 0.2332$ occurring at DC. This means that the closed loop system is more sensitive to uncertainty at low frequencies. Given that

$$\|\Delta_R\|_\infty \|M_R\|_\infty < 1 \tag{22}$$

and $\gamma_{R_{\max}} = 0.2332$, for robust stability

$$\|\Delta_R\|_\infty < 1/\gamma_{R_{\max}}. \tag{23}$$

This means that the uncertainty may increase by a factor of 4.3 before the worst-case uncertainty yields instability. Using the triangle inequality

$$\begin{aligned} \|R\|_\infty &\leq \|R_0\|_\infty + \|\Delta_R\|_\infty \\ \|R\|_\infty - \|R_0\|_\infty &\leq \frac{1}{\gamma_{R_{\max}}} \end{aligned} \tag{24}$$

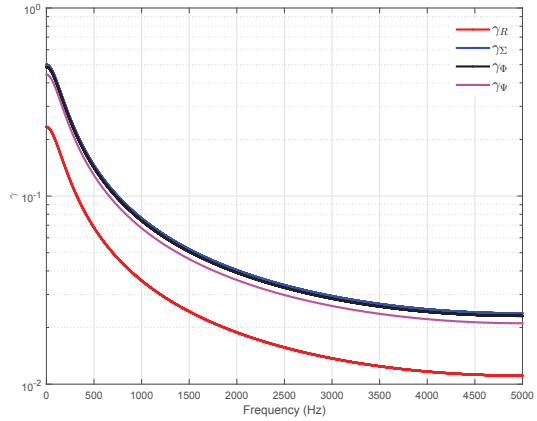


Figure 7: Size of γ_R (red), γ_Σ (blue), γ_Φ (black) and γ_Ψ (magenta) across frequencies up to 5 kHz using the singular value decomposition of the response matrix.

which gives an analytical bound on the variation of the response matrix from the nominal.

From Fig. 7, γ_Σ is also at a maximum at DC where the peak value is at $\gamma_{\Sigma_{max}} = 0.4999$ which means that the uncertainty in the singular values can increase by a factor of 2 for the closed loop system to become unstable. As before, a bound can be placed on the size of the uncertainty, i.e.

$$\|\Delta_\Sigma\|_\infty \leq 1/\gamma_{\Sigma_{max}}. \tag{25}$$

The uncertainty, Δ_Σ, is diagonal, so a bound on each singular value can be determined by observing that, for Eq. 25 to hold,

$$|\delta_{\Sigma_i}| \leq 1/\gamma_{\Sigma_{max}}. \tag{26}$$

Since

$$\delta_{\Sigma_i} = \sigma_{0_i}^{-1}\sigma_i - 1 \tag{27}$$

then the bound on each singular value to ensure robust stability can be expressed as

$$\sigma_i \leq \frac{\sigma_{0_i} + \sigma_{0_i}\gamma_{\Sigma_{max}}}{\gamma_{\Sigma_{max}}}. \tag{28}$$

From Fig. 7, the maximum bound on M_Φ and M_Ψ occurs at low frequencies and is $\gamma_{\Phi_{max}} = 0.4844$ and $\gamma_{\Psi_{max}} = 0.4431$ respectively. This means that the uncertainties Δ_Φ and Δ_Ψ can increase by a factor of 2.1 and 2.3 respectively before the worst case uncertainty yields instability. The uncertainties in Φ_0 and Ψ_0 are not diagonal, therefore a bound cannot be placed on the individual elements on the matrices Φ_0 and Ψ_0, but only placed on the matrix norm. From Eq. 16, a bound on Φ_0 is determined as

$$\begin{aligned} \|\Phi\|_\infty &\leq \|\Delta_\Phi\Phi_0\|_\infty + \|\Phi_0\|_\infty \\ \|\Phi\|_\infty &\leq \|\Delta_\Phi\|_\infty\|\Phi_0\|_\infty + \|\Phi_0\|_\infty \\ \frac{\|\Phi\|_\infty - \|\Phi_0\|_\infty}{\|\Phi_0\|_\infty} &\leq \|\Delta_\Phi\|_\infty \\ \frac{\|\Phi\|_\infty - \|\Phi_0\|_\infty}{\|\Phi_0\|_\infty} &\leq \frac{1}{\gamma_{\Phi_{max}}} \end{aligned} \tag{29}$$

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

and likewise

$$\frac{\|\Psi\|_\infty - \|\Psi_0\|_\infty}{\|\Psi_0\|_\infty} \leq \frac{1}{\gamma_{\Psi_{max}}} \tag{30}$$

gives a bound on Ψ_0.

ROBUST STABILITY ANALYSIS: FOURIER METHOD

In [2] a harmonic decomposition of the response matrix is presented, where R is written as

$$R_0 = \hat{\Phi}_0 \hat{\Sigma}_0 \hat{\Psi}_0^{-\mathrm{T}} \tag{31}$$

where

$$\begin{aligned} \hat{\Sigma}_0 &= \mathrm{diag}_{f=-F,\ldots,F}\{\hat{\sigma}_f\} \\ \hat{\Phi}_0 &= \mathrm{diag}_{m=1,\ldots,M}\{\sqrt{\beta_m}\}\mathrm{Re}\left[\mathrm{e}^{if\tilde{\eta}_m}\right]_{mf} \\ \hat{\Psi}_0 &= \mathrm{diag}_{n=1,\ldots,N}\{\sqrt{\beta_n}\}\mathrm{Re}\left[\mathrm{e}^{-if\tilde{\eta}_n}\right]_{nf} \end{aligned} \tag{32}$$

and $\hat{\Phi}_0$ is determined by the beta function β_m and normalised phase advance η_m at BPM locations and $\hat{\Psi}_0$ is determined by the beta function β_n and normalised phase advance η_n at corrector locations and $\hat{\Sigma}_0$ is determined by the Fourier coefficients which depend on the tune ν such that

$$\hat{\sigma}_f = \frac{1}{2\pi}\frac{2\nu}{\nu^2 - f^2} \quad (f = 0, 1, 2, \ldots). \tag{33}$$

Similar to the treatment using SVD, uncertainties can be included in the matrices of the harmonic decomposition where

$$\begin{aligned} \hat{\Sigma} &= \hat{\Sigma}_0\left(\Delta_{\hat{\Sigma}} + I\right) \\ \hat{\Phi} &= \left(\Delta_{\hat{\Phi}} + I\right)\hat{\Phi}_0 \\ \hat{\Psi} &= \hat{\Psi}_0\left(\Delta_{\hat{\Psi}} + I\right) \end{aligned} \tag{34}$$

and the associated transfer matrix M for each case is as described in Eq. 14, Eq. 17 and Eq. 20 but with the harmonic matrix counterpart to the SVD matrices i.e.

$$\begin{aligned} M_R &= -q(z)g(z)W_R\Psi\hat{D}\hat{\Sigma}^{-1}\hat{\Phi}^{\mathrm{T}} \\ M_{\hat{\Sigma}} &= -q(z)g(z)W_{\hat{\Sigma}}\hat{D}\hat{\Sigma}^{-1} \\ M_{\hat{\Phi}} &= -q(z)g(z)W_{\hat{\Phi}}\hat{\Phi}_0\hat{D}\hat{\Phi}^{\mathrm{T}} \\ M_{\hat{\Psi}} &= -q(z)g(z)W_{\hat{\Psi}}\hat{\Psi}_0\hat{D}\hat{\Psi}^{\mathrm{T}}. \end{aligned} \tag{35}$$

Fig. 8 shows the sizes of γ_R, $\gamma_{\hat{\Sigma}}$, $\gamma_{\hat{\Phi}}$ and $\gamma_{\hat{\Psi}}$ up to 5kHz, and for each case, the system is determined to be robustly stable for all frequencies. The peak value of $\gamma_{\hat{\Sigma}}$ is 0.3732 which means that the norm of the closed loop transfer matrix $M_{\hat{\Sigma}}$ can increase by a factor of 2.7 before the system becomes unstable. However this uncertainty in Fourier coefficients can also give a bound on the tune. In [2] the relationship between Fourier coefficients and the tune is given as

$$\Delta_{\hat{\Sigma}} = W_{\hat{\Sigma}}\Delta_\nu. \tag{36}$$

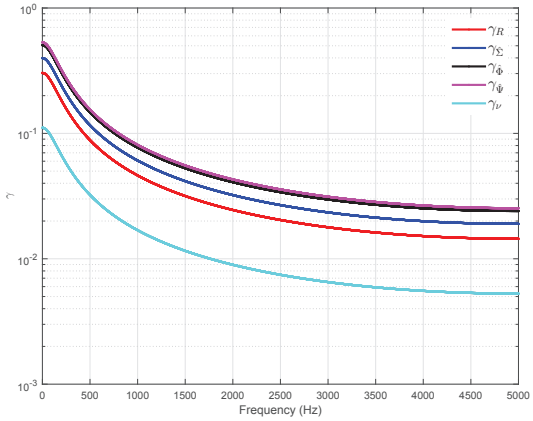


Figure 8: Size of γ_R (red), $\gamma_{\hat{\Sigma}}$ (blue), $\gamma_{\hat{\Phi}}$ (black), $\gamma_{\hat{\Psi}}$ (magenta) and γ_ν (cyan) across frequencies up to 5 kHz using the harmonic decomposition of the response matrix.

Since $\|\Delta_{\hat{\Sigma}}\|_\infty \leq 1/\gamma_{max_{\hat{\Sigma}}}$ then

$$\|\Delta_\nu\|_\infty = \frac{1}{\gamma_{max_{\hat{\Sigma}}}}\frac{1}{\|W_{\hat{\Sigma}}\|_\infty} \tag{37}$$

which is the maximum allowable size of uncertainty in the tune for the closed loop system to remain stable. The size of γ_ν is also shown in Fig. 8 and from the peak value, M_ν can increase by a factor of 9. This results corresponds to a tune change of 0.2 %, which means that for very small tune drifts, the closed loop system is guaranteed stable. The results are however conservative as it is assumed that the only source of uncertainty in the response matrix is from tube drift.

CONCLUSION

The small gain theorem was used to express analytical criteria for closed loop stability in the presence of several sources of spatial uncertainty. The robust stability test applied to the SVD matrices is useful for determining the closed loop stability when the controller is designed using SVD. However in order to determine bounds on beam parameters, the Fourier decomposition approach is better suited.

REFERENCES

[1] S. Skogestad and I. Postlethwaite, *Multivariable feedback control - analysis and design (2nd edition)*. Wiley, 2005.

[2] S. Gayadeen, M. T. Heron, and G. Rehm, "Uncertainty modelling of response matrix," in *MOPGF178, these proceedings, ICALEPCS'2015*, (Melbourne, Australia), 2015.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

UNCERTAINTY MODELLING OF RESPONSE MATRIX

S. Gayadeen, M. T. Heron, G. Rehm, Diamond Light Source, Didcot, UK

Abstract

Electron orbit feedback controllers are based on the inversion of the response matrix of the storage ring and as a result, mismatches between the accelerator model and the real machine can limit controller performance or cause the controller to become unstable. In order to perform stability analysis tests of the controller, accurate uncertainty descriptions are required. In this paper, BPM scaling errors, actuator scaling errors and drifts in tune are considered as the main sources of spatial uncertainties and because most electron orbit feedback systems use Singular Value Decomposition (SVD) to decouple the inputs and outputs of the system, the uncertainty can be expressed in terms of this decomposition. However SVD does not allow the main sources of uncertainty to be decoupled so instead, a Fourier-based decomposition of the response matrix is used to decouple and model the uncertainties. In this paper, both Fourier and SVD uncertainty modelling methods are applied to the Diamond Light Source storage ring and compared.

INTRODUCTION

The response matrix, $R \in \mathbb{R}^{M\times N}$ is the steady state (DC) response of the correctors to a change in the transverse orbit position measured at the mth beam position monitors (BPMs) due to a transverse kick at the nth corrector. For correction of the electron orbit, a map from BPM to corrector is required and therefore the inverse of the nominal (or "golden") response matrix is required. The inverse is normally computed using a Singular Value Decomposition (SVD) approach, such that

$$\begin{aligned} R_o &= \Phi_o \Sigma_o \Psi_o^{\mathrm{T}} \\ R_o^{-1} &= \Psi_o \Sigma_o^{-1} \Phi_o^{\mathrm{T}}. \end{aligned} \tag{1}$$

However because of BPM and corrector scaling errors from imperfect calibrations or drifts in the tune, the actual response of the machine may differ over time from the ideal response. This means that the control action calculated from the ideal response may be either too aggressive or too conservative. In the case of such model mismatch, applying the control action may result in closed loop instabilities or even if the closed loop remains stable, steady state performance will degrade. By modelling the various sources of the uncertainties, the mismatch between the machine response and the ideal response can be quantified and the uncertainty description can then be used in stability analysis techniques to determine the allowable bound on the size of uncertainty to ensure closed loop stability ([1] in these proceedings). In this paper, modelling of the uncertainties associated with the response matrix is presented. Firstly, the uncertainties are described within the commonly used SVD framework. However there are limitations to this approach and as an alternative, a harmonic decomposition of the response matrix is presented and the uncertainty is expressed in terms of the Fourier coefficients of the response.

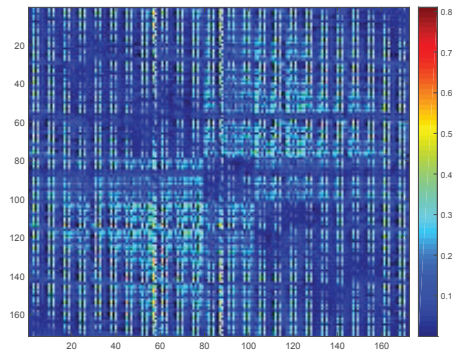


Figure 1: Additive uncertainty in response matrix R_0 ($|\Delta_R|$).

UNCERTAINTY DESCRIPTIONS USING SINGULAR VALUE DECOMPOSITION

The uncertainty in the response matrix can be expressed by an additive error,

$$R = R_o + \Delta_R \tag{2}$$

where R represents a measured response matrix and R_o represents the nominal response matrix. Figure 1 shows the structure of Δ_R for the Diamond storage ring taken from one measurement of R. The uncertainty in the response matrix combines the errors in Φ, Σ and Ψ and it is not straightforward to distinguish, for example, the effect of BPM errors from corrector errors. It is therefore more useful to model the uncertainty in the process output (sensor values) by multiplicative operators [2], which can be expressed as

$$R = (I + \Delta_B) R_0 \tag{3}$$

so that

$$\Delta_B = R R_0^{-1} - I. \tag{4}$$

The uncertainty can be projected into "mode space", such that

$$\Delta_B = \Phi \Sigma \Psi^{\mathrm{T}} \Psi_0 \Sigma_0^{-1} \Phi_0^{\mathrm{T}} - I. \tag{5}$$

Likewise, the same treatment can be given to uncertainties in the process input (corrector values), which is expressed as

$$R = R_0 (I + \Delta_C) \tag{6}$$

so that

$$\Delta_C = \Psi_0 \Sigma_0^{-1} \Phi_0^{\mathrm{T}} \Phi \Sigma \Psi^{\mathrm{T}} - I. \tag{7}$$

By assuming that all the uncertainty lies in either Φ or Ψ, then the uncertainties can be expressed as

$$\begin{aligned} \bar{\Delta}_B &= \Phi \Phi_0^{\mathrm{T}} - I \\ \bar{\Delta}_C &= \Psi_0 \Psi^{\mathrm{T}} - I. \end{aligned} \tag{8}$$

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

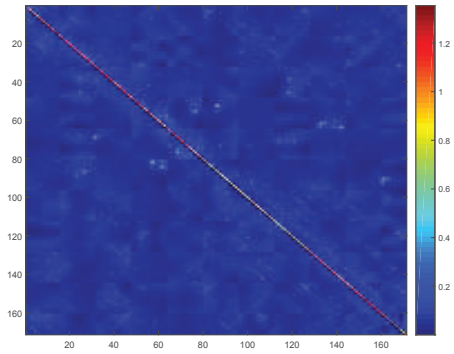


Figure 2: Multiplicative uncertainty in left singular vectors, Φ_0 ($\bar{\Delta}_B$).

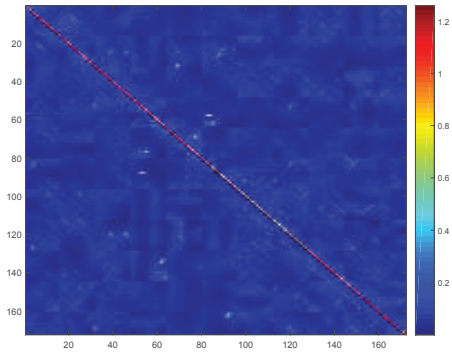


Figure 3: Multiplicative uncertainty in right singular vectors, Ψ_0 ($\bar{\Delta}_C$).

The shape of these uncertainties are shown in Fig. 2 and Fig. 3 respectively where the uncertainty between modes are the most significant (represented by the diagonal elements).

The uncertainty in the singular values can also be included as a multiplicative error, such that

$$\Delta_\Sigma = \Sigma_0^{-1}\Sigma - I \tag{9}$$

where Δ_Σ is diagonally structured since Σ_0 and Σ are diagonal. The diagonal values are shown in Fig. 4 where the size of the uncertainty in the higher order modes (associated with small singular values) is greater than the error in low order modes. Therefore the high order modes are more vulnerable to stability problems.

The response matrix with all sources of uncertainty can then be written as

$$R = (I + \bar{\Delta}_B)\Phi_0 \;\; \Sigma_0(I + \Delta_\Sigma) \;\; \Psi_0(I + \bar{\Delta}_B) \tag{10}$$

While the uncertainty descriptions using the SVD matrices are useful for analysis, such as identifying which modes are more affected by uncertainties, there is no physical interpretation of the uncertainty structure and effects from individual BPMs and correctors are difficult to decouple. Instead, a harmonic decomposition method is proposed in the next section so that the effects of BPMs, corrector and tune drift can be decoupled and modelled.

UNCERTAINTY DESCRIPTIONS USING HARMONIC DECOMPOSITION

Harmonic Decomposition of the Response Matrix

For a transverse excitation θ_n at the nth corrector and measured by the mth BPM, the disturbed closed orbit may be expressed as

$$y_m = \frac{\theta_c\sqrt{\beta_m}\sqrt{\beta_n}}{2\sin\pi\nu}\cos\left(\nu\pi - (|\eta_m - \eta_n|)\right) \tag{11}$$

where β_m and β_n are the magnitudes of the beta function at BPM and corrector locations respectively and η_m and η_n are the phase advances at BPM and corrector locations respectively and ν is the tune. By introducing the Courant-Snyder variables [3]

$$\tilde{y}_m = \frac{y_m}{\sqrt{\beta_m}}, \quad \tilde{\eta} \;\; = \frac{\eta}{\nu} \tag{12}$$

Eq. 11 can be written as

$$\tilde{y}_m = \frac{\theta_n\sqrt{\beta_n}}{2\sin\pi\nu}\cos\nu\left(\pi - (|\tilde{\eta}_m - \tilde{\eta}_n|)\right). \tag{13}$$

Fourier analysis of Eq. 13 gives [4, 5]

$$\tilde{y}_m = \frac{\theta_n\beta_n}{2\sin\pi\nu}\sum_{f=0}^{\infty}\hat{\sigma}_f\cos f(\tilde{\eta}_m - \tilde{\eta}_n) \tag{14}$$

where

$$\begin{aligned}\hat{\sigma}_f &= \frac{(-1)^f}{\pi}\int_{-\pi}^{\pi}\cos f\tilde{\eta}\cos\nu\tilde{\eta}\,\mathrm{d}\tilde{\eta} \\ &= \frac{2\nu}{\nu^2 - f^2}\frac{\sin\nu\pi}{\pi} \quad (f = 1, 2, \ldots).\end{aligned} \tag{15}$$

The expression in Eq. 14 can be further simplified by

$$\tilde{y}_m = \theta_n\beta_n\sum_{f=0}^{\infty}\hat{\sigma}_f\cos f(\tilde{\eta}_m - \tilde{\eta}_n) \tag{16}$$

where the Fourier coefficients are redefined as

$$\hat{\sigma}_f = \frac{1}{2\pi}\frac{2\nu}{\nu^2 - f^2} \quad (f = 0, 1, 2, \ldots). \tag{17}$$

Therefore each element of the response matrix is defined as y_m/θ_n is given by

$$\begin{aligned}r_{mn} &= \sqrt{\beta_m}\sqrt{\beta_n}\sum_{f=-F}^{F}\hat{\sigma}_f\cos f(\tilde{\eta}_m - \tilde{\eta}_n) \\ &= \sqrt{\beta_m}\sqrt{\beta_n}\sum_{f=-F}^{F}\hat{\sigma}_f\mathrm{Re}(\mathrm{e}^{if\tilde{\eta}_m}\mathrm{e}^{-if\tilde{\eta}_n})\end{aligned} \tag{18}$$

for $f = [-F, \ldots, F]$ where F is chosen (without loss of generality) so that $F \leq M$ where $M \leq N$. In matrix form

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

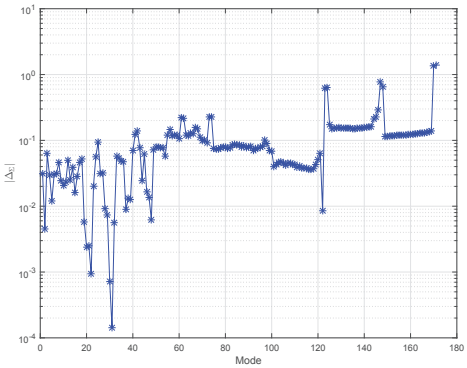


Figure 4: Multiplicative uncertainty in singular vectors, Σ_0 ($|\Delta_\Sigma|$).

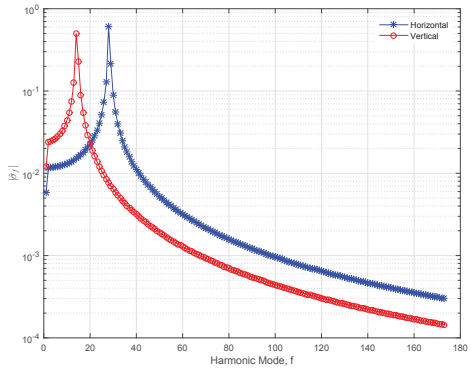


Figure 5: Horizontal ($*$ blue) and vertical ($\circ$ red) Fourier coefficients $|\hat{\sigma}_f|$ with vertical tune $\nu_y = 13.324$ and $\nu_x = 27.266$.

Eq. 18 becomes

$$R_0 = \hat{\Phi}_0 \hat{\Sigma}_0 \hat{\Psi}_0^{-\mathrm{T}} \tag{19}$$

where

$$\begin{aligned} \hat{\Sigma}_0 &= \mathrm{diag}_{f=-F,\ldots,F}\{\hat{\sigma}_f\} \\ \hat{\Phi}_0 &= \mathrm{diag}_{m=1,\ldots,M}\{\sqrt{\beta_m}\}\mathrm{Re}\left(\left[e^{if\tilde{\eta}_m}\right]_{mf}\right) \\ \hat{\Psi}_0 &= \mathrm{diag}_{n=1,\ldots,N}\{\sqrt{\beta_n}\}\mathrm{Re}\left(\left[e^{-if\tilde{\eta}_n}\right]_{nf}\right) \end{aligned} \tag{20}$$

and $\hat{\Phi}_0 \in \mathbb{R}^{M\times(2F+1)}$, $\hat{\Sigma}_0 \in \mathbb{R}^{(2F+1)\times(2F+1)}$ and $\hat{\Psi}_0 \in \mathbb{R}^{N\times(2F+1)}$ [5]. The Fourier coefficients defined in Eq. 17 are completely determined by the tune and are shown in Fig. 5 for the vertical and horizontal response matrices of the storage ring. The resonance properties of the disturbed closed orbit are evident and show that the orbit is most sensitive to those Fourier components of the perturbation whose order is close to the tune. The left and right matrices $\hat{\Phi}_0$ and $\hat{\Psi}_0$ are determined by the betatron phases of the BPMs and corrector magnets, which in turn depends solely on the position of the BPMs and the strengths of the corrector magnets respectively. The columns of $\hat{\Phi}_0$ and $\hat{\Psi}_0$ are not orthogonal because of the uneven spacing of BPMs and corrector magnets. However, the matrices are block orthogonal reflecting the repetition of cells and symmetry of the ring.

Modelling Uncertainties using Harmonic Decomposition

Given the expression for Fourier coefficients in Eq. 17, a change in a single fourier coefficient with respect to the tune can be expressed as

$$\Delta_{\hat{\sigma}_f} \approx \left.\frac{\delta\hat{\sigma}_f}{\delta\nu}\right|_{\nu=\nu_0} \Delta_\nu \tag{21}$$

where ν_0 is the nominal tune and the partial derivative term can be written as

$$\left.\frac{\delta\hat{\sigma}_f}{\delta\nu}\right|_{\nu=\nu_0} = \frac{\hat{\sigma}_f}{\nu_0}\left(\frac{f^2+{\nu_0}^2}{f^2-{\nu_0}^2}\right). \tag{22}$$

Therefore, the uncertainty in a Fourier coefficient, $\Delta_{\hat{\sigma}_f}$ is dependent only on the tune and can be represented by a multiplicative operator

$$\Delta_{\hat{\sigma}_f} = \left(\frac{\hat{\sigma}_f}{\nu_0}\left[\frac{f^2+{\nu_0}^2}{f^2-{\nu_0}^2}\right]\right)\Delta_\nu \tag{23}$$

where Δ_ν represents the change in tune from the nominal value. In matrix form,

$$\Delta_{\hat{\Sigma}} = W_{\hat{\Sigma}}\Delta_\nu \tag{24}$$

where

$$W_{\hat{\Sigma}} = \mathrm{diag}\left\{\frac{\hat{\sigma}_f}{\nu_0}\left[\frac{f^2+{\nu_0}^2}{f^2-{\nu_0}^2}\right]\right\}. \tag{25}$$

Uncertainty can also be included in the left and right harmonic matrices as in the SVD approach, so a complete description of the spatially uncertain response can be written as

$$\begin{aligned} R &= \left(I+\Delta_{\hat{\Phi}}\right)\hat{\Phi}_0\ \hat{\Sigma}_0\left(I+\Delta_{\hat{\Sigma}}\right)\ \hat{\Psi}_0^{-\mathrm{T}}\left(I+\Delta_{\hat{\Psi}}\right) \\ &= \left(I+\Delta_{\hat{\Phi}}\right)\hat{\Phi}_0\ \hat{\Sigma}_0\left(I+W_{\hat{\Sigma}}\Delta_\nu\right)\ \hat{\Psi}_0^{-\mathrm{T}}\left(I+\Delta_{\hat{\Psi}}\right) \end{aligned} \tag{26}$$

where $\Delta_{\hat{\Sigma}}$ is a diagonal matrix with elements determined by Eq. 23, $\Delta_{\hat{\Phi}}$ represents errors in the BPM positions and $\Delta_{\hat{\Psi}}$ represents variations in the strengths of the corrector magnets.

Uncertainty Analysis

By introducing a constant energy deviation of 0.03 GeV, the beta function and phase advance changes, such that the resulting uncertainties in $\hat{\Phi}$ and $\hat{\Psi}$ associated with these changes are shown in Fig. 6 and Fig. 7. Because $\hat{\Phi}$ and $\hat{\Psi}$ are determined by the beta function and phase advance, both $\Delta_{\hat{\Phi}}$ and $\Delta_{\hat{\Psi}}$ have a large diagonal structure but coupling between the beta function and phase advance within a cell results in the off-diagonal elements being non-zero. However, each element of $\hat{\Phi}$ and $\hat{\Psi}$ is associate with beam parameters η and β at BPM and corrector locations respectively. Likewise the energy deviation causes a 0.2 % change in tune which

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

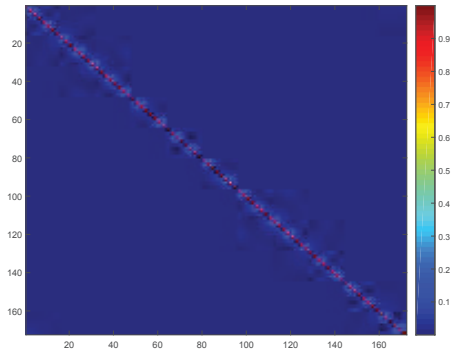


Figure 6: Multiplicative uncertainty in left harmonic matrix, $\hat{\Phi}_0$ ($\Delta_{\hat{\Phi}}$).

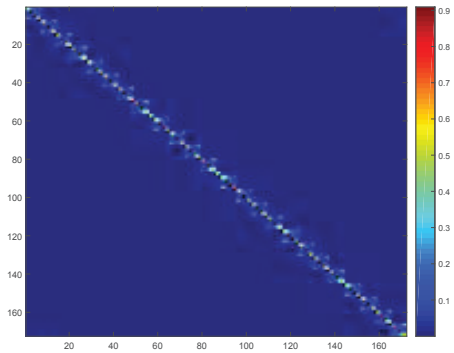


Figure 7: Multiplicative uncertainty in right harmonic matrix, $\hat{\Psi}_0$ ($\Delta_{\hat{\Psi}}$).

affects the Fourier coefficients, represented by $\Delta_{\hat{\Sigma}}$ shown in Fig. 8. The uncertainty is diagonally structured and the largest error is seen at the modes associated with the tune i.e. the 13th mode (where $\nu_y = 13.36$). For this calculation $F = 40$ was chosen. However, given the relationship in Eq. 24, the uncertainty in the tune can be expressed as

$$\Delta_\nu = W_{\hat{\Sigma}}^{-1} \Delta_{\hat{\Sigma}} \tag{27}$$

and using this relationship, size of uncertainty in the tune at each harmonic harmonic modes is shown in Fig. 8. Therefore for a given error in each Fourier coefficient, the resulting change in tune at each harmonic mode can be determined.

CONCLUSION

Uncertainty in the response matrix can be described by multiplicative uncertainty descriptions using either a Singular Value or Harmonic decompositions of the response matrix. Using the SVD approach, the uncertainty in the

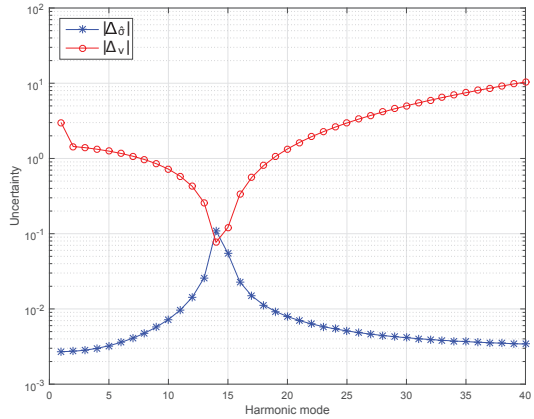


Figure 8: Multiplicative uncertainty in Fourier coefficients, $\hat{\Sigma}_0$ ($\Delta_{\hat{\Sigma}}$) and in the tune across modes (Δ_ν).

response matrix can be expressed in terms of the left and right singular vectors and the singular values. This is useful for assessing the stability of the closed loop which is based on SVD, however it is difficult to interpret the uncertainty description in terms of physical beam parameters and to decouple the error observed at a singular BPM or corrector. The Fourier approach on the other hand, offers uncertainty descriptions which are directly associated with the betatron function and phase advance changes at BPM and corrector locations. Furthermore, errors in the Fourier coefficients are directly associated with tune drifts. Therefore the Fourier approach offers physical interpretation of the uncertainty observed in the response matrix. In the companion paper in these proceedings, the uncertainty descriptions are used to determine closed loop stability [1].

REFERENCES

[1] S. Gayadeen, M. T. Heron, and G. Rehm, "Robust stability analysis of orbit feedback controllers," in *MOPGF177, these proceedings, ICALEPCS'2015*, (Melbourne, Australia), 2015.

[2] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback Control Theory*. Macmillan, 1992.

[3] E. B. Courant and H. S. Snyder, "Theory of the alternating gradient synchrotron," *Annals of Physics*, vol. 3, pp. 1–48, 1958.

[4] L. H. Yu, E. Bozoki, J. Galayda, S. Krinsky, and G. Vignola, "Real time harmonic closed orbit correction," *Nuclear Instruments and Methods in Physics Research Section A*, vol. 284, no. 2–3, pp. 268–285, 1989.

[5] S. Gayadeen, *Synchrotron Electron Beam Control*. PhD thesis, Department of Engineering Science, University of Oxford, UK, 2014.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

STATUS OF THE SOLARIS CONTROL SYSTEM - COLLABORATIONS AND TECHNOLOGY *

P. Goryl#, C.J. Bocchetta, P. Bulira, Ł. Dudek, P. Gałuszka, A. Kisel, W. Kitka, M. Kopeć, P. Kurdziel, M. Ostoja-Gajewski, M.J. Stankiewicz, J. Szota, A.I. Wawrzyniak, K. Wawrzyniak, Ł. Żytniak, Solaris, Krakow, Poland
T. Szymocha, ACK Cyfronet AGH, Krakow, Poland
V. Hardion, J. Jamróz, J. Lidon-Simon, M. Lindberg, A.G. Persson, D. Spruce, MAX IV Laboratory, Lund, Sweden
I. Dolinsek, Cosylab, Ljubljana, Slovenia

Abstract

The Solaris is a synchrotron light source starting just now in Krakow, Poland. It is built with strong collaboration with other European accelerator facilities. The MAX-IV project in Lund, Sweden and Tango Community are the most important partners in the project in respect to the control system. Solaris has built a twin copy of MAX-IV 1.5GeV ring and linear accelerator based on the same components as the ones of MAX-IV. Thus, both facilities share know-how and apply similar technologies for the control system, among them the Tango CS is used for software layer. Status of the control system in Krakow as well as collaborations and technological choices impact on its success are presented in this paper.

THE ACCELERATOR AND BEAMLINES

The Solaris machine is 1.5GeV storage ring the same as one of the MAX-IV supplied with a linear accelerator providing electron beam up to 600MeV. Since the linac is not providing full energy beam there is need of energy ramping in the storage ring. This is a difference to MAX-IV setup. The Solaris project includes also two beamlines. One is using bending magnet radiation with XAS and PEEM end-stations and the other is basing on undulator source with an UARPES end-station [1].

The accelerators and beamlines are installed except few components of PEEM beamline and two Landau cavities in the storage ring. There is on-going commissioning of the accelerator [2]. The latest result is accumulation of 20mA current in the storage ring which then has been ramped to full energy of 1.5GeV. Currently certain optimization has been done that let us by the day of writing this paper (17.10.2015) accumulate the current of 48mA. The work is in progress to do energy ramping of this much od current. For now limitation are instabilities due to the ion trapping process.

There is planned upgrade shutdown starting end of November 2015 till beginning of January 2016. During this period the landau cavities will be installed as well as missing components of the PEEM beamline. Network infrastructure will be supplemented with a new core switch to enable 10GBps connections through the whole network. It is also planned to do upgrade of the Tango control system to the newest version of Tango 9.1.

After the shutdown commissioning of the beamlines will proceed. It is expected that the Solaris will be ready for the first external users in second half of 2016.

THE CONTROL SYSTEM

The control system of the accelerators and beamlines has been deployed and is operational. All major issues has been debugged and solved. However, few low level components still need optimization. Certain high level applications for future simple operation are upon development.

Tango Control System

The Tango Controls [3] is used for systems integration. Since the Solaris and MAX-IV have similar systems most of the software components are shared and lot of them have been developed in Lund [4]. Solaris and MAX-IV are developing their applications with the Python programming language. The PyTango, the Taurus and the Sardana packages developed at ALBA are used for this purpose. An archiving system which role is to store history of controlled signals is using HDB and TDB tools developed at Soleil [5].

Some of the device servers provided by the Tango Community [6] were reused.

The integration work has been contracted by Solaris to a commercial company Cosylab (Ljubljana, Slovenia). They have also developed few device servers and part of GUI applications including an open source ControlProgram which is an application being an entry point to all control system functionalities needed for operation [7]. The ControlProgram has been published on the public Solaris GitHub repository [8].

The Matlab Midlayer with Tango-Matlab binding [9] is used for machine physics control. The basic configuration has been done by the Cosylab then the work has been overtaken by Solaris. The work were conducted with support from Soleil.

Control System Hardware

The timing system has been provided within the contract with Cosylab. The system based on Micro Research

*Work supported by the European Regional Development Fund within the frame of the Innovative Economy Operational Program: POIG.02.01.00-12-213/09
#piotr.goryl@uj.edu.pl

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Finland (MRF) hardware has been delivered with Tango software – device servers and GUIs – developed by the Cosylab.

Following the MAX-IV, Solaris is using as a standard the IcePAP[10] system for motion control. The system has been developed by ESRF and ALBA then it was shared within so called IcePAP collaboration. Even the Solaris, due to formal limitations, is not part of the formal IcePAP collaboration the permission given by the ESRF let us purchase the hardware. Solaris get also permission from ALBA, to use IcePAP Control Management System software and related libraries.

PLC systems

A machine protection PLC system (MPS) is based on Rockwell Automation solutions as the one of MAX-IV. However there are some differences between MAX-IV and Solaris, both systems use the same philosophy of interlock handling. The PLC software has been developed in strong collaboration between engineers from Krakow and Lund.

The personal safety system (PSS) which provides radiation safety is using Siemens hardware. The PSS is outcome of collaboration with Elettra (Trieste, Italy). They have provided conceptual design and software whereas executive design and hardware has been procured locally.

Status

All the control system required for the machine commissioning has been provided. There is about 1500 devices available through the Solaris Tango system providing access to about 5000 signals. All of these devices are available to operator either through generic GUI Taurus panels or through dedicated engineering screens. Dedicated group GUI panels are provided for vacuum, RF, magnet and interlock subsystems.

These applications give very detail information and functionalities which are necessary for the commissioning. Currently work has started on preparation so called operation GUIs which will aggregate information and functionalities to the level required for day to day machine and beamlines operation.

As it was mentioned before, there is already planned upgrade of the network infrastructure (the core switch) to allow 10GBps operation and effective handling of expected beamline data transfers. In parallel the Tango system, its libraries and related packages will be upgraded to the newest versions to make the system ready and up to date for the users. As it is common in the upgrade process there are risks of regression or introducing new issues. To minimize this pre-tests are planned. The schedule of the Solaris which assumes first users coming in second half of 2016 gives enough time to solve possible problems.

COLLABORATIONS

Three international collaborations contributed mostly to the Solaris control system. These are: MAX-IV, Elettra and Tango Community.

MAX-IV

The Solaris adopted concept of the linear accelerator and design of 1.5GeV storage ring provided by the MAX-IV. However, the Solaris and MAX-IV projects are tightly connected these are formally two separate projects realized by two remote organizations. This could potentially bring difficulties in case of weak connections between teams.

For the first 4 years of the project, till October 2013, the MAX-IV hosted in Lund part of the Solaris team. The number of Solaris members staying in Sweden for more than one month varied from 2 to 6 persons. At the beginning the main purpose was to let the Solaris people get knowledge and experience in the field of accelerators. This allows for effective knowledge transfer as well as active participation of the Solaris in both projects. Till now the Solaris people regularly visits MAX-IV. The direct contacts maid have positive impact on continuation of effective cooperation.

As a consequence of sharing the design of machine both facilities shares technologies used for control systems. This prevents duplication of work and lower overall cost of the control system. Due to availability of developers with certain experience, schedule constraints and miscommunication some of packages initially developed by MAX-VI in Python has been re-implemented in C++ for Solaris during tests and validation process. However, in such cases the Tango interface where kept equal.

Elettra

The collaboration with Elettra was based initially on in-kind support and then it was converted to consultancy contract. This allows on one side pay for the support and on other side solves administration issues and limitations that arise in case of in-kind support. For the control system the collaboration lets Solaris cover differences to MAX-IV [11] and supplements Solaris knowledge and experience. As it was mentioned they provide design and software of the PSS system. Elettra developed also the Energy Ramping application. Both of these projects introduced new technologies at Solaris – one is solution based on Siemens PLCs, the other is usage of the C++ QTango library. Elettra has supported control group with among others requirements gathering providing useful reports on prioritization of control system components in respect to commissioning needs.

Tango Community

The Tango Community provided a base of the Solaris control room software – the TANGO Controls suit [3], library of Device Servers [6] and applications. They provide also great support by serving advises and sharing ideas. Formal and informal discussions where useful for decisions making on technical and management aspects.

PL-Grid

There is also local collaboration between Solaris and the PL-Grid Plus project [12] and especially with the ACK Cyfronet AGH. The PL-Grid Plus project prepared so called domain related services built upon the high power

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

computing infrastructure in Poland (PL-Grid). Among others services Elegant, Tracy and Virtual Accelerator were implemented giving access to two computational tools used for accelerators design and verification and simulated control system environment used to test Solaris MML configuration. The storage services provided by the PL-Grid will be used to store future experiments' data. The ACK Cyfronet AGH is also supporting Solaris with development of the so called Digital User Office – a system to handle users beam-time applications and conducting of experiments. The system is developed on platform already used for e-gran system of the European Grid Initiative.

ACKNOWLEDGMENT

As it was show in the paper the Solaris success would not be possible without support of colleagues from other laboratories. Not all were mentioned. We received lot of useful advices and equally important encouraging words. Thus authors would like to thank all who shared their knowledge, time, resources and development. Among others they are:

- The Solaris Team
- MAX-IV Team and especially KITS for the control system
- Tango Community
- ELETTRA
- ALBA
- ESRF
- SOLEIL
- DESY
- PL-Grid Plus Project
- Others

REFERENCES

[1] C.J. Bocchetta et al., "Solaris Project Progress", IPAC2013, Shanghai, 2013

[2] A.I. Wawrzynak et al., "Firs Results of Solaris Synchrotron Commissioning", WEDLA01, IBIC2015, Melbourne, 2015

[3] Tango Controls website: http://www.tango-controls.org/

[4] V. Hardion et al., "MAXIV Laboratory, Milestones and Lessons Learned", MOB3O03, ICALEPCS'15, Melbourne, 2015

[5] G. Abeille et al., "Tango Archiving Service Status", ICALEPCS'11, Grenoble, 2011

[6] Tango Device Servers Sourceforge project: http://sourceforge.net/projects/tango-ds

[7] V. Juvan et al., "A Structured Approach to Control System GUI Design for the Solaris Light Source", WEPGF145, ICALEPCS'15, Melbourne, 2015

[8] Solaris public GitHub repository: https://github.com/synchrotron-solaris

[9] L.S. Nadolski at al.,"Control Applications for SOLEIL Commissioning and Operation", THPCH109, EPAC2006, Edinburgh 2006

[10] N. Janvier et al., "IcePAP: An Advanced Motor Controller for Scientific Applications in Large User Facilities", TUPPC081, ICALEPCS2013, San Francisco, 2013

[11] P.P. Goryl et al., "Solaris Project Status and Challenges", MOPMU008, ICALPCS2011, Grenoble, 2011

[12] T. Szymocha et al., "Services for synchrotron deployment and operations", proceedings of CGW'13, Kraków, 2013

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DETECTOR CONTROLS MEETS JEE ON THE WEB

F. Glege, M. Janulis, A. Andronidis, O. Chaze, C. Deldicque, M. Dobson,
A. Dupont, D. Gigi, J. Hegeman, R. Jiménez Estupiñán, L. Masetti, F. Meijers,
E. Meschi, S. Morovic, C. Nunez-Barranco-Fernandez, L. Orsini, A. Petrucci, A. Racz,
P. Roberts, H. Sakulin, C. Schwick, B. Stieger, S. Zaza, P. Zejdl (CERN, Geneva, Switzerland),
U. Behrens (DESY, Hamburg, Germany), O. Holme (ETH Zurich, Switzerland),
J. Andre, R. K. Mommsen, V. O'Dell (Fermilab, Batavia, Illinois, USA), G. Darlea,
G. Gomez-Ceballos, C. Paus, K. Sumorok, J. Veverka (MIT, Cambridge, Massachusetts, USA),
S. Erhan (UCLA, Los Angeles, California, USA),
J. Branson, S. Cittolin, A. Holzner, M. Pieri (UCSD, La Jolla, California, USA)

Abstract

Remote monitoring and controls has always been an important aspect of physics detector controls since it was available. Due to the complexity of the systems, the 24/7 running requirements and limited human resources, remote access to perform interventions is essential.
The amount of data to visualize, the required visualization types and cybersecurity standards demand a professional, complete solution.
Using the example of the integration of the CMS detector controls system into our ORACLE WebCenter infrastructure, the mechanisms and tools available for integration with controls systems shall be discussed. Authentication has been delegated to WebCenter and authorization been shared between web server and control system. Session handling exists in either system and has to be matched. Concurrent access by multiple users has to be handled. The underlying JEE infrastructure is specialized in visualization and information sharing. On the other hand, the structure of a JEE system resembles a distributed controls system. Therefore an outlook shall be given on tasks which could be covered by the web servers rather than the controls system.

INTRODUCTION

The Compact Muon Solenoid (CMS) detector is one of the four experiments of the Large Hadron Collider (LHC) at CERN. The Detector Controls System (DCS) is needed to bring the detector in a state to collect physics data. The scale of the system is illustrated with the following numbers:

- ~3 million parameters
- ~700.000 lines of code
- ~35000 finite state machine nodes
- 34 SCADA systems
- 29 redundant PC pairs (Windows)
- ~50 DB schemas (ORACLE)
- O(TB) of data in database schemas

The system is running in production since several years with a high efficiency. It is built using the SCADA system WinCC OA from the Austrian company ETM. At CERN the Joint Controls Project (JCOP) created a toolkit based on WinCC OA to facilitate the development of the experiment controls systems. Today this is used by all LHC experiments and for several other projects at CERN.

PART1: REMOTE CONTROL VIA WEB

Visualization in WinCC OA

In WinCC OA visualization is done through so-called panels. Using a graphical editor, graphics objects (buttons, text, lines, etc.) can be placed on a window. Each element can be enhanced by scripts, which allow to change the properties of the element itself. The scripts have access to the process data and can therefore reflect the process information through the visualization. The graphical editor produces a file describing the panel layout and functions. This file is then used by a WinCC OA process to visualize these panels. This process uses Qt as a graphics library, with all graphics elements being drawn using the Qt graphics API. The Qt Library itself translates the drawing of the panel into primitive paint commands which are sent to the device used for visualization.

Remote Control

Around 4000 members of the CMS collaboration should be able to access the DCS information remotely in a read only manner. The data should ideally be live to ensure a good user experience. Experts should in addition be able to interact and remotely control their systems. Authorization and authentication are required to enable interaction.
The means for remote control currently used in CMS are

- snapshots taken regularly of selected panels being served as images vie the web server
- HTML/java script pages providing live information collected
  - from the logging database
  - through protocols supported and used by the controls system

A full discussion of the current remote control solution in CMS DCS can be found in [1].

Live Web Access

Thousands of panels have been created for the detector controls of the LHC experiments. Reimplementing all of them at once to provide a pure HTML solution which

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

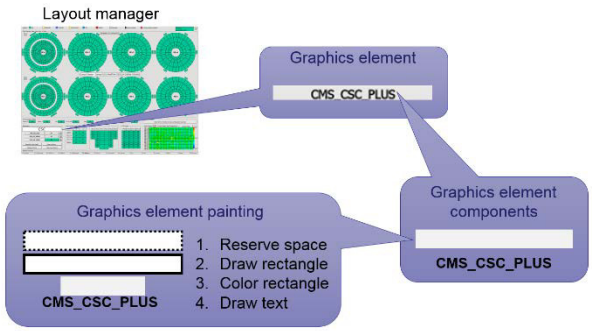


Figure 1: The graphics stack.

would offer live data access using standard web tools is not an option. Giving access to WinCC OA panels directly through remote access tools like remote desktop, VNC, etc. would require too many computing resources caused by the large number of possible clients.

A good compromise represents the usage of the Qt platform abstraction mechanism [2]. Graphics in applications are usually created on several layers (see Fig. 1). On the highest layer a layout manager will position graphics objects according to the rules the designer has set. Each graphics object is composed of graphics primitives. Those primitives are drawn using basic commands to reflect its geometry.

The Qt platform abstraction mechanism allows to intercept the basic drawing commands (see Fig. 2). It happens that those commands are often similar for different visualization devices. Also the commands used in an HTML5 [3] canvas are similar to those used in Qt. The solution is to intercept those commands in Qt and send them to a web browser, where they are applied to an HTML5 canvas. Doing so creates an exact image of the Qt graphics inside the browser. Sending all events of the HTML5 canvas (mouse movement, clicks, button press, etc.) back to Qt allows to have exactly the same control remotely as one would have on the local Qt graphics display.

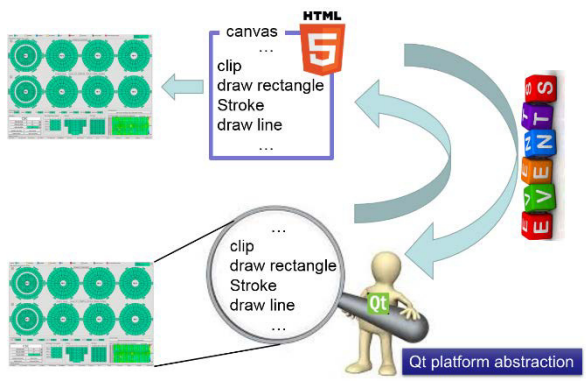


Figure 2: Paint command redirection.

Web Server as a Gateway

To make the above mentioned mechanism usable for many clients in a real environment, a web server offering JEE (JAVA Enterprise Edition)[4] has been added as a gateway between Qt and the browser. Doing so provides

- authentication and authorization based on standards
- multi user access
- firewalling using well tested, widely used software
- business logic implementation and process modelling

A resource broker is required to handle the graphics sources (in our case WinCC OA user interface managers) and steer the data exchange. On request the resource broker starts a new user interface manager, if the requested panel is not yet served by another manager, and tells the Web socket proxy to set up the connection (see Fig. 3). In response to the initial browser request an HTML client containing the HTML5 canvas and all necessary Java Script code will be returned. This client will open a web socket to the web socket proxy which will then open the connection to the user interface manager and start proxying data. The web server infrastructure will ensure that the client is authenticated and authorized to do this action.

Two modes are possible using this mechanism:

- One client per panel, allowing for read write access to the panel. This is to be used by system experts, who need to interact with their system
- Many clients per panel, allowing only read access. This is to be used by any authenticated user to gather the current system status.

In our case the resource broker and the web socket proxy have been implemented as a proof of concept and run in an ORACLE portal infrastructure.

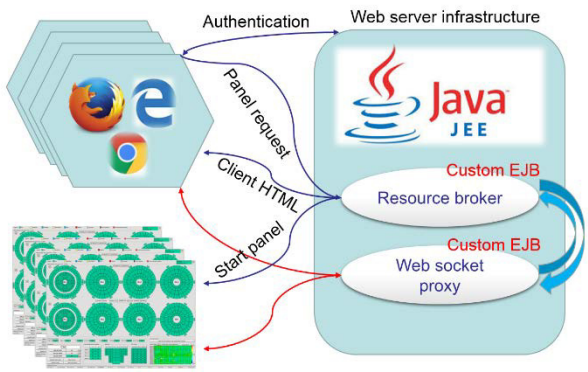


Figure 3: Work flow.

Why JEE?

Our web server infrastructure provides JEE. The JEE concept contains EJBs (enterprise JAVA beans) which have several features that ease the implementation of the described remote access mechanism:

- EJBs model the business logic of a JEE application
- EJBs are not limited to a request
- EJBs provide intercommunication
- EJBs allow modularity and scalability
- EJBs allow for distributed systems

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

By implementing a resource broker and a web socket proxy as an EJB, they profit from the inherited infrastructure. The reception of the initial browser request is very simple in an EJB as is the communication amongst different EJBs. These components can be distributed over several web server instances (e.g. to do load balancing) using the same communication mechanism.
Thus the use of JEE significantly eased the development for CMS and allows building complex system architectures out of the box to fulfil functional requirements.

Performance

The first implementation simply expressed all paint commands intercepted in Qt as HTML5 canvas paint commands and sent them to the browser. The bandwidth required to show a reference panel was ~1.1MB/s. This did not allow for a good user experience since drawing on the canvas was lagging behind and even crashing the browser. Several optimizations have been applied to improve this:

- Command indexing:
  - The paint commands have been replaced by a numerical representation.
- Command sequence caching:
  - All commands are aggregated in a continuous buffer. If the server identifies a command or sequence of commands which has already been sent, it just sends a pointer and the length of the sequence to be repeated.
- Image caching.
- Optimized TCP packet usage:
  - Commands are sent in chunks to minimize the network protocol overhead.
- Update frequency Reduced to human perceivable rate
- Optimized (reduced) event sending:
  - A mouse movement on the client side will create a move event for every pixel it moves. Small movements are collected, combined and sent as a single movement.

After applying the previously mentioned optimizations, the required band width could be reduced to ~20kB/s. This provides a sufficiently good user experience where the remote graphics are nearly in sync with the local one.

Next Step

Many panels provide navigation mechanisms: either they contain tabs or provide access to other panels. Ideally navigation should be available to all users and not only to the expert, with rights to interact, since the full information is only obtained through this mechanism. In the current mechanism it is not possible to implement this since the navigation items (tabs, buttons, etc.) are not identifiable on the client side. (The HTML5 canvas only knows about an accumulation of geometrical objects)
Furhermore, accepting events on navigation items from all users would mean that the resulting changes in the panel would also be shown to all users.
To overcome this problem and allow for basic navigation, the graphics elements used for navigation could be replaced by HTML objects. Now the browser would identify actions on those elements and could request the underlying action to the server. In case of tabs, the resource proxy would redirect the connection of this browser to a panel showing the requested tab.
Since the navigation elements will no longer be transferred as graphics primitives but as high level HTML code, the required bandwidth is reduced and performance increased. Using object based authorization the access to those elements could be restricted on a per user basis.

Final Goal

The final goal is to replace all panels with pure HTML panels. The advantages would be maximum performance through minimal required bandwidth and multi user read/write access. As mentioned before this will require a sizable development effort and is therefore not achievable in short term.

Summary Part1

We implemented a solution for web based remote control mirroring WinCC OA panels using Qt platform abstraction. No modification of panels is needed to provide single user read/write or multi user read only with sufficient performance. An evolution towards a fully HTML based remote visualization is possible and foreseen.

PART2: JEE GRAPHICS AND BEYOND?

The simplified architecture of a controls system could be sketched as shown in Fig 4.

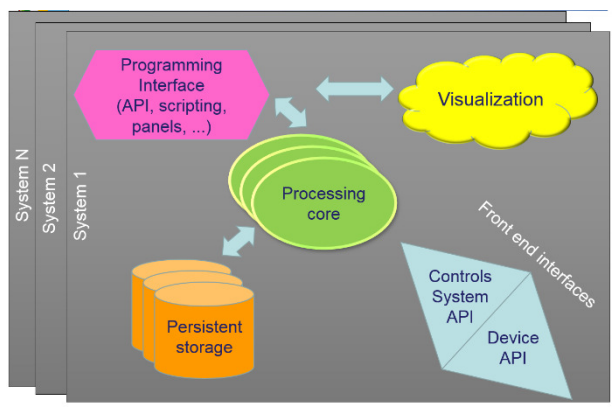


Figure 4: Simple sketch of a controls system architecture.

Apart from the programming interface, all parts are logically independent. The separation of processing core and programming interface might not be as clear as shown and depends on the implementation. The connection to the devices is often device specific. Interface concepts like OPC UA and TCP based bus systems should generalize the front end connections and remove the need for specific hardware interfaces.
After some reshuffling and abstraction the JAVA EE architecture looks like in Fig. 5.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

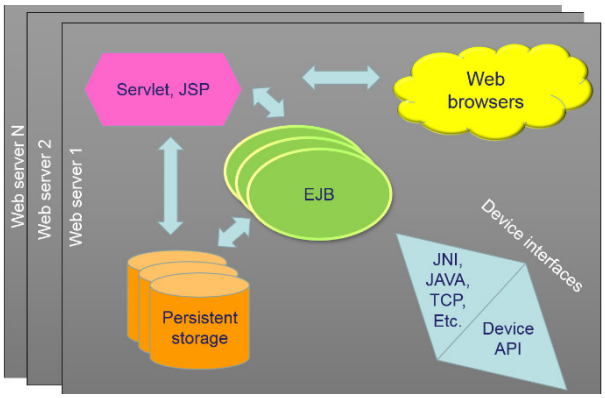


Figure 5: JEE architecture.

It's obvious that a web server can be used for visualization. (The first part of this paper proofs this). The persistent storage is the same in both cases. Through the Java native interface (JNI) any device library can be connected to JAVA. According to their definition the EJBs are meant to be used to model business logic. That is what the processing cores in any controls system do. Also here the borderline between EJBs, servlets and JSP is not a clear separation. Some tasks might be implemented with either type. The general concept therefore looks the same.

JEE for Controls?

Seeing that the concept fits, could one use JEE to build a control system?

A web browser as visualization is certainly very powerful and provides a large amount of freely available components. This is certainly more than most SCADA (Supervisory controls and data acquisition) systems provide.

The access to persistent storage systems is part of JEE. Usually different types of storage are supported by the web server infrastructure. JEE even provides a possibility to declare data items to be made persistent, which is done transparently to the user. The achievable performance has to be tested but the modular nature of the JEE concept should allow to scale up sufficiently to reach the required performance.

The connection to front ends is possible by design, since any library can be connected to JAVA through JNI.

The processing core in JEE, i.e. JSPs (JAVA server pages), servlets and EJBs provide inbuilt local and distributed communication means allowing for a distributed design. Also here a certain performance has to be reached but since JAVA is a standard programming language, this is not more or less difficult than with any other language. The ease of factorization, however, is an advantage over bare C++ for instance.

Summary Part 2

JEE is not a readily built SCADA software. JEE is a concept for which different implementations exist and provides many components and interfaces required to build a controls system. It is well documented, widely used and often free of charge.

For those reasons, JEE seems well suited as a basis for the development of a new controls system.

REFERENCES

[1] Lorenzo Masetti et al., a scalable and homogenous web-based solution for presenting CMS control system data, THCOAAB01, proceedings ICALEPCS 2013.
[2] http://wiki.qt.io/Qt_Platform_Abstraction
[3] http://www.w3.org/TR/html5
[4] http://www.oracle.com/technetwork/java/javaee/overview/index.html

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CS-STUDIO SCAN SYSTEM PARALLELIZATION*

K.U. Kasemir, M.R. Pearson, ORNL, Oak Ridge, TN37831, USA

Abstract

For several years, the Control System Studio (CS-Studio) Scan System has successfully automated the operation of beam lines at the Oak Ridge National Laboratory (ORNL) High Flux Isotope Reactor (HFIR) and Spallation Neutron Source (SNS). As it is applied to additional beam lines, we need to support simultaneous adjustments of temperatures or motor positions.

While this can be implemented via virtual motors or similar logic inside the Experimental Physics and Industrial Control System (EPICS) Input/Output Controllers (IOCs), doing so requires a priori knowledge of experimenters requirements. By adding support for the parallel control of multiple process variables (PVs) to the Scan System, we can better support ad hoc automation of experiments that benefit from such simultaneous PV adjustments.

MOTIVATION

EPICS has been used with great success on the SNS accelerator, and is now a key component of on-going beam line software updates [1]. Compared to the accelerator, each beam line presents a much smaller number of devices to control. At the same time, the beam line environment is more flexible as devices are added, replaced, or operated in different ways.

The CS-Studio Scan System was developed to allow the flexible assembly and execution of "recipes", that is lists of commands [2]. After gaining operational experience on a couple of beam lines, the need for parallel command execution became obvious.

For example, the control of incident beam energy typically requires the adjustment of several neutron chopper settings as well as motor positions for slits or sample orientation. Executing these in parallel instead of sequentially allows for a significantly faster experiment preparation.

Another example is the combined movement of motors or temperatures for ad-hoc experiments. Ideally, the need for the parallel movement of motors is known in advance. Choosing suitable motor controller hardware will then allow configuring the desired motion curves into the hardware, resulting in the most efficient and accurate position control. In reality, the need for ganged motor movement can arise without prior notice. The control system software then needs to make a reasonable attempt at combined motor movement.

*This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

SCAN SYSTEM

Basic Scan Commands

While the scan system supports several commands [2], it is most important to understand the fundamental "Set" command, which writes a value to an EPICS channel:

```
Set("MotorChannel", value=14)
```

It can wait for a read-back to match the written value, with configurable tolerance and timeout in seconds:

```
Set("MotorChannel", value=14,
    readback="Motor.RBV", tolerance=0.1,
    timeout=30)
```

Alternatively, it can await "completion":

```
Set("MotorChannel", value=14,
    completion=True,
    timeout=30)
```

Finally, it can await "completion" and then check a read-back for matching the written value:

```
Set("MotorChannel", value=14,
    completion=True,
    readback="Motor.RBV", tolerance=0.1,
    timeout=30)
```

In the last case, the original implementation used the timeout both to await completion and then to wait for the read-back to match. Operationally, we found it more practical to only apply the timeout to the completion. Once the channel completes, the read-back is expected to match right away as soon as the scan server reads the most recent value of the read-back channel.

A motor-related channel will signal "completion" after the controller hardware moved the motor toward the commanded position, verified the encoder read-back, maybe performed several retries, corrected for backlash, and finally confirmed that the motor rests at the requested position.

If a device does not support "completion", it is tempting to rely on only the read-back as in the second example given above. That command will indeed wait until the read-back of the device matches the desired value, but it is misleading. The read-back may temporarily match during an overshoot or backlash

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

compensation while the controller is still adjusting the value.

The completion mechanism, called "put-callback" in the EPICS documentation, is therefore key to the reliable control of intelligent beam line devices. Consequently, the device support for SNS beam lines was carefully selected or updated to support "completion" confirmation, for example for the Parker 6K motion and Lakeshore 336/350 temperature controllers [3].

Commands Added for Parallelization

The "Parallel" command was added to the scan system. In the following example it concurrently moves two motors and adjusts one temperature:

```
Parallel(Set("Motor1", 13, completion=True, …),
         Set("Motor2", 24.5, completion=True, …),
         Set("Temperature3", 5.9, …) )
```

The scan server performs a "Parallel" command by submitting each command in its body to a separate thread, and then awaiting completion of all threads. Each individual command in the body may await completion and/or check read-backs as previously described, but in the example such details has been replaced by "…" for clarity.

In some cases the actions that need to be performed in parallel are not individual "Set" commands but again consist of multiple commands. The "Sequence" command allows construction such command chains:

```
Parallel(
  Sequence(Set("XYZMode", "Off"),
           Set("XYZSetpoint", 13, completion=True),
           Set("XYZMode", "Active"),
  Sequence(Set("ABCMode", "Off"),
           Set("ABCSetpoint", 13, completion=True),
           Set("ABCMode", "Active") )
        )
```

In this example, we assume that the theoretical "XYZ" and "ABC" devices can only be commanded to a new set point while turned off, and then need to be re-enabled. The scan commands will perform this sequence of commands for both devices in parallel.

Client/Scripting Library

Scan commands are submitted to the scan server network interface in an XML representation. Initially, we used a Java library, invoked by Jython from within the CS-Studio user interface, for this task [2]. To allow more software tools to interact with the scan server, we implemented the PyScanClient, a library that supports assembling scans, submitting them to the scan server, monitoring and controlling the execution, and downloading logged data [4]. The PyScanClient can be used with Jython as well as C-Python.

All code examples in this paper are based on the PyScanClient syntax. A short but complete program for submitting a scan looks like this:

```
from scan import *
client = ScanClient("localhost")
cmds = [ Set("Motor1", 43) ]
client.submit(cmds)
```

While the coordinated movement of two motors is best implemented in the controller hardware, a script like the following can approximate it by computing the desired points and assembling them into a list of parallel moves:

```
cmds = []
for pos in range(0, 100):
  # Demonstration of moving motors X, Y along
  # 0 .. 100 resp. 0 .. 200
  cmd.append(Parallel(Set("X", pos),
                      Set("Y", 2*pos) )  )
```

Site-Specific Settings

EPICS is a loosely coupled, distributed system. Given solely a channel name "XYZ", it is impossible to tell if that channel supports "completion", if there is a related read-back channel, or which timeout and tolerance would be suitable for comparing the read-back to the desired value.

A site-specific standard for channel names can often determine that for example all channels with names containing ":Mot:" are motors, so they should be controlled with completion, using a read-back channel that is based on the same channel name with an added ".RBV" suffix.

The PyScanClient library supports creating such a site-specific setup based on a combination of regular expressions for channel names with python code to handle specific channel names:

```
# Site-specific scan settings
class BeamlineScanSettings(ScanSettings):
   def __init__(self):
      super(BeamlineScanSettings, self).__init__()
      self.defineDeviceClass(".*:Mot:.*",
            completion=True, readback=True,
            tolerance=0.1, timeout=30)
      self.defineDeviceClass("BL:Mot:X42",
            completion=True,
            readback="BL:Mot:X42_Encoder",
            tolerance=0.001, timeout=60)

   def getReadbackName(self, device_name):
      if ":Mot:" in device_name:
        return device_name + ".RBV"
      return device_name
```

Based on this example, scan commands that access channels which contain ":Mot:" in their name will use

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

completion and read-back verification as just described with a default tolerance of 0.1 and a timeout of 30 seconds. One specific motor, "BL:Mot:X42", will use a designated encoder channel for read back instead of the default "BL:Mot:X42.RBV".

When the PyScanClient library has such site-specific settings, individual scan commands can still override them. For example, while a "Set" command for a temperature controller may default to awaiting completion, an experiment that needs to take data while the temperature changes can issue a "Set" command with "completion=False" to override the default.

Meta Commands

The site-specific settings can introduce meta-commands. For example, starting an experiment may be defined as follows to reset counts, start the data acquisition and assert that it enters the correct mode:

```
def Start():
    return Sequence(Set("Det:Counts", 0),
                    Set("BL:DAQ:Mode", "ON"),
                    Wait("DAQ:State", "Running") )
```

When assembling a scan within a python script, this "Start" command can now be used just like the predefined scan server commands. Wrapping its internal commands into a "Sequence" asserts that they will be executed sequentially even when added to the body of a "Parallel" command.

Table Scan

The PyScanClient library offers a table-based abstraction for creating scans. It its most basic form, each column of the table specifies a channel name. Cells in each row provide desired values:

Temperature3	BL:Mot:X42
50	1
100	2

This table creates the following scan:

```
Set("Temperature3", 50)
Set("BL:Mot:X42", 1)
Set(Temperature3", 100)
Set("BL:Mot:X42", 2)
```

Cells can remain empty if a device should not be changed in that row. Cells that contain ranges or lists are expanded left to right:

Temperature3	BL:Mot:X42
[50, 100]	range(1, 3)

This table creates the following scan:

```
Set("Temperature3", 50)
Set("BL:Mot:X42", 1)
Set("BL:Mot:X42", 2)
Set(Temperature3", 100)
Set("BL:Mot:X42", 1)
Set("BL:Mot:X42", 2)
```

"Wait For" and "Value" columns are used to start data acquisition, await a condition, and stop data acquisition. The specifics of how to start and stop data acquisition are provided in the form of meta-commands in the site-specific PyScanClient settings. For SNS, these consist of "Set" commands that instruct the streaming neutron event acquisition to start respectively stop. A simple yet complete scan could be expressed like this:

BL:Mot:X42	Wait For	Value
range(1, 3)	NeutronCount	1e5

This table creates the following scan, assuming that the start and stop commands in the PyScanClient are configured to write to a "DataAcq" channel:

```
Set("BL:Mot:X42", 1)
Set("DataAcq", "Start"),
WaitFor("NeutronCount", 1e5),
Set("DataAcq", "Stop")
Set("BL:Mot:X42", 2)
Set("DataAcq", "Start"),
WaitFor("NeutronCount", 1e5),
Set("DataAcq", "Stop")
```

Before taking data as just shown, a preceding table row to can adjust the sample environment The sample environment adjustments need to complete before taking data, but they can be performed in parallel. A "+p " prefix in the column header instructs the PyScanClient library to use parallel commands:

+p T	+p P	BL:Mot:X42	Wait For	Value
30	45			

This table results in the following commands:

```
Parallel(Set("T", 30),   Set("P", 45))
```

The following table describes an experiment that commands two temperature controllers to initial setting, and then takes data while the temperatures are changed to a different value:

+p T1	+p T2	Wait For
30	40	
300	330	Completion

Additional columns could previously adjust the rate-of-change on the temperature controllers to obtain the desired temperature variation over time.

User Interfaces

The primary SNS beam line user interface is CS-Studio, which includes PyDev, a Python development environment. Users can edit, inspect, debug and execute

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Python scripts that assemble and submit scans in CS-Studio.

In practice, however, most users prefer a graphical interface to a script editor. For routine operations, a CS-Studio operator interface panel specifically created for the task allows adjusting the required parameters. When users push a "Submit" button, this triggers a python script to read the parameters from the display, assemble the scan commands, and submit them to the scan server.

All SNS beam lines have so far been able to utilize a common "Alignment Scan" user interface that allows moving once device from start to end, logging an EPICS channel at each point, and finally performs a Gaussian fit to the data.

Table scan support is integrated into the CS-Studio user interface by allowing users to load, edit, save and submit table scans. The editing inside CS-Studio is admittedly limited, so more complex tables are best prepared in dedicated spread sheet programs like Gnumeric [5], and only loaded into CS-Studio for a final review and submission.

While the scan system user interfaces appear similar to those described in [2], they now utilize the PyScanClient library. As a result, it is now easy to extend and combine them. For example, a custom experiment that uses a python script to assemble scan commands can now include an alignment or table scan, while adding additional commands to be executed before and after those operations.

CONCLUSION

The scan system has been very reliable since its introduction in early 2013. It is now used on seven ORNL beam lines (HFIR CG1D, SNS USANS, VULCAN, CORELLI, HYSPEC, VISION, SEQUOIUA) as the main experiment automation interface for beam line staff and visiting experimenters. In case of problems, the combination of scan system console logs and archived control system data [6] has always allowed us to determine which device had timed out or not responded as expected.

We prefer to implement support for routine beam line operations in IOCs. Beam line energy adjustments, ganged operation of temperature controllers, complex multi-motor movements are whenever possible based on EPICS database logic, EPICS sequences, or python-based IOCs, because this allows for more thorough testing, optimization and also long-term archiving of key parameters.

Nevertheless, the introduction of "Parallel" and "Sequence" scan commands makes it very easy to perform timesaving instrument adjustments, or to approximate coordinated movements for ad-hoc scenarios when such prior IOC-based implementation was not possible.

As we update more advanced SNS instrument control systems, we initially assumed that the required automation could only be performed with custom scripting. In practice, we found that the Table Scan abstraction supports most of them. The table itself may no longer be entered by a user but is instead created by a python script, and the PyScanClient library allows such scripts to directly submit the resulting scan.

ACKNOWLEDGMENT

We thank Qiu Yongxiang, Dylan Maxwell, and Guobao Shen for their collaboration in the PyScanClient development.

REFERENCES

[1] X. Geng, X.H. Chen, K. Kasemir, "First EPICS/CSS Based Instrument Control And Acquisition System at ORNL", ICALEPCS 2013, San Francisco, CA, USA (2013).
[2] K. Kasemir, X. Chen, E. Berryman, "CSS Scan System", ICALEPCS 2013, San Francisco, CA, USA (2013).
[3] M. Pearson, "EPICS on SNS Instruments", EPICS Collaboration Meeting, Michigan State University, East Lansing, MI (2015).
[4] https://github.com/PythonScanClient/PyScanClient
[5] http://www.gnumeric.org
[6] K. Kasemir, "Control System Studio Data Browser", PCaPAC08, Ljubljana, Slovenia (2008).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ADVANCED WORKFLOW FOR EXPERIMENTAL CONTROL

D. Mannicke, N. Hauser, N. Xiong, ANSTO, Sydney, Australia

Abstract

GumTree is a java software product developed at ANSTO and used for experimental control as well as data visualization and treatment. In order to simplify the interaction with instruments and optimize the available time for scientists, a user friendly multi sample workflow has been developed for GumTree. Within this workflow users follow a step by step guide where they list available samples, setup instrument configurations and even specify sample environments. Users are then able to monitor the acquisition process in real-time and receive estimations about the completion time. In addition users can modify the previously entered information, even after the acquisition sequence has commenced. This paper will focus on how ANSTO integrated a multi sample workflow into GumTree [1], what approaches were taken to allow realistic time estimations, what programming patterns were used to separate the user interface from the execution of the acquisition, and the future opportunities the approach enables.

INTRODUCTION

Experiments have become more complex and involve different instrument configurations and sample environments. Our aim was to provide scientists with a user friendly application that would simplify the data acquisition process and the associated interactions with the instrument being used as well as help them to optimise the experimental time available.

For example with Quokka, a small angle neutron scattering beamline at ANSTO, it is very common to set up an experiment that requires measurements with 3 configurations, at least 7 samples and may also involve 5 different temperatures. This setup alone results in 105 single measurements which would be very time consuming to manually prepare.

In order to increase the acceptance of the provided solution by the scientific community, scientists and researchers were included in the development process. We asked about their expectations, ideas and experiences at other facilities. We received very different and sometimes contradicting responses. Finally we decided on a solution that delivered a compromise. The end result is suitable for the majority of experiments conducted on the beamlines.

Initially we assumed it would be sufficient to simply loop-over configurations, samples and temperatures in order to generate an acquisition sequence using existing scripting language commands. However, in practice many scientists desired more flexibility; they wanted to be able to adjust the order of the acquisition sequence including duplicate and remove measurements. The scientists also wanted the ability to make these changes after the acquisition sequence had commenced. The interactivity on a running command sequence could neither be implemented with a scripting language 'loop' construct nor with a simple 'command list' or 'command table', which are traditionally generated by unrolling nested loops.

A key requirement from the development perspective is that the system could be deployed and maintained on multiple beamlines with minimal effort. Therefore we had to develop a solution that keeps most of the implementation generic but provides sufficient flexibility to allow a customised user interface to be created for each beamline.

In addition, to improve the robustness it was decided to separate the system into a server and a rich client application. The server was developed to run in a headless mode on hardware with direct communication to the instrument control system. Furthermore, the server had to provide an interface that allows multiple clients to monitor and control the acquisition sequence.

IMPLEMENTATION

For the implementation of the user interface, the workflow was divided into four individual tabs where each tab represents a different aspect of the system (See Fig. 1). The first tab is dedicated to the sample stage; this includes specifying the properties of the used samples as well as the utilized sample holder. The second tab is concerned with setting up the desired beamline configurations which may include different sample to detector distances and aperture configurations. The next tab is concerned with sample environments; this is where a user can specify different temperature ramps or magnetic field strengths. On the final tab, the user is able to start and stop the acquisition as well as control and monitor the acquisition sequence.

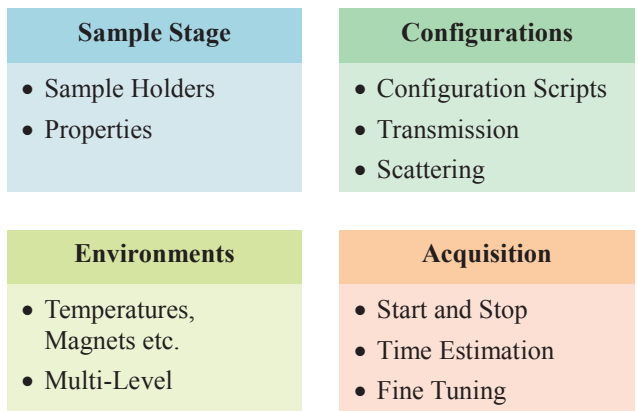


Figure 1: Workflow aspects

In order to keep the system easy to maintain for multiple beamlines we minimized the instrument specific components (See Fig. 2). For example on the server side, only an XML Schema Definition file, which is used as an

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

beamline definition file, is necessary to describe the elements of an acquisition sequence (e.g. Sample, Sample-List, Configuration etc.) including what properties it contains and their relationships. Furthermore, only one instrument specific python script is required that translates the instructions generated from the workflow into commands for the instrument control system. This product can be used on a variety of instrument control systems or hardware abstraction layers.

On the client side, the scientific user is presented with instrument relevant UI elements. To ensure that the client side development is kept to a minimum, a lightweight framework was developed to manage the client/server communication as well as the synchronisation of the client with the model.

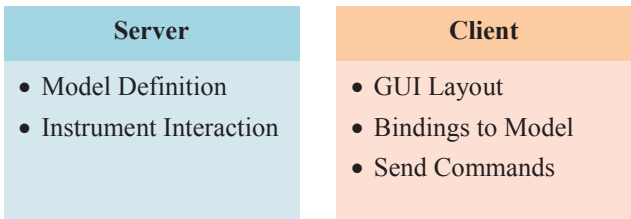


Figure 2: Instrument specific components

To minimize on-going maintenance efforts we ensured that most components are beamline independent (See Fig. 3); this allowed components to be shared across multiple beamlines while still providing the flexibility to cope with different instrument particulars. The beamline independent components on the server include the model database (which can be fully instantiated with the beamline definition file). Furthermore, all algorithms required for the time estimation as well as the communication protocols can be shared across all beamlines.

Figure 3: Beamline independent components

PROGRAMMING PATTERNS

Most influential programming design patterns were the *Model View ViewModel* (MVVM) in conjunction with the *Command* pattern. The MVVM pattern was developed by Microsoft to simplify event-driven programming of user interfaces. It consists of three components: *Model*, *ViewModel* and *View* as well as bindings and commands (See Table 1). The idea is to keep the *Model* and *ViewModel* independent from the code and technology used for the actual *View*. As a result most of the code relating to the *Model* and *ViewModel* can be shared across multiple beamlines and the beamline specific *View* can be easily maintained.

Table 1: Components of MVVM

Component	Description
Model	contains the bare information that describes the current state or content
ViewModel	provides bindable properties and commands and only the ViewModel has direct access to the Model
View	visualises the model via bindings to the properties and commands provided by the ViewModel

For further information, see [2], [3] and [4]; these resources provide great information and guided tutorials.

REALISTIC TIME ESTIMATION

Neutron scattering experiment sequences managed with this product can run for many hours, and are followed by sample changes that may take place at any time of day. Realistic time estimations were therefore a crucial feature for our scientists and thus we wanted to ensure that we are able to provide realistic values. We implemented a statistical approach that is generic enough to be used with different beamlines as well as able to capture the beamline specifics. Furthermore it was necessary to not just provide a single value for the estimation but also the uncertainty associated with it.

Since for us the beamline specific python script is the only code that directly interacts with the Instrument Control System we chose to add a custom profiler (via *sys.settrace*) into the python layer. This profiler is able to log all python calls, track the arguments and record the execution times of each function. Consequently we obtain a call graph with statistical metadata that also includes state transitions of the beamline configuration. This call graph then allows us to execute the code with different sets of input parameters without interacting with the beamline hardware. For a given experiment (which may contain multiple samples and configurations) an accumulated time estimation can be determined.

LESSONS LEARNED

The inclusion of the scientists in the overall design process resulted in positive feedback and ultimately increased their desire to see the project succeed. Involvement was facilitated by conducting review meetings. An agile approach to development was taken which relied heavily on the scientists being involved and providing feedback. This approach resulted in the success of the project and allowed us to understand the needs of the scientists as well as mange their expectations.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

FUTURE

One of the future advancements that the implemented system enables is to use the collected statistics about the instrument response times to provide information about the history of the instrument performance regarding motor movements. This could further be enhanced to automatically notify instrument responsible if, for example, certain motor movements take significantly longer than predicted.

Another future development could involve providing an http web client that can be used to monitor the acquisition sequence via a web browser on a remote computer or smart phone. This would be particularly useful if an experiment takes multiple hours to complete.

CONCLUSION

The advanced workflow described in this paper provides features not implemented previously in beamline workflows. These features include reordering, duplicating and removing tasks from a running workflow and the statistical method used for time estimations.

REFERENCES

[1] T. Lam, N. Hauser, A. Gotz, P. Hathaway, F. Franceschini, H. Rayner, "GumTree, an integrated scientific experiment environment", Physica B 385-386, 1330-1332 (2006)
[2] http://www.codeproject.com/Articles/100175/Model-View-ViewModel-MVVM-Explained
[3] http://www.wpftutorial.net
[4] https://sourcemaking.com/design_patterns/command

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ITERATIVE DEVELOPMENT OF THE GENERIC CONTINUOUS SCANS IN SARDANA

Z. Reszela, G. Cuni, C. M. Falcón Torres, D. Fernandez-Carreiras, C. Pascual-Izarra, M. Rosanes
Siscart, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain

Abstract

Sardana [1] is a software suite for Supervision, Control and Data Acquisition in scientific installations. It aims to reduce cost and time of design, development and support of the control and data acquisition systems. Sardana is used in several synchrotrons where continuous scans are the desired way of executing experiments [2]. Most experiments require an extensive and coordinated control of many aspects like positioning, data acquisition, synchronization and storage. Many successful ad-hoc solutions have already been developed, however they lack generalization and are hard to maintain or reuse. Sardana, thanks to the Taurus [3] based applications, allows the users to configure and control the scan experiments. The MacroServer, a flexible Python [4] based sequencer, provides parametrizable turn-key scan procedures. Thanks to the Device Pool controllers interfaces, heterogeneous hardware can be easily plug into Sardana and their elements used during scans and data acquisitions. Development of the continuous scans is an ongoing iterative process and its current status is described in this paper.

INTRODUCTION

Traditionally, a continuous scan needed an ad-hoc controls software being in charge of the configuration, process control, data acquisition and storage of the whole measurement. Its use was restricted to particular axes, hardware involved to count encoder or motor pulses and generate external triggers or specific experimental channels and detectors. The generalization of this type of scans is strategic in all synchrotrons and potentially in any other laboratory. A generic continuous scan framework provides the flexibility of step scans in a continuous and synchronized data acquisition, allowing time-resolved experiments. The complexity resides in using any arbitrary combination of movable elements, experimental channels, and detectors, generating the required triggers and evidently, allowing the coexistence of slow interpolated software-triggered channels and “virtual” elements like pseudocounters and pseudomotors.

EXPERIMENT CONTROL WITH SARDANA

Sardana has been designed for suiting large installations like a particle accelerator, or smaller such as experimental stations, up to small labs. Different characteristics and necessities of these type of facilities require from Sardana high flexibility and scalability.

In order to speed up the learning curve many Sardana features were inspired on SPEC [5], a complete and powerful software tool, very popular in building control systems for X-ray and neutron experimental stations. But Sardana goes beyond its functionalities. Its architecture is based on the client-server model with Tango [6] as the middleware. This allows to balance the workload in a distributed manner between multiple clients and multiple servers on both Linux and Windows platform PCs.

Sardana had proven to be a solution for synchrotrons, with its successful use in all beamlines, the accelerators and the peripheral laboratories of the Alba synchrotron [7]. It is also used in other similar installations like DESY in Germany, MaxIV in Sweden Solaris in Poland.

Taurus Based User Interfaces

Users can choose between Taurus based command line (CLI) and graphical interfaces (GUI) for interacting with the Sardana system.

From one hand exist turn-key applications. Spock, a single point of access CLI, provides the total control over the system. It allows to run the control procedures and interact with the laboratory elements. Its syntax tries to mimic as much as possible the SPEC commands, what makes the user transition between the two systems easier. A collection of Sardana specific, Taurus based, widgets (many of them as well available as standalone applications) includes: the experiment configuration tool, the scan plots, the user procedure editor, executors and sequencer as well as the elements' control widgets e.g. a generic motor widget.

From the other hand, users may create their own complete Sardana GUIs without the need to program a single line of code. The TaurusGUI framework [3] offers a wizard-based GUI creation process, in which the user just needs to specify which Sardana system the application should interface to and which graphical synoptic should it use. The application gets automatically populated with the previously listed widgets and the instruments' control panels bidirectionally connected with the graphical synoptic. At the application runtime, user can easily navigate in the laboratory by selecting instruments' panels via clicks on the synoptic.

Standard and Custom Experiment Procedures

Experiment and control procedures in Sardana are programmed as Python scripts and are called macros. The MacroServer manages available to the user macros and executes them either sequentially or simultaneously on the user request. Macro development and execution

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

provides many attractive features like the macro input and output parameters, user interaction at runtime, hook extension points, data exchange or plotting.

Sardana provides a miscellaneous set of standard macros including generic, n-dimensional scans available in various modes: step, hybrid and continuous, where distinct actions like motion, data acquisition and data storage are synchronized and optimized. Sardana allows the scan data to be stored in many different formats. This process is handled by one or more optional *recorders*. They receive the scan data and the experiment metadata from the scanning macro and transfer them to the destination e.g. a file, console output or to a data post-processing program.

Unified Access to the Hardware

Sardana interfaces all the equipments via the Device Pool (Pool) server and its plug-in controller classes. Instances of the Sardana controller classes may group many elements of the same type (see Table 1 for the list of the Sardana element types) analogously to what the hardware may do e.g. a single motor controller may control many moveable axes. Each of the Pool elements has a corresponding Tango device what facilitates its control to the Taurus clients.

Table 1: Sardana Element Types and its Examples

Element Type	Example of application
Motor	stepper, servo or piezo actuator
PseudoMotor	energy, HKL of a diffractometer, slit's gap or offset
CounterTimer	event counter, position measurement
PseudoCounter	vertical beam position in the X-ray beam position monitor (XBPM)
0DExpChannel	analog to digital converter (ADC), low current electrometer
1DExpChannel	position sensitive detector (PSD), multichannel analyzer (MCA)
2DExpChannel	CCD camera, 2D X-ray detector

The Pool optimizes common control actions. The grouped acquisitions are handled by the MeasurementGroup (MG) elements, which configure and synchronize the experimental channels previously selected by the user. Grouped movements are also synchronously started and controlled by the Pool's motion action. The API of the controllers and the actions' algorithms allow access the hardware efficiently.

Sardana system may consists of many distributed Pools. This architecture is successfully used at Alba to control the beamline elements in one Pool and the insertion device (ID) elements in another Pool, within the same Sardana system.

DEVELOPMENT APPROACH

Both Sardana and Taurus were conceived as Alba's internal projects. While gaining interest of other facilities, its development and decision-taking was opened to a community of users and developers. Discussions about the critical improvements and modifications in Sardana are organized and formalized around public process called Sardana Enhancement Proposal (SEP).

Design and development of the generic continuous scan framework is organized as SEP6. Alba drives this enhancement in a close collaboration with Desy. An internal to Alba *Scrum* team of 4 developers is in charge of this and other projects [8]. The complexity of the problem requires exploration and learning from experience. The iterative and incremental development allows to periodically deliver an improved version of the continuous scans to the beamlines. The evolving design was adapted to the user feedback and the complete model of the framework is presented in continuation.

The following limitations were assumed at the beginning of the project in order to reduce its scope and be able to deliver a working and quasi-backwards compatible version of Sardana to the laboratories as soon as possible:

- All the elements, but slave motors, participating in the continuous scans must be defined in the same Pool.
- Physical motors maintain uniform velocities while scanning – no trajectory control.
- All the experimental channels present in the MG share the same integration time across the scan point.

GENERIC CONTINUOUS SCAN MODEL

Three main actors participate in the continuous scan measurement and are controlled by their corresponding actions. The moveable objects vary the actuators' set points – motion action. The experimental channels acquire the measurement signals – acquisition action. The trigger or gate synchronizers (either software or hardware) are in charge of controlling the acquisitions based on the moveables' position updates – synchronization action.

Transparency

Users can easily switch between the scanning modes, by simply changing the macro name. Continuous scan names result from adding the *c* suffix to the step scan names e.g. ascan**c**. The order and meaning of the input parameters are preserved unchanged and the scan data outputs and files have identical formats. The same MG definition should be equally valid for both step and continuous scans (if the hardware allows that).

Generic Scan Framework (GSF)

Sardana implements a set of classes, called GSF, responsible for: parameters calculation, configuration,

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

execution and data storage of the scans. Both standard and custom scan macros extensively use the GSF what limits the experiment logic necessary to be programmed in the macro code.

N-dimensional equidistant continuous scans receive as parameters: the motor names, the scan ranges, the number of scan intervals and the integration time per scan point. These parameters are not sufficient to configure the involved elements. In order to calculate the scan parameters the GSF interrogate the involved physical motors for their maximum velocities, acceleration and deceleration times. The active MG and its experimental channels are interrogated for the *wait time* and the *re-arming* times respectively.

The scan is executed and controlled by the GSF in the following order, allowing users to interrupt it at any time:

- Move physical motors to the *pre-start* positions at the nominal speed.
- Configure all elements with the scan parameters.
- Start the experimental channels and later the synchronizers.
- Move physical motors to the *post-end* positions with the adequate parameters.
- When the scan finishes, move the physical motors to the end positions at the nominal speed.

During the scan, GSF progressively receives data updates reported by the experimental channels. They get classified into scan records and passed to the corresponding recorders. In some cases data interpolation may be applied in this process, always preserving information about the original data.

Motion

Any combination of Sardana motors and pseudomotors could be used in continuous scans. The following attributes: acceleration time, velocity and deceleration time are configured in a way that all the motors reach and leave the constant velocity region at the same time. While the acceleration and deceleration times are common to all the motors (based on the slowest motor), the velocities may differ between moveables (adapted to the necessary displacement).

The motion action controls whether the movement has finished and interrogate the motors for the position updates. The frequency of these queries should be configurable per motor due to its impact on the accuracy of the software synchronization.

Measurement Group and Configuration

MG aggregates experimental channels and either software (SW) or hardware (HW) synchronizers. Its role is to manage the configuration and supervise the acquisition and the synchronization actions. Coexistence of both types of synchronizers, e.g. as in the Fig. 1, is highly desired in many experiments.

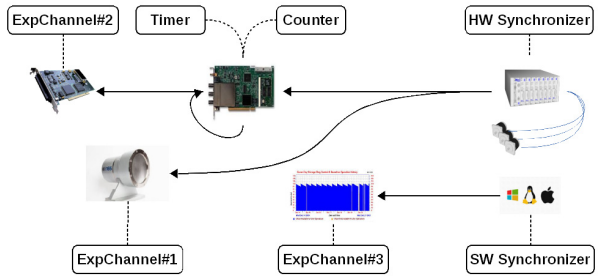


Figure 1: Example of the MG elements involved in the continuous scan.

Configuration of the MG, apart of other parameters, contains all the necessary information to describe the relation between the synchronizers and the experimental channels during the measurement. It is organized in a table with 1-to-1 relation between these elements e.g. Table 2. Single acquisition may be controlled by either *trigger* or *gate* signals. In case of the first one, the experimental channel is notified by the synchronizer when to start and decides on its own (based on the internal integration time control) when to stop acquiring. In the latter case, the experimental channel is also notified by the synchronizer when to stop.

Table 2: MG Configuration Corresponding to the Example from Figure 1.

Experimental channel	**Control**	**Synchronizer**
Timer	Trigger	HW Synchronizer
ExpChannel#1	Trigger	HW Synchronizer
Counter	Gate	Timer
ExpChannel#2	Gate	Timer
ExpChannel#3	Trigger	SW Synchronizer

MG contains other parameters used during the continuous scans: wait time, moveable reference source and synchronization. The first two smartly apply when the software synchronizers are in use. The first one expresses the desired wait time between the two acquisitions. It may help to avoid acquisitions being skipped in case the previous one is still in progress, but have a side effect and slows down the scan. The second one indicates the reference motor which position updates are used by the SW synchronizer to determine whether to emit start and stop signals.

Synchronization

A new Sardana element type TriggerGate (TG) was defined. It represents devices with trigger and/or gate generation capabilities. Their main role is to synchronize acquisition of the experimental channels. The synchronization characteristics could be described in either of two configuration domains: time or position. In the time domain, elements are configured in time units (seconds) and generation of the synchronization signals is based on passing time. The concept of position domain is

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

based on the relation between the TG and the moveable element. In the position domain, elements are configured in distance units of the moveable element configured/connected as the reference source (this could be: mm, mrad, degrees, etc.).

The synchronization data structure is prepared by the GSF and passed via MG to all the involved TG controllers. It is composed from the groups of equidistant acquisitions described by: the initial point and initial delay, total and active intervals and the number of repetitions. Due to the high flexibility of the TG hardware controllers, the synchronization description contains redundant information e.g. initial point and initial delay, always expressed in both: time and position domains. This way the controllers may choose which parameters and in which domain to use.

Acquisition

For the continuous scan needs the experimental channel controllers define the re-arming time (minimum time between the two consecutive HW controlled acquisitions) and whether the data readouts while acquiring are supported.

The acquisition action configures and controls the measurement process. In its start phase the following steps are executed:

- controllers are informed whether SW trigger or gate, or HW trigger or gate will control the acquisition (determined from the MG configuration)
- controllers are configured with the number of acquisitions to be executed
- integration times are loaded to all the channels controlled with the trigger signals
- start sequence is called on all the channels

The loop phase of the acquisition actions proceeds with the following steps for the channels still acquiring:

- call the state sequence in order to determine if the acquisition has finished
- call the read sequence to obtain the new data (if the controllers allow that)

Finally channels which do not support data readouts while acquiring, are interrogated for all the data.

Data Merging

Every value acquired during the continuous scan is stamped with an absolute time and the acquisition index. Data coming from the experimental channels synchronized by hardware provide the core part of the records. The software synchronized channels do not guarantee to provide data for each record. The GSF assigns data into records based on the index. The zero order hold [9] (constant interpolation) method is applied in order to fill gaps left by the skipped acquisitions. The real data must be easily distinguishable from the interpolated one, so each recorder could store/visualize them in its own way.

CURRENT STATUS AND FUTURE PERSPECTIVES

While the design of the generic continuous scan model is complete, the implementation is still ongoing. Its increments are gradually deployed in three Alba's beamlines.

The material science beamline (BL04) executes high resolution powder diffraction experiments (Fig. 2) in the continuous scanning mode. The measurement group involves fifteen hardware triggered counters and an arbitrary number of software triggered 0D experimental channels (mainly used for the sample environment monitoring). Mixing hardware and software synchronization gives a precise information about the sample experiment conditions (e.g. temperature) during the scan. Data post-processing programs calculate max, min and average temperature and exclude scan points based on temperature outliers from a given window.

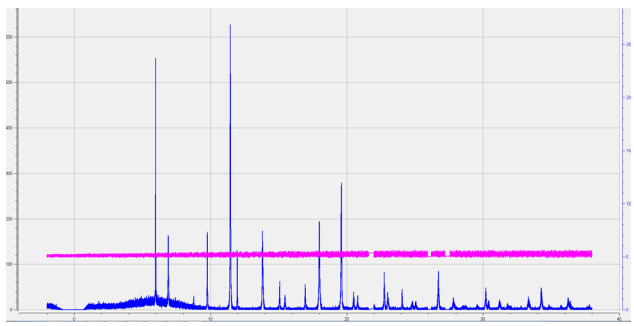


Figure 2: Diffraction patter (blue curve, left ordinate) and temperature (pink curve, right ordinate) measured during continuous scan of the diffractometer outer circle (abscissa) at BL04 – Alba.

The presented model covers the needs of equidistant continuous and time scans. Exchange of the parameters calculation layer in the GSF and use of the multiple groups in the synchronization description will allow non-equidistant scans. It is planned to allow trajectory control by varying physical motors' velocities while scanning.

ACKNOWLEDGEMENT

We would like to thank the Alba beamline scientists – especially François Fauth, Laura Simonelli and Manuel Valvidares for their valuable feedback during the commissioning. We would also like to thank the Controls and Electronics Sections of Alba for their active work in this project: Sergi Rubio, Fulvio Becheri, Roberto Homs, Daniel Roldan, Jordi Andreu and Xavier Serra. We also appreciate the Sardana Community feedback: Teresa Nunez, Thorsten Kracht and Jan Kotanski from Desy, Alejandro Homs, Tiago Coutinho and Jens Meyer from ESRF.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REFERENCES

[1] T. Coutinho et al. "Sardana, the Software for Building SCADAs in Scientific Environments", WEAAUST01, ICALEPCS2011, Grenoble, France, (2011)

[2] D. Fernandez-Carreiras et al. "Synchronization of Motion and Detectors and Cont. Scans as the Standard Data Acquisition Technique", WECOAAB03, ICALEPCS2013, San Francisco, USA, (2013)

[3] C. Pascual-Izarra et al. "Effortless Creation of Control & Data Acquisition Graphical User Interfaces with Taurus", THHC3O03, ICALEPCS2015, Melbourne, Australia, (2015)

[4] Python website: http://www.python.org

[5] SPEC website: http://www.certif.com/spec.html

[6] Tango website: http://www.tango-controls.org

[7] Alba website: http://www.albasynchrotron.es

[8] G. Cuni el al. "Introducing the SCRUM Framework as Part of the Product Development Strategy for the ALBA Control System", MOD3O04, ICALEPCS 2015, Melbourne, Australia, (2015)

[9] Wikipedia article "Zero-order hold": https://en.wikipedia.org/wiki/Zero-order_hold

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE MODULAR CONTROL CONCEPT OF THE NEUTRON SCATTERING EXPERIMENTS AT THE EUROPEAN SPALLATION SOURCE ESS

T. Gahl, R. J. Hall-Wilton, O. Kirstein, T. Korhonen, A. Sandstroem, I. Sutton, ESS Lund, Sweden
T. R. Richter, J. Taylor, ESS Copenhagen, Denmark

Abstract

The European Spallation Source (ESS) in Lund, Sweden has just entered into neutron beam line construction starting detailed design in 2015. As a collaboration of 17 European countries the majority of hardware devices will be provided in-kind. This presents numerous technical and organizational challenges for the construction and the integration of the neutron instrumentation into the facility wide infrastructure; notably the EPICS control network and the facilities absolute timing system.

In this contribution we present a strategy for the modularity of the instruments hardware with well-defined standardized functionality and a minimized number of control & data interfaces. Key point of the strategy is the time stamping of all readings from the instruments control electronics extending the event mode data acquisition from detected neutron events to all metadata. This gives the control software the flexibility necessary to adapt the functionality of the instruments to the demands of each single experimental run. We give examples for the advantages of that approach in classical motion control as well as in complex robotics systems and discuss matching hardware requirements necessary.

INTRODUCTION

The European Spallation Source (ESS) is designed as a long pulse neutron source with the maximum overall flux as its main objective. With a 5MW linear proton accelerator it will be the world's most powerful source of neutrons for science applications. Integrated flux levels will be much higher than existing facilities and the geographical layout will comprise instruments of 160m and longer spanning over 3 instruments halls. At the same time the experimental end stations (instruments) for these kinds of sources need to be more flexible and complex with up to 20 choppers along the beam line. All this presents operational challenges that are best addressed with a good mixture of techniques from existing neutron facilities and other disciplines like x-ray experiments, fundamental physics or industrial applications. Advanced grounding concepts, high rate data acquisition, flexible experimental setup, advanced remote diagnostics tools, pre-emptive maintenance are a few keywords for that.

At the same time the ESS project represents also an organisational challenge with most of the instruments hardware provided in-kind by the 17 European partner countries. Integration of these contributions either as single instruments components or complete instruments into the ESS infrastructure is only possible with a flexible and modular instrument control system. This also supports clear definitions of functionalities and interfaces of the modules. We will present the evolution of our ideas of such a system first formulated in [1] following the current design phases and addressing technical, operational and organisational challenges of the future ESS.

ESS NEUTRON BEAM

Neutron Beam Characteristics

ESS is designed as a long-pulse neutron source with 14Hz repetition rate and 4% duty cycle resulting in 2.86 ms pulse length [2]. With an average power of 5MW of the proton accelerator it will exceed existing neutron sources - either steady state or pulsed - in average neutron flux as well as in peak brightness.

Neutron Beam Development

The energy spectrum of the neutron pulse will range from thermal to cold neutrons with travelling speeds between 2000 and 200 m/s. After a certain travel distance in the neutron guide the pulse will transform into a constant beam modulated in time by neutrons with different energy (Figure 1). Mechanical choppers are used to form a slit system in the time domain to enhance the energy resolution of the resulting neutron beam and/or to limit the spectrum used for the experiments. The exact configuration is determining the length of the instruments ranging from 30 to 160m.

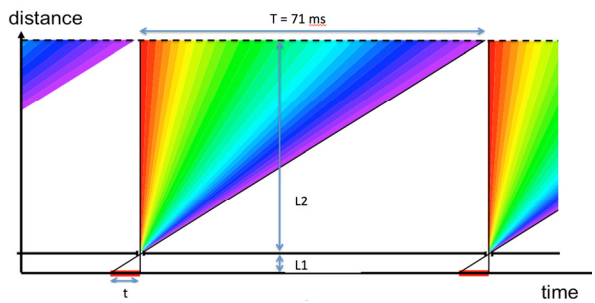


Figure 1: Time-distance diagram of beam development.

Due to the time characteristics of the final neutron beam at the experiments synchronisation and referencing of all experimental data to the proton pulse is a key requirement of instruments control and data acquisition at ESS.

ESS CONTROL INFRASTRUCTURE

EPICS Control System (ICS)

ESS will use EPICS as a global control system for instruments, machine (accelerator, target), central infrastructure and part of conventional facilities. It acts as a

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

horizontal device layer for data exchange between the different parts of the facility although for instrument control and operation the main signal flow will be vertically between the instruments hardware below and the user interfaces on top of the layer (Figure 2). All ESS hardware links to the EPICS layer by means of standardised input/output systems (control boxes). The Integrated Controls System Division (ICS) will provide control boxes, EPICS network and hardware driver all over the facility. This ensures a high degree of standardisation and synergy within ESS.

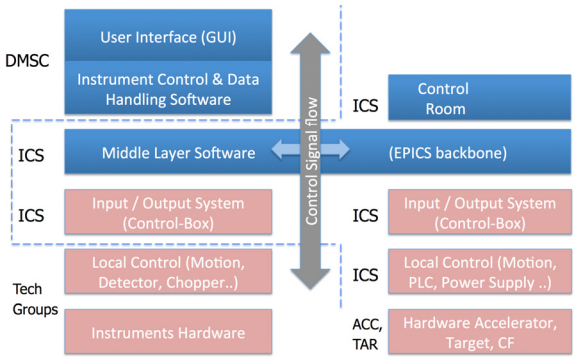


Figure 2: ESS facility wide control system layers (blue: software, rose: hardware).

User Interface (DMSC)

All user interface scripts/GUIs as part of the software layers on top of EPICS are provided by the Data Management and Software Centre (DMSC) of ESS Copenhagen. The entire control loop of the experiment including scientific calculations, coordinate transformations, sequencing, local experiment monitoring, data aggregation and all user interface is handled within this layers. This includes data acquisition and data reduction for monitoring purposes as well. The DMSC software interfaces to EPICS and for large data volumes directly with the instruments hardware.

It should be noted that nothing is configured locally 'on the instrument'. This makes the DMSC interface the only access for the instruments users; there is nothing like a local monitoring at the instrument. The only local systems are for expert diagnostics and commissioning. The instrument cannot perform normally (even in a limited fashion) without DMSC supervision.

Timing Network and Signals (ICS)

ESS is using a centralised absolute timing system that is provided by ICS [2]. A generator is distributing the clock of a master oscillator and the absolute facility time to a number of timing receivers through a dedicated optical fibre network. A transport layer solution presented by Micro Research Finland [3] is envisaged as technical solution. Per default a timing receiver is present in every control box and thus makes the functionality available to all hardware connected to EPICS. Delay times in the network are compensated in steps of 11.357ns (equals 2.4m fibre length) once the exact position of the receiver and thus the length of the fibre are fixed.

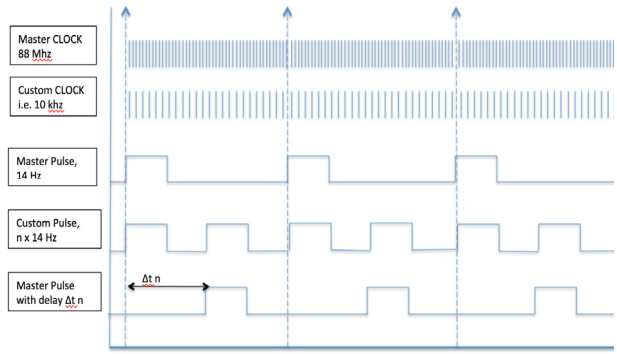


Figure 3: Set of synchronous strobe signals available at the timing receiver interface.

Timestamps are global, across the entire facility, although the resolution of the time stamp used can be optimised in each instance; the highest resolution is 11.357ns. Coordinated synchronous strobes are available at the timing receiver interface (Figure 3) to facilitate coordinated synchronous control across instrument subsystems. So, for example, instrument run control can be coordinated to any required precision. Signals include 88.0525Mhz master clock, custom clocks in fractions of the master, 14Hz master pulse, and any custom pulses in multiples and fractions of 14 Hz; all synchronised with jitter requirements as low as 1ns and possibly delayed to the master pulse in multiples of 113,57ns.

INSTRUMENT CONTROL AND READOUT CONCEPT

Modular Concept

The control and readout model that has been developed for ESS instruments is modular, where an instrument is made up from a number of independent subsystems ("modules") that do not interact with each other, but only centrally through the EPICS and DMSC interfaces. Each module simply takes care of it's own responsibilities, collecting data (which can be neutron or meta data) or controlling some physical configuration of the instrument (motion control, chopper speed and phase, magnets, etc.) or often both. These systems present the data in the natural form for the module – data is time-stamped, but the significance of the timing cannot be realised until the data is combined at the DMSC. Similarly, detector data is formatted in the natural units of the detector (channel number etc.) rather than physical units (e.g. position) so that the local readout systems do not have to change when an instrument is reconfigured.

Control Topology

Backbone of the ESS instrument control is the EPICS and Timing network of the ICS with the control boxes as interface points (grey, Figure 4). Physically there will be one EPICS subnet per instrument with connectivity to the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

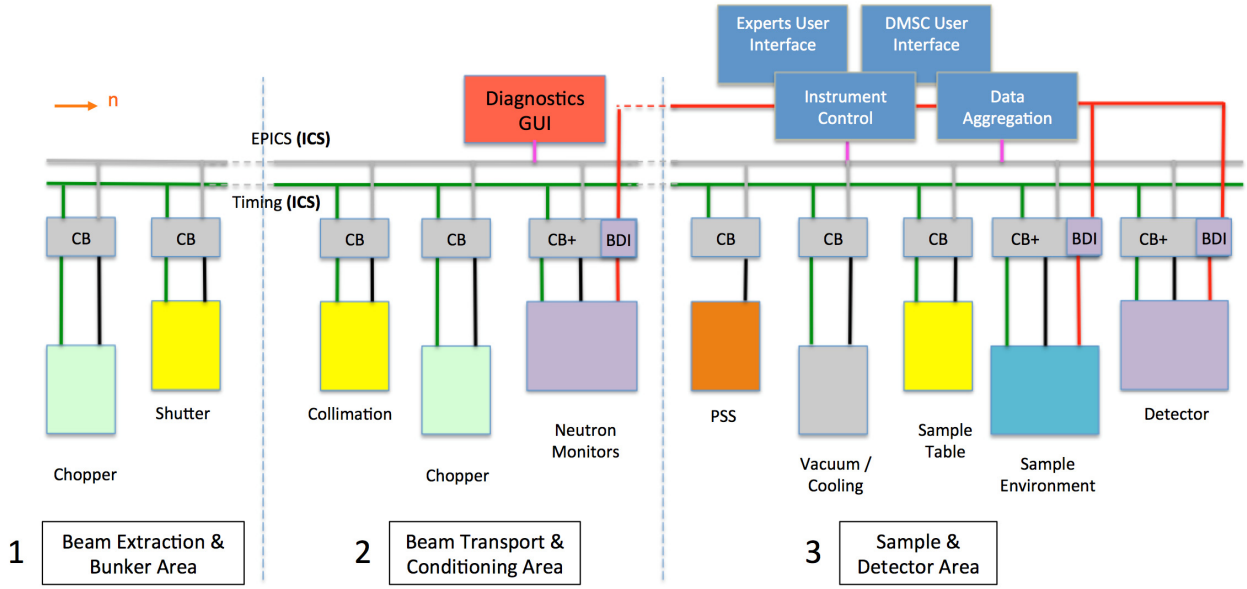


Figure 4: Topology of the modular instrument control concept for a 160m long Neutron Scattering Instrument at ESS. (CB = Control Box, BDI = Bulk Data Interface).

whole facility. All instruments subsystems (different colours according to functionality) link to these boxes and could be clustered in electrically isolated groups following the geographical layout of the instrument in the different instrument halls. The user interface of the DMSC (blue) is connecting to the EPICS layer via a data gateway. Modules with larger volumes of data like detector electronics or fast sample environment require a dedicated link to a DMSC aggregator node called a bulk data interface (BDI, violet, Figure 3). The red box and the blue "Experts User Interface" indicate engineering user interfaces on the different layers for diagnostics purposes.

Design Principles

Every control process that can be done (or has to be done) locally in one of the modules will be done locally, every signal that needs to be related to data from other modules has to be time stamped and sent to the DMSC user interface. That gives clear functionalities and clear interfaces for single modules, easy to specify and to bring in as in-kind contributions or from external suppliers. All modules are linked together by the instrument control/ data acquisition software of the DMSC (via the EPICS layer); it's a crucial part of the instrument and will be tailored to the needs of each instrument. It gives the flexibility necessary to adapt the same hardware to each of the ESS instruments and to future instrument upgrades: the hardware is done once and prepared for all future (hi-level) functionalities.

However care must be taken about the latency in collecting the data from all modules together to allow the formation of a full 'frame' of neutron data being presented to the user for monitoring purposes. We will address this by introducing a maximum tolerated 'latency budget' as a design requirement for the whole instrument control system.

MODULAR INTERFACE AND FUNCTIONALITIES

Each control module basically has two or three interface types to the upper device layers: Timing, command interface and - if necessary - an interface for larger volumes of data.

Command and Meta Data Interface

The command interface connects through the control box to the EPICS network and is able to transfer smaller volumes of instruments metadata through this channel as well. This data is bridged via gateway to the DMSC user interface 'off instrument'. Currently the data rate is limited to about 1kB/s but improvements of data rates in later versions of EPICS (version 4) will be considered. Physically this command/meta data connection is mostly a serial interface like RS232 or Ethernet with latencies between 20 and 100 ms.

Time Stamping

Following the modular control concept described before it is mandatory to timestamp all instruments data. This could easily be done by the timing receiver in the control box but is limited in precision by the latency of the connection between the modular control unit and the control box. For higher precision it is necessary to transfer the absolute time information from the control box to the control electronics of the module (or transfer the whole timing receiver). In Figures 5 and 6 we describe ideas of synchronising internal timing clocks in control modules with the absolute time of the timing receiver. Either a pulse from the control box is synchronising a timer in the control unit (motion control, chopper control) or a strobe is sent from the control module to the control

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

box and time stamped there (detector electronics). In all cases the absolute time of the event is sent to the module via the command interface where it can be related to the internal time of the module. Within the control module the absolute time information can be transferred further by means of real time field busses (e.g. ETHERCAT) allowing also very specialised motion controllers e.g. for robotics or hexapods to be integrated seamlessly into the synchronised control environment.

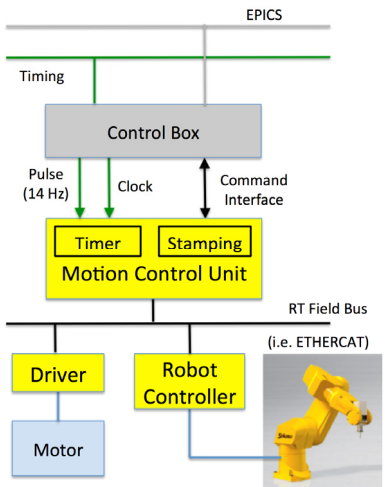


Figure 5: Interfaces for a motion control module

An elegant and high performing solution would be to transfer the timing information directly through a field bus coupler acting as the timing master of the distributed clock system of the real time field bus, although appropriated modules are not yet on the market.

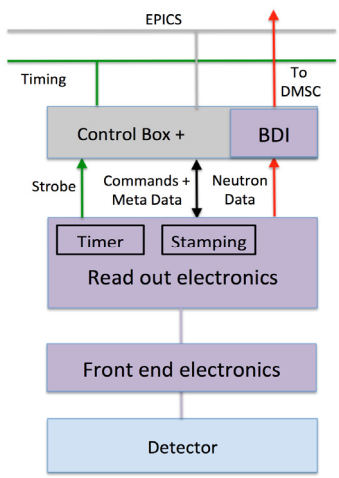


Figure 6: Interfaces for a detector readout module.

Detector Readout and Bulk Data Interface (BDI)

The readout electronics for detectors will be based on FPGAs that will perform data collection and transmission, and necessary (i.e. unavoidable) data reduction or matching. Although ESS has a high flux, the data rates are small compared to particle physics or radio astronomy and we are not too constrained by bandwidth concerns. We anticipate operating the BDI links at 10Gb/sec, but 40Gb/sec links are already readily available and affordable. To generalise, we expect to include basic back-ground suppression in firmware, and cross data flagging to ensure all relevant data is output, but processing such as cluster centroid finding or time based data matching would be performed in software.

CONCLUSION

With the proposed modular instrument control concept we address the demanding challenges for future Neutron Scattering Experiments. Instrument subsystems are simpler - easier to specify and easier to maintain. With clear defined functionalities and interfaces they are easy to bring in as in-kind contributions or from external suppliers. The proposed concept fits exactly in the topology of the ESS wide control infrastructure thus profiting from standardisation and maximising synergies within the whole ESS project.

In some cases the concept of an extended event mode data acquisition represents additional hardware requirements e.g. for time stamping or data transfer. Data rate and latency requirements have to be addresses carefully in the design phase. But once the hardware is standardised and in place the subsystems are prepared for the majority of the future functional and operational upgrades. The functionality of the whole system depends solely on the DMSC user interface software that will start with simple standard applications, introducing gradually more demanding functionalities in future upgrades.

Already now the concept is supporting advanced use cases for experiments and diagnostics like continuous scans, stroboscopic data acquisition or advanced neutron beam diagnostics. A lot more will follow in the future once the ESS has started commissioning and user operation.

ACKNOWLEDGMENT

We would like to thank our colleagues and collaborators in and outside ESS for the lively and fruitful discussions and the contribution of ideas and possible solutions with a special mention of ISIS, PSI, JCNS and CERN.

REFERENCES

[1] T. Gahl et al., “Hardware Aspect of an Event Mode Data Acquisition and Instrument Control for ESS“, proceedings ICANS XXI, Mito, Japan (2014)
[2] S. Peggs et al., “ESS Technical Design Report”, European Spallation Source, Lund, Sweden (2013)
[3] J. Pietarinen, “MRF Timing System”, Timing workshop CERN, Geneva, Switzerland (2008)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE LMJ SYSTEM SEQUENCES ADAPTABILITY (FRENCH LASER MEGAJOULE)

Y.Tranquille-Marques, J. Fleury, H. Graillot, CEA, CESTA, Le Barp, France
N. Chapron, S. Bailleux, AREVA TA, Le Barp, France
J. Gende, ALTEN, Mérignac, France
J. Nicoloso, CEA, DIF, Bruyères le Châtel, Arpajon, France

Abstract

The French Atomic and Alternative Energies Commission (CEA : Commissariat à l'Energie Atomique et aux Energies Alternatives) is currently building the Laser MegaJoule facility. In 2014, the first 8 beams and the target area were commissioned and the first physics campaign (a set of several shots) was achieved. On the LMJ, each shot requires more or less the same operations except for the settings that change from shot to shot. The supervisory controls provide five semi-automated sequence programs to repeat and schedule actions on devices. Three of them are now regularly used to drive the LMJ. Sequence programs need to have different qualities such as flexibility, contextual adaptability, reliability and repeatability. Currently, the calibration shots sequence drives 328 actions towards local control systems. However, this sequence is already dimensioned to drive 22 bundles, which will lead to manage almost 5300 actions. The presentation introduces the organization of the control system used by sequence programs, the sequence adjustments files, the GRAFCETs of sequences, the GUIs, the software and different tools used to control the facility.

LASER MEGAJOULE SHOT OVERVIEW

The LMJ facility will count 176 laser beams. It is dimensioned to deliver 1.4 MJ shot of UV laser at 351 nm to a 10mm target. The LMJ is designed to study high density plasma physics. It's also a part of the French Simulation Program that forms the basis of the safety and reliability of French nuclear weapons.

The LMJ facility is divided in 4 laser bays. In its center, takes place the target bay. A shot is composed of laser pulses which are amplified in the laser bays and focused on a target in the target bay.

A timing shot that lasts 1s, consists in switching on the laser amplifiers lamps, creating, shaping and pre-amplifying the laser pulses, sending it through the amplification section to fill up, converting laser infrared pulses into laser UV pulses. Then, the energy heats the target and the diagnostics record the experiment radiations data. [1]

Shot sequence manages shot preparation steps, timing shot and post shot step.

First comes facility preparation step. It does not include time critical actions.

The preamplifiers perform several triggers to ensure delivered energy repeatability. If needed, inserters carrying diagnostics move at that time.

Then we have the Alignment step. Each laser beam must hit the millimeter target after hundreds meters propagation. Different areas get their own alignment: Amplification sections, Transport Sections, Frequency converter Sections and Target Chamber.

Once alignment is finished, a whole test without Power conditioning is performed: the validation shot or rod shot. It checks that all the required equipment is activated without wasting the target. This equipment is configured and triggered, laser beam measures are analyzed by automatic data processing.

When the validation shot has passed, power shot repeats the validation shot, adding the power amplifiers charge leading to timing shot.

After the shot, amplifier flash lamps test is performed while other devices are secured. Results are stored in a shot database. This is post shot step.

CONTROL SYSTEM

Data Model: Key Words

The main concept of the data model is called a "resource". A resource represents an equipment (motors, instrumentations, diagnostics…) or a high level function (alignment, amplification section, power conditioning bundle). LMJ counts 200 000 resources.

A resource owns a name and attributes:

- Reserved : to forbid unexpected use,
- Context: resource membership of a perimeter. A perimeter is a group of resources existing for the same purpose such as sequence.
- Operational State : Running, Standby or Stop

A resource can be attached to other objects:

- Functions : what a resource can do,
- Control points : resource information,
- Alarms: asynchronous resource events.

Architecture

The LMJ control system architecture is built on a 4 layers architecture as shown in Fig. 1:

- Level 3 system supervisory provides facility planning and operation functions which are dedicated to the shot director
- Level 2 consists of the supervisory system (SVP) which provides GUI to the shot director. As system sequences are SVP programs, system sequences

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

drive subsystems through SVP interfaces with supervisory subsystems.

- Level 1: supervisory subsystems that allow operators to drive facility subsystems (capacitor banks, preamplifier, alignment control system…)
- Level 0: equipment controls (PCs or PLCs).

All the command control software developed for the supervisory layers uses a common framework based on the industrial PANORAMA E² SCADA from Codra society. This framework includes resource management, sequence management and configuration management.[2] [3]

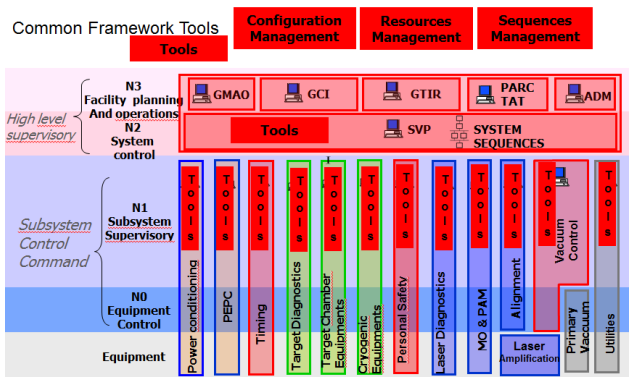


Figure 1: LMJ control system architecture.

Resource Manager

The resource manager provides the ability to link resources with different kind of relationships in order to describe the whole structure of the facility. A resource may be composed of other resources or a resource may use other resources. Such links make perimeter definition easier. For example, working with an 8 beams bundle (resource called Bxx), required to add Bxx resource to the perimeterF ; then the distributed resources manager analyses links and automatically inserts to the perimeter resources that compose Bxx… and so on with every resource that is found and filled in this perimeter.

Configuration Manager

The configuration database contains resources and functions default settings. The configuration manager duplicates for each perimeter the configuration database restricted to the sequence perimeter, adding settings from the sequence adjustment file.

Sequence Manager

Sequences are driven by a GRAFCET. In each step of the GRAFCET, actions are implemented in VB Script. For instance, an action may be a call to another sequence (subsequence) or function. The sequence engine executes the GRAFCET diagrams and the scripts included into the sequence.

The sequence is suspended if an alarm is encountered on a resource that belongs to the sequence perimeter. During suspension, the shot director has the ability to solve the problem or to abandon the default resource, before going on. He can also choose to abandon the sequence itself.

Sequencer debug mode allows executing the sequence GRAFCET step by step, monitoring its execution, debugging VB scripts and inspecting SCADA object properties.

GRAFCET

The shot sequence main GRAFCET splits into several steps. The two first steps consist of the Sequence Program Initialization: shot database initialization, Resources Lock (reservation), GUI's are raised to the shot director in order to check sequence adjustments. Next five steps have been described in the shot overview: facility preparation, alignment, validation shot, power shot and facility securing. Last step is always executed. It drives actions to ensure facility safe state mode before exit the main sequence.

Transitions take place between steps. Moving from one step to another requires shot director authorization. Transitions are also implemented to authorize restarting a main step.

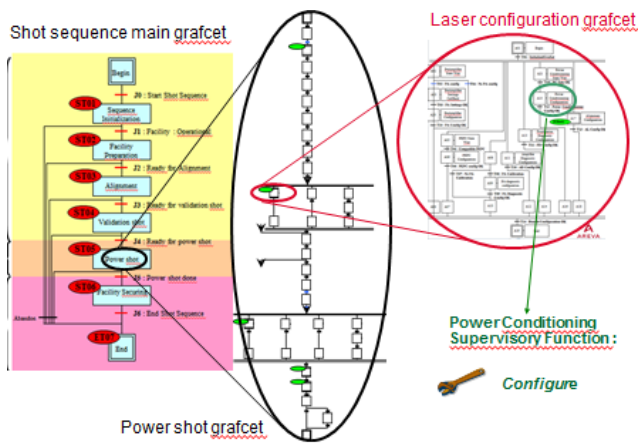


Figure 2: Subsequence calls.

Programming Rules

For readability, one development rule consists in implementing only one action per step.

In shot sequence example as shown in Fig. 2, step 5 manages the power shot and it counts a lot of actions. To respect the above-mentioned rule, the step 5 VB script is a call to the Power Shot subsequence.

In the same way, in the power shot subsequence one step is a call to a laser configuration subsequence because it counts many actions too.

In the laser configuration subsequence, one step is the Power Conditioning configuration function call. This

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

function is run by the Power Conditioning supervisory controls.

GRAFCET Behavior

When a resource state changes to the default state, the resource generates an alarm. If this default resource belongs to sequence perimeter, the sequence manager suspends the main GRAFCET.

Sequence programmer implements suspension behavior concerning subsequences and functions in each GRAFCET program. Most of the time, function suspension is time consuming. Only functions that operate iterative actions are worth being suspended. In system sequence, suspension usually propagates to subsequence and does not to functions.

System Sequences

At that time, 5 system sequences are available:

- Shot sequence: it drives a power conditioning amplified shot sequence on the target.
- Calibration sequence: it drives a power conditioning amplified shot sequence without any target. Calorimeters are inserted through the beams to calibrate laser energy.
- Laser bay non amplified shots sequence is dedicated to numerous repetitions of non-amplified shots in laser bay.
- Target bay non amplified shots sequence is dedicated to numerous repetitions of non-amplified shots in the target bay.
- Target bay equipment calibration sequence is dedicated to inserters positioning calibration.

	Calibration sequence	Shot sequence	Laser Bay non amplified shots sequence	Target Bay non amplified shots sequence	Target Bay equipment calibration sequence	Total
Number of Grafcets	21	31	14	27	13	106
Number of Functions	68	107	43	86	14	318
Number of Actions	267	400	162	325	77	1231

Figure 3: Sequences size.

The shot sequence is the most complex sequence as it almost uses all the facility resources. The 31 GRAFCET of the shot sequence lead to 107 function calls. Figure 3 shows a main difference between the number of function calls (107) and the number of actions (400). Functions may be considered as equipment actions. Actually, GRAFCET management actions, control point check actions and GUI display actions represent a predominant part of the sequence programs.

Calibrating the LMJ and making all kind of shots should need 18 system sequences. Sequence design and adjustment policy are combined to reduce the number of sequences, and thus to reduce maintenance costs. For instance, Calibration sequence can be adjusted to be used for laser energy calibration in amplification section or in focusing section.

ADJUSTMENT

Sequence main goal is function executing. From one shot to another, function settings are different.

Sequence Model File

To define what kind of settings is expected by a sequence, each sequence owns a sequence model file. The sequence model file includes identification, supervisory list, main resource classes, excluded resource classes, resource class priorities and settings names ordered by supervisory subsystems, by sequence main steps and by function classes.

Sequence Adjustment File

Before the beginning of the shot sequence, the shot database manager combines the sequence model file to shot requirements. With PARC help, the Automatic Settings Prediction System [4], the shot database manager builds the sequence adjustment file. This file gives settings values to each function which resource belongs to the perimeter.

Adjustment files contain instances instead of classes. It is shaped like sequence model file:

- Program identification.
- Supervisory list adjusts SVP to address settings to subsystems.
- Main resources list.
- Excluded resource.
- Resource priorities
- Settings value ordered by supervisory, by sequence main step and by function.

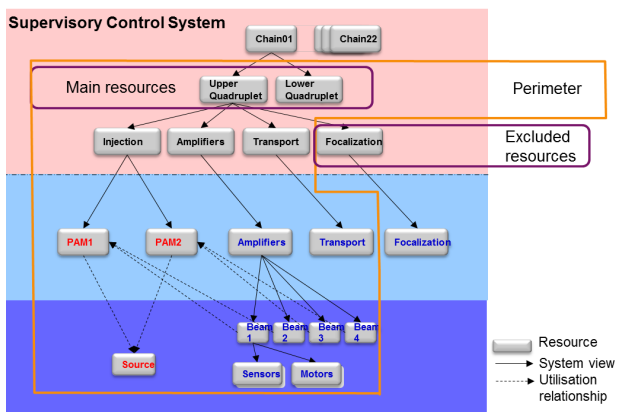


Figure 4: Resource tree and perimeter concept.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Most of the time, main resources are the 4 beams bundles, named “quadruplet”. The facility has got 44 quadruplets. For each quadruplet resource class in the model file, the shot database manager can determine which quadruplets resources are required.

For instance, considering upper and lower quadruplets from the chain number 01 as main resources, sequence perimeter is built thanks to these 2 resources. Resource managers follow resource links to add resources to the perimeter automatically. Resources that are not required appear in the excluded resources list. And resource managers automatically adapt the perimeter as shown in Fig. 4.

Function Adjustment

The last items of sequence adjustment file are settings value with a specific filing. It is explained by subsystem settings reading upon sequence main steps.

For example, considering that one subsystem has a “*configure*” function that has to be adjusted by a “*shot type*” setting. Considering also that the adjustment file defines values for both 4 and 5 main steps. In step 4, it is equal to “*Validation*”. In step 5, it is equal to “*Power*”.

Before starting the sequence, SVP sequence manager performs « *sequence injection* ». It consists in perimeter creation and setting distribution to subsystems by splitting the adjustment file, it ends by a sequence GRAFCET “*start*”. Perimeter creation is executed by resource managers. Settings loading is executed by configuration managers that create contextual configurations restricted to the perimeter. At the end of the injection, a “*shot type*” setting value is set to a default value.

When GRAFCET main step 4 is reached, the sequence manager notifies the current step to subsystems sequence managers. Next subsystems update their “*shot type*” settings to “*Validation*”. When step 5 is reached, the sequence manager notifies the subsystems sequence managers again and the “*shot type*” setting changes to “*Power*”.

SHOT SEQUENCE GUI

Sequence GUI development is part of sequence program development. All system sequence GUIs have the same layout. Figure 5 presents the Shot sequence GUI. After Injection, the shot director drives shots using this window.

GUI philosophy is to control actions in progress. It means to be able to control each running actions state. Colors indicate actions state: red is *error*, green is *done* and *correct*, yellow is *in progress*, white is *inactive*. Progress bars indicate the remaining time when actions last more than a few seconds.

The GUI splits into 7 areas. On top, the first area indicates file adjustment name, sequence name, general information about main GRAFCET state and progress. Under the first area takes place the main step bar. It indicates complete main steps and the running one.

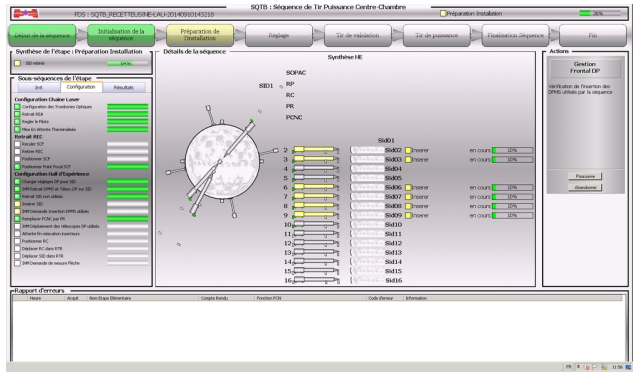


Figure 5: Shot sequence GUI.

On the top left, the third area details progress information of the running main step. On the left, fourth area presents all the actions that occur during the main step execution. In the middle, the fifth area details one action. For instance in figure 5, GUI displays progress information and the state of all the started « *insert* » function as they result from the same “*insert*” function class. In this case, the sequence manager provides the ability to call all these functions at once.

In area 4, « *Insert Inserters* » action state is a synthesis of the 6 run inserters function states. If one indicates an error then synthesis is in error. Else, if all of them are complete and correct, synthesis is complete and correct. Else, if one is running, synthesis is running. Else, if all are inactive, synthesis is inactive. This synthesis construction highlights potential difficulties: errors and remaining actions.

On the right, area 6 is dedicated to shot director authorizations to carry on with sequence. On the bottom, area 7 shows errors that occur during one main step.

SUMMARY

Adjustment file eases resources addition to sequence perimeter and GUI synthesis philosophy does not depend on resource number. Consequently, system sequences are ready for the next LMJ bundles commissioning to come.

REFERENCES

[1] TUD3I01, The LMJ Target Diagnostics Control System Architecture, by S. Perez CEA/DIF, Bruyères le Châtel, F-91297, Arpajon, France and by T. Caillaud, CEA/CESTA, Le Barp, F-33114, France

[2] FRA3O02, The Laser Megajoule Facility Control System Status Report, ICALEPCS 2015, by J. Nicoloso, CEA/DIF, F-91297 France.

[3] “Configuration and sequencing tools for the LMJ control system”, by J.Nicoloso, J.-J.Dupas, J.-C. Picon, F. Signol,

[4] MOC3O06 : PARC : How to computerise Laser Setting on the Megajoule Facility, ICALEPCS 2015, by S. Vermersch, CEA/CESTA, F- 33114 France.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MACHINE PROTECTION AND INTERLOCK SYSTEMS FOR LARGE RESEARCH INSTRUMENTS

R. Schmidt, CERN, Geneva, Switzerland

Abstract

Major research instruments such as accelerators and fusion reactors operate with large amount of power and energy stored in beams and superconducting magnets. Highly reliable Machine Protection systems are required to operate such instruments without damaging equipment in case of failure. The increased interest in protection is related to the increasing beam power of high-power proton accelerators such as ISIS, SNS, ESS and the PSI cyclotron, to the large energy stored in the beam (in particular for hadron colliders such as LHC) and to the stored energy in magnet systems such as for ITER and LHC. Machine Protection includes process and equipment monitoring, a system to safely stop operation (e.g. dumping the beam or extracting the energy stored in the magnets) and an interlock system for highly reliable communication between protection systems. Depending on the application, the reaction of the protection function to failures must be very fast (for beam protection systems down to some µs). In this presentation an overview of the challenges for protection is given, and examples of interlock systems and their use during operation are presented.

INTRODUCTION

Accelerators, as all other technical systems, must respect some general principles with respect to safety and protection. Protection of people from different threats (radiation, electrical, oxygen deficiency, ..) has always the highest priority and follows legal requirements. The main strategy to protect people during operation is to keep them away from hazards, ensured by a personnel access system. Protection of the environment is the second priority. In this paper the protection of equipment (the investment) is discussed, e.g. accelerators and beam targets, experiments and fusion reactors. Designing a machine protection system is challenging and requires an excellent understanding of the system and its operation, to anticipate and avoid possible failures that could lead to damage or mitigate the consequences.

Protection of accelerators was to topic of a recent Joint Accelerator School, the proceedings with many relevant contributions will be published by the end of 2015 [1].

In general, risks come from energy stored in a system (measured in Joule) as well as from power when operating the system (measured in Watt). Particle accelerators and fusion reactors are examples of such systems, since they operate with large amount of electrical power (from a few to many MW). The energy and power flow needs to be controlled. An uncontrolled release of energy or an uncontrolled power flow can lead to unwanted consequences such as damage or activation of equipment and loss of time for operation.

MACHINE PROTECTION AT LARGE RESEARCH INSTRUMENTS

Many accelerators operate with high power beams or beams with a large amount of stored energy. Accelerators operating with superconducting magnets require sophisticated systems to protect the magnets in case of a quench.

Accelerators with large stored energy in the beams are hadron synchrotrons and colliders (Figure 1), e.g. LHC, RHIC, SPS and FCC (a study for a proton collider with a circumference of 100 km and 100 TeV cm energy, [2]),

Many proton accelerators operate with high power beam, such as the PSI cyclotron, ISIS, SNS, JPARC and ESS (planned to start operation in 2019 [3]). A particular challenge are future ADS machines (Accelerator Driven Spallation) that require to operate with very high beam power and extremely high availability [4]. A prototype is expected for the next decade.

Machine protection is also relevant for some electron accelerators, e.g. free electron lasers (XFEL), synchrotron light sources, e+e- circular and linear colliders. In particular protection at linear colliders is challenging, with MW beam power and tiny beam spot sizes.

Installations with large superconducting magnet systems are also considered, e.g. fusion reactors such as ITER operate with magnets that store more than 50 GJ of energy.

To get an idea what this amount of energy means some examples are given. The energy of pistol bullet is about 500 J, the energy of 1 kg TNT about 4 MJ. The energy of 1 l fuel is about 36 MJ, to melt 1 kg of steel about 800 kJ are required (the energy to melt 1 kg of copper is similar). An accidental release of an energy above one MJ can cause significant damage. With the energy stored in the ITER toroid magnet it is possible to melt 60 tons of copper. Even an accidental release of a small amount of energy in the order of some hundred Joule can lead to some (limited) damage if the energy is released in sensitive equipment.

HAZARDS AND RISKS

Definition of Risk

A hazard is a situation that poses a level of threat to the installation. Hazards are dormant or potential, with only a theoretical risk of damage. Once a hazard becomes "active" it becomes an incident or accident. An accident is defined as an unfortunate incident that happens unexpectedly and unintentionally, typically resulting in damage or injury.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

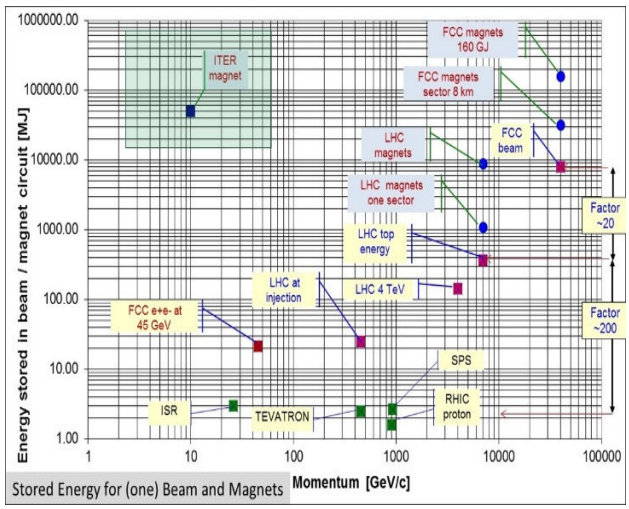


Figure 1: Stored beam energy as a function of particle momentum for a number of accelerators. For comparison, the energy stored in magnet circuits is also shown.

The risk allows to measure the threat of a hazard, by multiplying consequences and probability for a hazard becoming active: **Risk = Probability * Consequences**

Hazards becomes accidents, in general due to a failure or a combination of failures. Related to research instruments, probability and consequences for many different types of failures (e.g. for an accelerator failures leading to beam loss) need to be estimated to evaluate the risk.

Machine protection systems prevent damage to equipment and reduce the risk, either by preventing that a failure occurs, or by mitigating the consequences of a failure. The higher the risk, the more important becomes a robust protection system.

Hazards Related to Magnet Systems

Accelerators and fusion reactors operate with high field superconducting magnets systems. The energy stored in the magnets increased over the years (at TEVATRON, HERA, LHC, ITER, FCC, ….).

Superconducting magnets may quench – and without a protection system such magnets could be damaged. There are many mechanisms that can trigger a quench. A very small amount of energy is sufficient to quench a magnet (down to few mJ). As an example, the loss of a fraction of 10^{-8} of the LHC proton beam in one dipole magnet can lead to a quench. E.g. quenches were induced by the interaction of a dust particle (UFO) with the circulating beam.

Hazards Related to Particle Beams

Regular beam losses during operation lead to activation of equipment and possibly to quenches of superconducting magnets. Radiation induced effects in electronics (Single Event Effects) can perturb the operation of an accelerator.

For accidental beam losses due to failures the hazards need to be understood, e.g. probability and consequences. To understand the consequences, the energy deposition by particles and mechanisms for damage of components need to be estimated.

Examples of Past Accidents

During the first phase of CERN-LHC operation between 2009 and 2013 the magnetic field and therefore the particle momentum was limited to 4 TeV/c. This was the consequence of the 2008 LHC accident that happened during magnet test runs without beam. A magnet interconnect was defective and the circuit opened. An electrical arc provoked a helium pressure wave damaging about 600 m of the LHC and polluting the beam vacuum over more than 2 km. An overpressure from the expansion of liquid helium damaged the structure. A total of 53 magnets had to be repaired [5].

In December 2013 a vacuum leak on a below of LINAC 4 at CERN developed in the MEBT (Medium Energy Beam Transfer) line. The analysis showed that the very low power beam has been hitting the bellow during a special measurement with very small beams in the vertical plane. About 16 % of the beam was lost for about 14 minutes and damaged the bellow. The consequences were minor since LINAC4 is still being commissioned and not used in the chain of LHC injectors. The event demonstrates that beams with very low power (~Joule) can already cause damage.

MACHINE PROTECTION SYSTEM

Figure 2 illustrates an approach to the design of a machine protection system. Hazards are identified and the risk is estimated. Sometimes protection might not be possible, e.g. if a superconducting magnet has a too low ratio between stabilising copper and superconductor, or for devices in an accelerator that can accidentally deflect the beam to the outside of the aperture at an inadequate position.

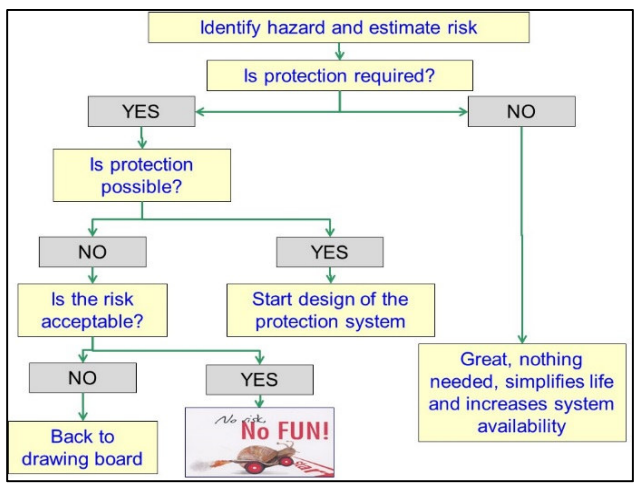


Figure 2: Approach to analyse the need for a protection system starting from the identification of a hazard.

Three Principles for Machine Protection (P3)

Providing equipment for machine protection system is not sufficient to ensure safe operation, other consideration are required:

- Protect the equipment (machine protection systems + interlock systems). The level of protection that is required needs to be defined based on risk.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- Protect the process (high availability protection systems). Machine protection systems will always contribute to downtime. The protection action should be performed ONLY if a hazard becomes active (e.g. something went wrong threatening to damage equipment).
- Provide the evidence (post mortem, logging of data) [6] for different events: 1) a failure is detected and a protection action is performed, 2) a failure in the protection systems leads to a protection action that is not required, 3) a near miss and 4) an accident. In all cases it is essential to understand the event. All relevant system should record their proper data (e.g. with circular "post mortem" buffers in the equipment to record data, and stop and read out the data after the protection action is performed). Slow logging, typically in the order of one Hz, is also very helpful. Synchronisation of different systems is required, to exactly understand the sequence of events. With such data, post operational checks can be performed by the controls system and/or by operators.

Active Protection

Active protection requires the detection of the failure by a sensor. This could be an instrument in an equipme system, or by beam instrumentation detecting when the beam starts to be affected by the failure (for example, increased beam losses or a different beam trajectory). For superconducting magnets, the quench detection syste detects the start of a quench by measuring the resistive voltage across parts of the electrical circuit.

When a failure is detected, operation must be stopped with an actuator. For beam in synchrotrons and storage rings the beam is extracted by a fast kicker magnet and transported to a beam dump block. The block must be designed to accept the beam pulse without being damaged. Injection must be stopped. For linacs beams, the beam is stopped in the low energy part of the accelerator b switching off the source, deflecting the low energy beam by electrostatic plates ("choppers") or by switching off the RFQ for proton linacs. For an accelerator complex with a chain of several accelerators, injection of beam into the next stage of the accelerator complex should be prevented.

For superconducting magnet systems there are severa methods to extract the energy from the circuit with the quenched magnet, e.g. to switch a resistor into the circuit and fire quench heaters [7].

Experience from LHC shows that for most type of failures a careful and fast monitoring of hardware parameters allows stopping beam operation before th beam is affected. Parameters monitored include state and analogue signal. As an example, when a trip of a magnet power converter is detected, the beams are extracted before there is any effect on the beam.

It is not always possible to detect failures at the hardware level. The second method is to detect the initial consequences of a failure with beam instrumentation and to stop the beam before equipment is damaged. This requires reliable beam instrumentation such as beam loss, beam position or beam current monitors.

An electronic interlock system links the different parts of the protection system, the sensors and the actuator. For magnets circuits the interlock system informs the system for energy extraction about a quench. The interlock system might include complex logics that depends on the operational state.

Passive Protection

There are failures (e.g. ultra-fast losses) when active protection is not possible. One example is protection against misfiring of an injection or extraction kicker magnet in an accelerator. A beam absorber or collimator is required to stop the mis-steered beam in order to avoid damage. All possible beam trajectories for such failures must be considered, and the absorbers must be designed to absorb the beam energy without being damaged. Another example is a fast extraction of high-intensity beam from a circular accelerator into a transfer line. When the extraction takes place, the parameters of the transfer line, e.g. the current of the magnets, must be correctly set since for a wrong magnet current the beam would be mis-steered and risk to damage vacuum chamber and other components. An installation of absorbers in critical places can mitigate the consequences.

The machine protection system has also to monitor the parameters before the beam transfer, and only allowing extraction if all parameters are within specified limits.

LHC STRATEGY FOR MACHINE PROTECTION AND INTERLOCKS

In this section we discuss the strategy adopted for LHC machine protection from beam hazards and the related systems:

- Definition of the aperture by collimators.
- Stop beam by beam absorber / collimator for specific failures, e.g. at injection.
- Detect failures at hardware level and stop beam operation for critical failures.
- Detect initial consequences of failures on the beam with beam instrumentation.
- Transmit the signal from instrumentation via a highly reliable interlock system to the extraction kickers and injection system.
- Stop beam operation by extracting the beams into beam dump block.
- Inhibit injection into LHC and extraction from the SPS (the pre-accelerator for LHC) in case of a failure.

Figure 3 illustrates the interlock systems for LHC. The core is the Beam Interlock System that receives beam dump requests from many connected systems. The system is based on FPGAs with a µs reaction time. If a beam dump request arrives, a signal is send to the beam dumping system to request the extraction of the beams. At the same time, a signal is send to the injection system to block injection into LHC as well as extraction of beam from the SPS. A third "post-mortem" signal is provided to the timing

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

system that sends out a request to many systems for providing data that were recorded just before the beam dump, to understand the reasons for the dump (using data from beam loss, beam position, beam current, magnet currents, etc.).

The most complex technical system of the LHC is the superconducting magnet and powering system. The Powering Interlock System (PIC) ensures communication between systems involved in the powering of the LHC superconducting magnets [8]. This includes power converters, magnet protection system, UPS (un-interruptible power supplies), emergency stop of electrical supplies (AUG) and the cryogenic system. As an example, in case a magnet quench is detected by the quench protection system (QPS) the power converter must stop. When a failure is detected that will stop powering of magnets, a beam dump request is send to the Beam Interlock System.

A second systems is managing interlocks from normal conducting magnets and their power supplies (WIC) that ensures protection of the magnets in case of overheating.

Both Powering Interlock System PIC and WIC are based on PLCs that are much slower than the Beam Interlock System, with ms reaction time.

The machine interlock system is strictly separated from interlocks for personnel safety such as the personnel access system, however, an interlock from the access system is send to the Beam Interlock System.

Many other systems provide also beam dump requests in case of failure: vacuum system, RF, devices that can potentially move into the beam pipe, LHC experiments.

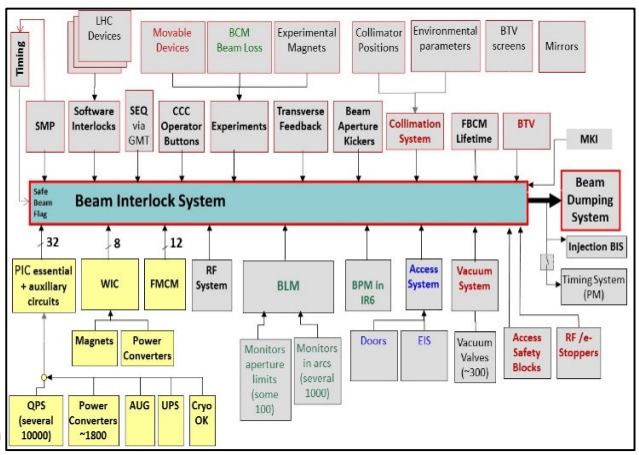


Figure 3: Interlock systems for the CERN-LHC. The core is the Beam Interlock System, many other systems are connected to it that can request a beam dump.

The LHC Software Interlock System ensures redundant protection for many hazards and early detection of failures. It also verifies that the LHC operational parameters remain within well-defined boundaries (e.g. closed orbit deviation within specifications). If a failure is detected, a signal is either send to the Beam Interlock System, or injection is blocked.

In total, there are several 10 thousand interlock signals.

DESIGN OF INTERLOCK SYSTEMS

The most critical parameter for the design of a protection and interlock system is the reaction time. For beam operation in an accelerator there are many failures that require a reaction with a reaction time down to µs. Interlock systems for superconducting magnets require in general a much slower reaction, in the order of milliseconds to seconds.

Fast interlock systems are in general based on hardware (Electronics/Asics), they might include intelligent controllers (FPGAs, DSPs). Such systems can be extremely fast, down to a few ns.

For slower interlock systems, PLCs (Programmable Logic Controllers) are widely used. Using standard PLCs reaction times of one to few ms can be achieved, with safety PLCs the reaction time is in general between several 10 to hundred ms.

At CERN, a Software Interlock System was introduced with great success. Many failures can already be seen seconds before the beam is affected, or a magnet quenches. With the SIS, a reaction time in the order of one second is achieved. During the 6 years of operation no “unsafe” failure of the system was observed.

The second most important parameter for protection and interlock system is the required level of protection. The standard IEC 61508 is a basic functional safety standard for all kinds of industry. The safety integrity level (SIL) provides a target to attain in regards to a system's development that is widely used in industry, mainly for the safety of people. To follow such standard for research instrument turned out to be not very practical, since the procedures to be used by the standard are rather cumbersome. However, for comparing the risk for different hazards and for formulating the requirements for the protection system such approach is very useful. Inspired by the SIL levels, the Protection Integrity Level (PIL) has been introduced, with four levels, from PIL1 (protection for hazards with the lowest risk) to PIL4 (protection for hazards with the highest risks) [9].

There are a number of considerations for selecting a system:

- In a radiation environment (e.g. Single Event Effects) radiation tolerant electronics is required. PLCs are excluded. Whenever possible, an installation in such environment should be avoided!
- An interlock system communicates between several systems. This can be done by current loops, frequency loops, or use of intelligent network (Profibus, Profisafe, Ethernet, ..).
- Time for development (e.g. in-house design of electronics versus buying and programming PLCs, ..) needs to be considered.
- The system should match the lab environment and standards (e.g. hardware such as choice of crates, software, etc.).
- The competence in the lab and the long-term maintainability need to be considered.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- Interlock systems are not major cost drivers, therefore the cost is not a decisive criterion.

TESTING, COMMISSIONING AND OPERATING

Already during the early phase in the design of the protection system, functional testing needs to be considered. Correct commissioning and regular testing of protection systems is vital to ensure reliable operation.

Repeated testing is very time consuming, can be extremely boring and prone to errors, in particular if done by humans. Automatic test procedures and automatic validation of the results via the controls system are very helpful. For LHC, a framework for automatic testing was developed and used for LHC magnet system commissioning, with ~10000 tests performed about once per year [10].

Partial commissioning of an accelerator, in particular for linacs, should be taken into account for the development of the protection system, to avoid frequent reconfiguration. If only the first part of the linac is commissioned, interlocks using downstream equipment should not obstruct commissioning.

For a large system such as LHC several million parameters for the protection systems need to be maintained. Many parameters can only be defined with operational experience. Management of critical parameters and the access to such parameters need to be considered. Regular comparison ensure that parameters in the database and in the hardware are identical. At CERN access to critical parameters with the highest PIL is not possible via the controls system. Parameter with medium PIL can be changed via the control system, but strict rules are defined, e.g. two people must be present to be perform a parameter change. For low PIL, parameters can be changed via the control system.

Several 10k interlock channels are present, all can prevent operation. This can be a nightmare for starting-up a system, in particular if the risk is (close to) zero, e.g. for the commissioning of an accelerator with very low beam intensity. If the option of bypassing of interlocks is considered during the design phase, bypassing by manual procedures on operator's discretion should be avoided.

DESIGN RECOMMENDATIONS AND AVAILABILITY

Considering the experience at CERN and elsewhere some design recommendations are formulated:

- Avoid (unnecessary) complexity for protection systems.
- Failsafe design and detection of internal faults.
- Possibility for remote testing at regular time intervals, for example between two runs.
- Critical equipment should be redundant (possibly diverse redundancy, using different types of equipment).
- Critical processes not by software and operating system.
- No remote changes of most critical parameters.
- Calculate safety / availability / reliability by methods to analyse critical systems and predict failure rate.
- Managing interlocks, always have a clear view of what is interlocked.
- Bypassing of interlocks is common practice (keep track!). For the LHC, bypassing of some interlocks is possible for "setup" beams (low-intensity beams).
- Time stamping for all system with adequate synchronisation is essential.

Availability

If the only objective is maximising safety and too many interlocks are present, this might reduce the overall availability. The challenge it to find a reasonable compromise between safety and availability.

As a technique to improve availability while maintaining safety, majority voting can be considered. An optimum has been found with 2oo3 voting systems that ensure an excellent level of safety, while not stopping operation if one of the three redundant branches indicates a failure, and therefore increasing the availability [11]. A prototype for the powering interlock system for the ITER superconducting magnets has been build according to this principle [12].

MACHINE PROTECTION AND CONTROLS

The controls system has a very important role in the context of machine protection. In many institutes, the responsibility for the hardware and software of the interlock system is within the responsibility of controls. As already discussed in this paper, many control tools can contribute to safe and efficient operation.

- Logging and Post Mortem recording of data, together with accurate and reliable time stamping.
- Framework for managing critical parameters.
- Framework to relax interlock conditions when risks are low ("masking or bypassing of interlocks").
- Framework for automatic testing of machine protection functionalities.
- Framework to respect operational boundaries (sequencer, state machine).
- Clear on-line display of critical parameters to operators (e.g. display of beam losses).
- Feedback systems to keep parameters within predefined limits (e.g. closed orbit).

ACKNOWLEDGMENT

Many colleagues contributed to these considerations on machine protection systems, and it is not possible to acknowledge all of them. However, I like to mention Jorg Wenninger and Markus Zerlauth, who have been very close collaborators for a period of more than 10 years. Without them, neither the LHC machine protection system nor this paper could have been realised.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

REFERENCES

[1] "Joint International Accelerator School on "Beam Loss and Accelerator Protection"," 2014. [Online]. Available: http://uspas.fnal.gov/programs/JAS/JAS14.shtml.

[2] "FCC Week 2015," 2015. [Online]. Available: http://indico.cern.ch/event/340703/timetable/#all.detailed.

[3] A. Nordt, "DEVELOPMENT AND REALISATION OF THE ESS MACHINE PROTECTION CONCEPT," in *15th International Conference on Accelerator & Large Experimental Physics Control Systems*,, Melbourne, Australia, 2015.

[4] W. Pam, "OVERVIEW OF WORLDWIDE ACCELERATORS FOR ADS," in *IPAC2014*, Dresden, Germany, 2014.

[5] J. Wenninger, "Machine Protection and Operation for LHC," Newport Beach, USA CA, 2014.

[6] M. Zerlauth, "The LHC Post Mortem Analysis Framework," in *12th International Conference on Accelerator & Large Experimental Physics Control Systems*, Kobe, Japan, 2009.

[7] H. Pfeffer, Protection of Hardware: Powering Systems (PC, NC and SC Magnets), Newport Beach, USA CA: JAS / CAS, 2014.

[8] M. Zerlauth, "A Retrospective View to the Magnet Interlock Systems at CERN," in *5th International Particle Accelerator Conference*, Dresden, Germany, 2014.

[9] M. Kwiatkowski, *Methods for the application of programmable logic devices in electronic protection,* Geneva, Switzerland: CERN-THESIS-2014-048, 214.

[10] A. Gorzawski, "The AccTesting Framework: An Extensible Framework for Accelerator Commissioning and Systematic Testing," in *14th International Conference on Accelerator & Large Experimental Physics Control Systems*, San Francisco, CA, USA, 2013.

[11] S. Wagner, "Architecture for Interlock Systems : Reliability Analysis with Regard to Safety and Availability," in *13th International Conference on Accelerator and Large Experimental Physics Control Systems*, Grenoble, France, 2011.

[12] M. Zaera-Sanz, "Design, Development and Implementation of a Dependable Interlocking Prototype for the ITER Superconducting Magnet Powering System," in *14th International Conference on Accelerator & Large Experimental Physics Control Systems*, San Francisco, CA, USA, 2013.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DESIGN, IMPLEMENTATION AND SETUP OF THE FAST PROTECTION SYSTEM FOR CSNS *

D. P. Jin#, Y. L. Zhang, P. Zhu, IHEP, Beijing, China; DNSC, Dongguan, China

Abstract

The China Spallation Neutron Source (CSNS) [1, 2] is being constructed in Dongguan, Guangdong Province, China. The first proton pulse will hit on the target in March, 2018 as planned. CSNS is a large neutron beam research facility with a proton beam power of 100 kW. Three neutron instruments are being constructed for the first stage, with another 17 to be built later. The Fast Protection System (FPS) is one of the key systems to prevent the accelerator components along the proton beams from being damaged. Design, implementation, setup and tests of the FPS are introduced in this paper.

INTRODUCTION

As the other high power accelerators in the world, components along the proton beam will possibly be damaged by a mis-steering beam, especially for the DTL (Drift Tube LINAC) cells. Another issue is the post irradiation due to the beam loss, which should be controlled strictly below 1 W/m for hand maintenance [3, 4]. A Fast Protection System (FPS) with a response time of around 10 μs is strongly needed. A self-designed FPS has been developed and preliminary tests has been brought out. Till now, most of the signals from the power supplies and beam loss monitors along the linear accelerator are collected by the Fast Protection System.

DESIGN AND IMPLEMENTATION OF THE FPS FOR CSNS

FPGA is chosen as the base of the design since its flexibility and fast response time. High speed serial transmission technique is used to reduce the number of connections tremendously. A 6U standard VME crate is used to ease the maintenance and upgrade, and to improve the reliability and scalability. Figure 1 shows the distribution of the FPS.

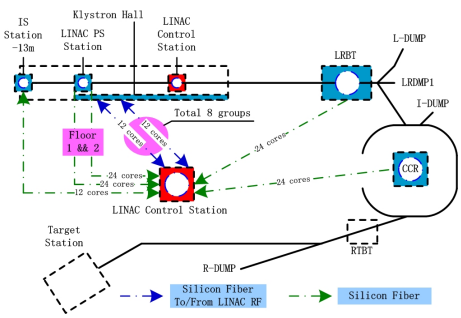


Figure 1: Distribution of the FPS.

Figure 2 shows the structure of the FPS.

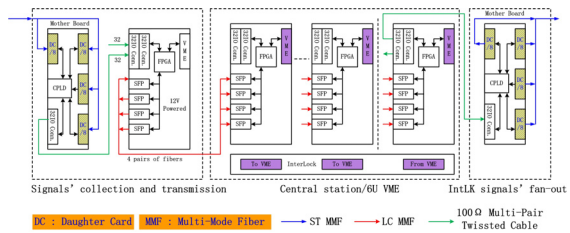


Figure 2: Structure of the FPS.

Signals collection are done locally near the interlock sources, and then transmitted to the central station via high speed serial optical links. Total 16 signals are carried with one pair of fibers. The high speed serial link are implemented with the Rocket I/Os of the XC6SLX100T-3FGG900 from Xilinx and optical transceivers. A dedicated auto-sync logic has been developed for the serial link to work when both fibers of the same link are plugged into the optical transceivers under normal conditions.

Figure 3 shows the picture of the main board. Either VME +5V or external +12V can be used to power the board.

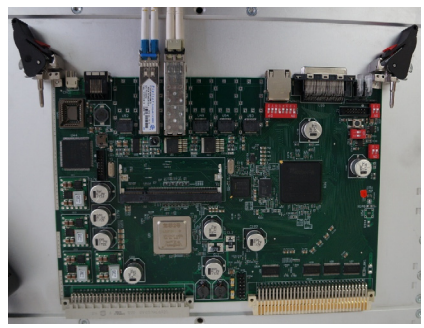


Figure 3: Picture of the main board of FPS.

To improve the overall reliability, a real-time heart-beat function is used for each input signal. That is, heart-beat signals are generated at the collection part, and then checked in the main boards at the central station. A width of 100ns is defined for the heart-beat signals in our case, see Fig. 4.

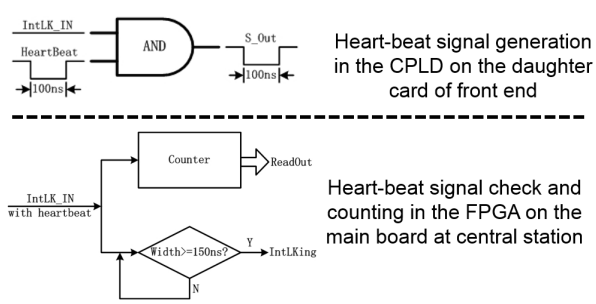


Figure 4: Illustration of the heart-beat function.

*Work supported by National Development and Reform Commission of China, Chinese Academy of Sciences, and Guangdong Province.
#jindp@ihep.ac.cn

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The Bit error rate for the high speed serial link has been tested with no errors for a period of 27 hours for two links. Figure 5 shows the scheme for bit error rate test.

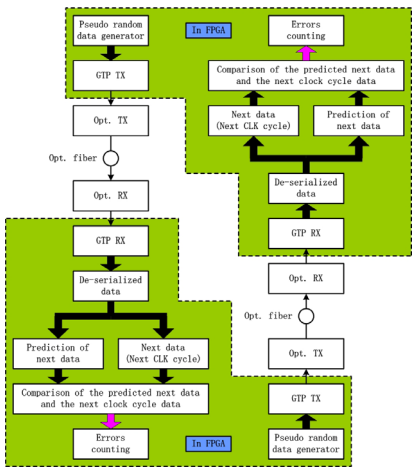


Figure 5: Scheme of the bit error rate test.

Figure 6 shows the field photos of the central station and one of the local stations of the FPS.

Figure 6: Photos of the central station and one of the local stations of the FPS.

FIELD TESTS OF THE FPS

Interlocking functions and heart-beat functions were tested first with all functions ok. Time consumption has been measured thoroughly since the critical requirement. Figure 7 shows the critical paths for the FPS. Delay of cables and fibers contribute mainly.

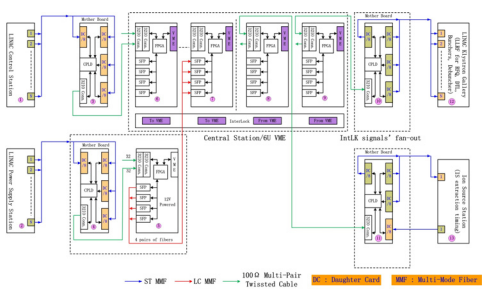


Figure 7: Illustration of critical paths for the FPS.

For the high speed serial link with no fibers, the delay is around 230ns. All delays of the parts on the route from interlock inputs to outputs are tested. Total delay from the inputs to stop the RFQ power is around 1.8 μs, and that to stop the extraction timing of the ion source is around 2.4 μs. For power supplies, response time of themselves is between 2.8 μs to 3.4 μs. So, the total response time is no more than 5.2 μs to stop the RFQ power, and no more than 5.8 μs to stop the extraction timing of the ion source. For the beam loss monitors, the response time of themselves is defined as 10 μs with a compromise among noises, cosmic rays and real loss signals, so, total response time is no more than 11.8 μs to stop the RFQ power, and no more than 12.4 μs to stop the extraction timing of the ion source.

Further tests will be brought out with the equipment and proton beams later.

SUMMARY

A Fast Protection System for CSNS has been developed and setup. FPGA and high speed serial transmission technique is used for a fast and compact design. Heart-beat function is implemented to improve the overall reliability. Function tests and time consumption tests have been done, results of which show that the system can fulfill the requirement well. Further tests will be done according to the construction procedure.

REFERENCES

[1] IHEP Report IHEP-CSNS-Report/2004-01E (2004).
[2] J. Wei et al, “China Spallation Neutron Source Design”, APAC 2007, Indore, India.
[3] C. Sibley, “Machine Protection Strategies for High Power Accelerators”, PAC03.
[4] M. Ross, “Single Pulse Damage In Copper”, SLAC, Stanford, CA94309, USA.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DEVELOPMENT AND REALISATION OF THE ESS MACHINE PROTECTION CONCEPT

A. Nordt, R. Andersson, T. Korhonen, A. Monera Martinez, M. Zaera-Sanz, European Spallation Source ERIC, Lund, Sweden
A. Apollonio, R. Schmidt, CERN, Geneva, Switzerland
C. Hilbes, ZHAW, Winterthur, Switzerland

Abstract

The European Spallation Source ERIC (ESS) is facing extremely high beam availability requirements and is largely relying on custom-made, specialised, and expensive equipment for its operation. The average proton beam power of 5MW per pulse will be unprecedented and its uncontrolled release can lead to serious damage of the delicate equipment, causing long shutdown periods, inducing high financial losses and, as a main point, interfering drastically with international scientific research programs relying on ESS operation. Implementing a fit-for-purpose machine protection concept is one of the key challenges in order to mitigate these risks. The development and realisation of the measures needed to implement such concept to the correct level in case of a complex facility like the ESS, requires a systematic approach, and will be discussed in this paper.

ESS MACHINE PROTECTION

The European Spallation Source ERIC (ESS), currently under construction, consists of a 600m long high power proton Linac, accelerating proton pulses of 2.86ms length to the energy of 2GeV with a repetition rate of 14Hz. The proton pulses (5MW) then interact with a rotating tungsten target wheel. Neutrons are created due to the spallation process and further guided through 22 different neutron beam lines towards the experimental stations. It is expected to have the first protons on target mid 2019 [1].

As a user facility for neutron science, overall availability of the ESS needs to be defined from a user point of view. Hence, it should be characterized by the average neutron production during a certain time period. ESS availability is interpreted as the average proportion of beam production time during scheduled operation time. In general, the availability characteristics of a system are determined by its reliability, maintainability and inspect-ability. High operational availability is achieved by increasing the mean time between maintenance while avoiding large mean downtimes. A detailed discussion is presented in [2, 3].

In the context of ESS Machine Protection (MP), the term “machine” or Equipment Under Control (EUC) encompasses all elements in the Accelerator, Target Station and Neutron Science system segments - all being necessary for neutron beam production and its further use by the neutron science experiments.

Annika.Nordt@esss.se

ESS MP Design Approach

MP is concerned with operational goals of the ESS, that means, enabling neutron science and investment protection. It is not concerned with safety aspects that are regulated by legal authorities, such as personnel safety or public safety. Nevertheless, because of the high-risk potential associated with damage to the machine, elements related to MP will be implemented in accordance with functional safety standards [4, 5]. To keep the distinction between safety and protection requirements as transparent as possible, adequate definitions for MP have been introduced. The IEC 61511 is defining a Safety Integrity Level (SIL) [5], whilst we refer to the Protection Integrity Level (PIL) for MP instead [6, 7]. Protection Integrity describes the average probability of a protection system satisfactorily performing the required protection functions under all the stated conditions within a stated period of time.

ESS MP Goals and Top Level Requirements

The EUC is exposed to potential damage sources related to proton and neutron beam properties, related radiation, electrical power, vacuum, cooling, RF, etc. The severity of damage is defined with respect to neutron beam quality losses, quality loss duration and resource costs for the recovery of operational capabilities. The goals for MP are defined as follows [6]: MP shall, in that order, prevent and mitigate damage to the machine, be it beam-induced or from any other source, in any operating condition and lifecycle phase, in accordance with beam- and facility-related availability requirements. In addition, MP shall protect the machine from unnecessary beam-induced activation having a potential to cause long-term damage or increase maintenance times.

These two goals can be achieved if MP fulfils the top-level functional and operational requirements being described in the following. MP shall detect all relevant off-nominal states that can lead to damage or unnecessary beam-induced activation and take all the necessary actions for its prevention and mitigation. Protection functions shall be implemented with timing and protection integrity levels in accordance with damage risk reduction requirements. The protection functions shall be implemented such that the probability of spurious trips is reduced. All information about detected off-nominal states, performed prevention and mitigation actions shall be recorded, allowing for an a-posteriori reconstruction and analysis. Operation during all foreseen lifecycle phases of the machine, for example assembly and

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

installation, commissioning, tuning, operation, fault-finding, maintenance, and dismantling shall be supported. Also all operating modes of the machine, for example proton beam up to intermediary targets, proton beam with reduced beam power or alternative beam envelopes, and proton beam with alternative duty cycles shall be supported. It is important that MP supports operation in case of degraded equipment under control and in case of degraded protection functions.

FUNCTIONAL ARCHITECTURE CONCEPT

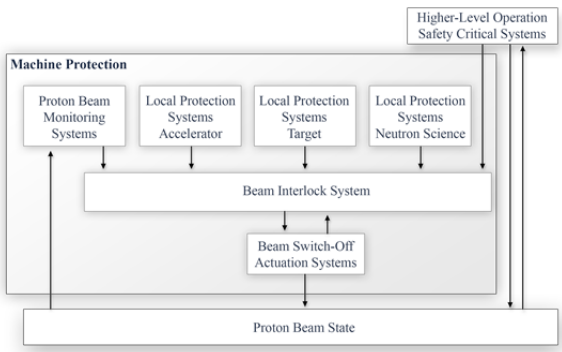


Figure 1: Functional architecture concept for ESS MP. Details are described in this section [6].

Off-Nominal States

The states that have to be monitored for off-nominal state detection can be grouped into four major parts.

Monitoring of the local state of the accelerator segment EUC. Accelerator EUC includes e.g. magnets, magnet power-supplies, RF generators, cavities, beam-choppers, proton source, vacuum valves, vacuum, and cryogenics. Respective local state variables would be vacuum valve positions, magnet coil temperatures, magnet power-supply output currents, etc.

Monitoring of the local state of the target segment EUC. Target EUC includes the target wheel, moderator, reflector, cooling system, etc. Respective local state variables would be target wheel rotation velocity and wheel temperature, etc.

Monitoring of the local state of the neutron science segment EUC. Neutron science EUC includes e.g. neutron choppers, light shutters, neutron guides, and experiment specific equipment. Respective local state variables would be neutron chopper velocity, and light shutter positions.

Monitoring of the proton beam state. State variables include beam current, beam shape, beam position, beam losses.

In principle, deviations of any of the mentioned state variables could lead to damage or to beam-induced activation.

Local Protection Systems (LPS)

These systems are in charge of monitoring specific EUC within a specific segment and detecting off-nominal local states that can lead to damage. They need to take the necessary actions to prevent damage to the monitored EUC in such a case. Depending on the EUC, off-nominal states might have an influence on the beam and cause beam-induced damage or beam losses leading to activation further downstream. For this reason, LPS are required to take any necessary action to switch off the beam. Starting beam operation while knowing that something is wrong should be avoided. This would unnecessarily increase the demand rate and protection integrity requirements for the Proton Beam Monitoring Systems. Therefore the LPS shall additionally confirm that everything in their segment is ready for beam production, if relevant (i.e. provide a beam permit signal).

Proton Beam Monitoring Systems (PBMS)

As the beam itself is a potential source of damage and the source for beam-induced activation, MP has to make sure that the beam parameters are as they should be. Detection of off-nominal beam states is allocated to the PBMS. It should be noted that proton beam monitoring could only help mitigate effects of already produced beam.

Beam Interlock System and Beam Switch-Off Actuation Systems

The Beam Interlock System (BIS) collects the signals from all LPS and PBMS and combines them into one global beam permit state. The BIS is controlling a set of beam switch-off actuation systems that will finally switch off beam. The actuation systems can be divided into two classes: those that are able to prevent new beam production and those that are able to mitigate the effects of beam that has been already produced and is injected into the Linac.

By switching off the magnetron of the proton source, new beam production can be prevented (it takes ~100µs until no further plasma is being created). The creation of trigger signals for new pulse generation can be stopped at the level of the timing system.

By activating the LEBT chopper to deflect beam onto the LEBT absorber (rise time: 300ns), the effects of already produced beam can be mitigated. It should be noted that the LEBT absorber allows only a few nominal pulses to be dumped before it gets damaged itself. Also the MEBT chopper can be activated to deflect beam onto the MEBT absorber (rise time: 10ns). However, the MEBT absorber allows only for a partial beam pulse dump before it is damaged (~200µs). Furthermore it is possible to control the RF supply to the RFQ in order to stop beam from getting further accelerated (~20µs).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CHALLENGING ESS MACHINE PROTECTION REQUIREMENTS

During a preliminary risk and hazard analysis for the accelerator EUC, a total of 166 different protection functions (PF) have been identified [8]. The most stringent requirements resulting from this analysis will be described in more detail in the following.

The proton beam shall be stopped within 4 to 5µs when detecting non-nominal beam states in the low-energy part of the Linac. This short reaction time is resulting from the time needed to melt copper or steel at (low) beam energies of 1 to 3.6MeV which is around 10 to 20µs in case of perpendicular impact of a beam with 2mm size (see Tab. 1). In order to stop beam before melting starts, the overall reaction time has been set to 4 to 5µs for the low energy part of Linac (i.e. first 50m of the Linac). The reaction time shall include the detection of beam losses, the processing of this information and the stop of beam operation. Several systems are needed to assure this requirement is being fulfilled. The only Beam Instrumentation Systems, which can detect critical beam losses in the low energy part of the Linac, are the Beam Current Monitors (BCMs) and Beam Position Monitors (BPMs). Beam Loss Monitors (BLMs) are sensitive only for energies above 80MeV. The BCMs are designed such that a response time of 1µs can be achieved. The BCM response time includes the detection of beam losses, the comparison to a pre-defined beam loss threshold and the propagation of a digital signal towards the BIS, indicating that beam operation is permitted or that beam operation shall be stopped upon the detection of beam losses, which exceed the pre-defined thresholds. The BIS is then triggering the different beam switch-off actuation systems simultaneously within ~2 to 3µs. However, as it can be seen from the description above, only the LEBT chopper is able to stop the proton beam in less than 1µs and is able to absorb at least one nominal beam pulse of 2.86ms length without being damaged. Tab. 1 shows the different reaction times needed for different proton beam energies.

Table 1: Times needed to stop beam operation based on the time needed to melt steel or copper upon perpendicular impact of proton beam (with a size of 2mm).

Beam Energy	Melting Time	Beam Stop Time
1 - 3.6 MeV	10 - 20 µs	4 - 5 µs
3.6 - 90 MeV	20 - 200 µs	5 - 20 µs
90 - 216 MeV	200 - 400 µs	20 - 40 µs
> 216 MeV	> 400 µs	> 40 µs

A PIL2 for the protection functions with most severe impact on the overall ESS operational availability has been defined [8]. Since a protection function includes sensor systems, such as BCMs, BLMs, different LPS, etc., the BIS and the beam switch-off actuation systems, an allocation of the tolerable failure limits corresponding to a PIL has been proposed:

- 35% of the total PIL for BIS input systems,
- 15% of the total PIL for the BIS,
- 50% of the total PIL for the beam switch-off actuation systems.

The PIL2 overall requirement is then resulting in a tolerable failure rate of $1.5\text{x}10^{-7}$ – $1.5\text{x}10^{-8}$/h for the BIS which actually corresponds to the tolerable failure rate limit related to a PIL3 (see Fig. 2 and Tab. 2). The allocation mentioned above can be understood such, that e.g. the likelihood of the BIS to fail shall be only 15% out of the total PIL2, indicating that the BIS shall fulfil the most challenging requirement in terms of tolerable failure rates.

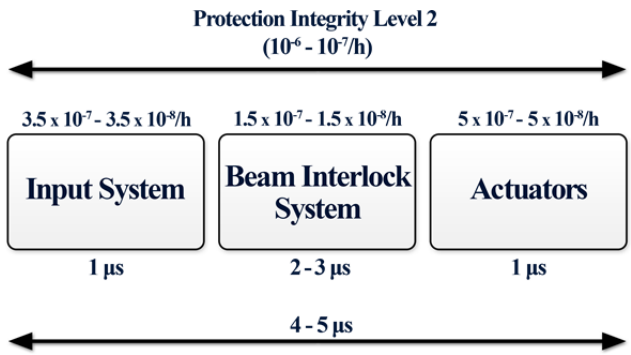


Figure 2: Shown is the PIL2 requirement and resulting tolerable failure rates allocated to different systems. The fastest reaction time and its allocation to the different systems are shown as well.

A first Failure Mode, Effects and Diagnostics Analysis (FMEDA) for the BIS prototype indicates that the required tolerable failure rates can be achieved [9]. It should be noted that only random hardware failures are analysed, whereas configuration errors, software, firmware and other failures are not considered in an FMEDA and need to be evaluated using other methods. A more detailed description of the BIS design can be found in [10].

Table 2: Different Rates of Tolerable Failures Per Hour [4, 5]

PIL (Protection Integrity Level)	Tolerable Failure Rates/h
1	10^{-5} – 10^{-6}
2	10^{-6} – 10^{-7}
3	10^{-7} – 10^{-8}
4	10^{-8} – 10^{-9}

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

THE MACHINE PROTECTION MANDATE AT THE ESS

ESS MP addresses stakeholder concerns and functions that cut across different ESS divisions and systems. Hence, a cross-divisional organizational unit is established for the overall coordination and decision-making on MP concerns, the ESS MP Committee (MPC). The MPC coordinates MP related activities with the relevant ESS divisions, working groups, in-kind contributors and experts of the ESS equipment and operation teams. One of the major tasks of this committee is to coordinate the identification, assessment and documentation of relevant risks, hazards, failure scenarios of the EUC. The committee is furthermore in charge of coordinating the coherent development (including design, integration, commissioning) of the EUC and its future changes or upgrades in regard to MP, but also the coordination of the ESS operation concerning MP.

The MPC formally approves overall MP decisions, including approval of overall MP requirements and protection functions, the overall technical decisions, the delegation of tasks (system development, commissioning, operation, etc.) to the divisions, and the overall development approaches for EUC local protection systems. In addition, the MPC defines boundary conditions for operation (proton beam power, repetition rate, etc.) and authorities/procedures for short-term interventions (e.g. overnight relaxation of operational boundaries).

The MPC is composed of representatives from all ESS divisions who are stakeholders in MP, with decision-making authority for their division. Currently it includes representatives of the Accelerator division, Target division, Integrated Control Systems division, Neutron Scattering Science division, and Operations division.

The MPC receives its mandate from the overall ESS management. Complementary to the MPC and its mainly formally approving character is the MP Panel (MPP), a discussion forum that meets regularly in order to gather relevant MP information and communicate MP issues into the organization [6, 11].

CONCLUSION

The European Spallation Source ERIC (ESS) is facing very challenging requirements for beam availability and systems reliability. At the same time the beam power is unprecedented (5MW) introducing a high risk potential for damaging delicate equipment. If the beam is for example not within the right beam parameter space, melting of copper or stainless steel can occur within a few microseconds only, potentially leading to long downtimes and costly repair actions, impacting significantly on the research mission of ESS. It is thus important to develop a fit-for-purpose machine protection concept, ensuring the high availability goals but also providing proper protection integrity. Such a concept cannot be developed and implemented by building sophisticated and dedicated protection systems only. It is equally vital to be able to raise awareness of the potential risks and to have a decision making body for machine protection relevant issues in place, being supported by the upper management.

The impact of certain decisions, when designing (local or single) systems, on the global performance level of a complex facility like ESS is often underestimated and not addressed in time, increasing the damage potential even further. At ESS, some effort has been taken in order to face these challenges and reduce the risks hopefully as good as possible.

REFERENCES

[1] J. Yeck, “Neutron facility: European Spallation Source is on track”, *Nature*, vol. 519, no. 7543, p. 291, Mar. 2015.
[2] E. Bargalló et al., “ESS Availability and Reliability Approach”, MOPTY045, IPAC’15, Richmond, USA (2015).
[3] E. Bargalló et al., “ESS reliability and availability requirements”, ESS Report No 0008886, 2015.
[4] IEC61508 Edition 2.0
[5] IEC61511 Edition 1.0
[6] A. Nordt et al., “ESS Machine Protection Concept”, ESS Report No 0035197 and No 0035388, 2015.
[7] Kwiatkowski, Maciej, “Methods for the Application of Programmable Logic Devices in Electronic Protection Systems for High Energy Particle Accelerators”, PhD thesis, Warsaw University of Technology, Warsaw, Poland, CERN-THESIS-2013-216 (2013).
[8] A. Nordt et al., “Preliminary Risk and Hazard Analysis of the ESS Machine Protection System with Emphasis on Production and Property Losses in the ESS-LINAC ”, ESS Report No 0003796, 2013.
[9] R. Andersson et al., “A modified functional safety method for predicting false beam trips and blind failures in the design of the ESS Beam Interlock System”, *these proceedings*, MOPGF126, ICALEPCS 2015, Melbourne, Australia (2015).
[10] A. Monera Martinez et al., “Overview and Design Status of the Fast Beam Interlock System at ESS”, *these proceedings*, MOPGF138, ICALEPCS 2015, Melbourne, Australia (2015).
[11] A. Nordt et al., “Machine Protection Strategy for the ESS”, MOPTY048, IPAC’15, Richmond, USA (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REUSABLE PATIENT SAFETY SYSTEM FRAMEWORK FOR THE PROTON THERAPY CENTRE AT PSI

P. Fernandez Carmona, M. Eichin, M. Grossmann, E. Johansen, A. Mayor, H.A. Regele
PSI, Villigen, Switzerland

Abstract

A new gantry for cancer treatment is being installed at the Proton Therapy Centre in the Paul Scherrer Institut (PSI), where already two gantries and a fix line operate. A protection system is required to ensure the safety of patients, requiring stricter redundancy, verification and quality assurance measures than other accelerators. It supervises the Therapy System, sensors, monitors and operator interface and can actuate magnets and beam blockers. We built a reusable framework to increase the maintainability of the system using the commercial IFC1210 VME controller, developed for other PSI facilities. It features a FPGA implementing all the safety logic and two processors, one dedicated to debugging and the other to integrating in the facility's EPICS environment. The framework permitted us to reduce the design and test time by an estimated 40% thanks to a modular approach. It will also allow a future renovation of other areas with minimum effort. Additionally it provides built-in diagnostics such as time measurement statistics, interlock analysis and internal visibility. The automation of several tasks reduces the burden of QA in an environment with tight time constraints.

INTRODUCTION

The Paul Scherrer Institut (PSI) was a pioneer in the field of proton therapy for cancer treatment by being the first centre to implement spot scanning for dose delivery back in 1996. Nowadays there exist many centres using such technology and several vendors offering commercial products. In order to increase the number of patients being treated it was the decided to buy a scanning gantry from Varian Medical Systems [1], while keeping research and development in the existing in-house engineered areas. The current facilities of the CPT consist of a fixed beam line for eye cancer treatment, operating clinically since 2010, and Gantries 1 and 2 operating since 1996 and 2013 respectively [2] - [3]. The beam is provided by a dedicated 250 MeV cyclotron from the company Varian Medical Systems.

There are several systems required to allow for a safe and accurate delivery of the prescribed dose to the patient. The most relevant ones are the Patient Safety System (PaSS), to prevent accidents and the Therapy Control System (TCS), to deliver dose and to verify the correctness of the delivery. Other systems working independently but interconnected are the Beam Tuning Verification System (BTVS), the Machine Control System (MCS), the Run Permit System (RPS) and the Main Patient Safety Switch and Controller (MPSSC). Finally there is a number of Dose and Beam Position Monitors and a set of final elements such as a kicker magnet and beam blockers.

In order to integrate the commercial gantry in the existing facility it was necessary to develop two adapters: The TCS adapter interfaces the vendor specific control system commands (such as setting energy and beam current values) to the appropriate facility resources. The PaSS adapter autonomously takes care of preventing accidents, and also actuates some final elements on request of the gantry. Figure 1 shows an overview of all the systems to which the PaSS is connected.

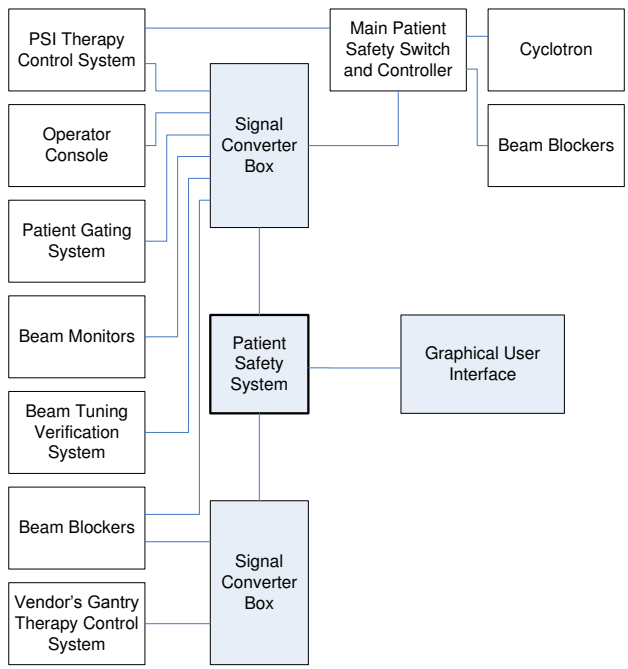


Figure 1: System overview of the integration of the new gantry's Patient Safety System.

This document describes in detail the new Patient Safety System developed for the installation of the new gantry. Also the experience of conceiving and designing it as a reusable framework, comparing it to previous designs is presented. Final tests for regulatory approval and clinical commissioning will take place during 2016 and patient treatment will start later in the year.

PATIENT SAFETY SYSTEM CONCEPT

The main goal of the patient irradiation for CPT is that each dose spot is delivered at the correct position and with the correct dose quantity. According to the International Commission on Radiation Units and Measurements (ICRU) [4], the aim in intensity modulated radiation therapy is to achieve an accuracy of 5% of the

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

total treatment dose. CPT adheres to the following safety goals [5]:

1. "No radiation accident", considering a worst case local 5% dose excess.
2. "No error in the delivered dose", avoiding dose distribution errors ≥ 2% of the planned field dose.
3. "No error in dose position", aiming at ± 1mm in lateral and depth direction
4. "Delivered dose and dose position must be know at all times" so irradiation can be interrupted and resumed safely.

The systems verifying that the previously defined goals are achieved are the Patient Safety System and the Therapy Control System, working independently. The PaSS collects information from several sensors and can actuate on certain final elements. Internally it has a hierarchical structure with ready signals and three levels of interlocks, from low to high severity as detailed in Table 1. Each level has a success supervision system and in case it fails, it escalates to the next level. All the logic inside PaSS is hard wired either in a Field Programmable Gate Array (FPGA) or with relays. This is required to maximize stability, predictability and minimize response time. Physically the interconnection lines are based on redundant three wire logic cables. All the information is sent together with its inverse using current loops that allow for detection of short or open circuits.

Table 1: Interlock system levels

Severity	Measures to prevent beam
ALOK	Close local beam blocker and activate deflector magnet
ATOT	Close main blockers, stop the proton acceleration in the cyclotron, plus all final elements actions of ALOK
ETOT	Switch off the cyclotron's acceleration system, the ion source and all final elements actions of ALOK and ATOT

There are two modes of operation, one for therapy and one for experiments and development. The operation mode has to be consistently selected by a physical key at the operator console and by the TCS. Patients can only be treated in Therapy mode and in this mode the PaSS cannot be configured or altered. In Experiment mode it is possible to "bridge" certain interlock values or to overwrite some configuration parameters.

SYSTEM ARCHITECTURE

The main two constraints at the beginning of the project were restricted manpower and limited specifications. For that reason, a modular architecture was chosen, reusing as many elements as possible.

Hardware

The platform used was IFC1210, a commercial Versa Module Europa (VME) Input Output Controller (IOC) from IOxOS Technologies [6]. It features a user programmable Virtex 6 FPGA and two PowerPC Central Unit Processors (CPU), both running SMP Linux. An EPICS kernel driver running on Linux implements the Ethernet-based Channel Access protocol and enables access to the registers inside the FPGA fabric. There are also two FPGA Mezzanine Card (FMC) bays, which were populated with Small Form-factor Pluggable (SFP) optical transceivers. The safety logic was implemented in the user FPGA and is totally autonomous after boot.

There were 98 interlock signals to be distributed to and from other systems in the facility. The connectivity was achieved with the Signal Converter Boxes, which are basically multiplexers. They are outsourced custom designed electronics, highly configurable with an Artix-7 FPGA, several SFP optical transceivers and ten generic plugin ports. The plugin ports are SMC mezzanine connectors to place application specific cards. In this way the platform has been defined to be flexible if the number or type of signals interfacing to the new gantry changed during the design phase. Already existing three wire logic plugins were used to interface with CPT systems, and three new types of plugins had to be developed according to specifications from Varian.

Firmware

IFC1210 provides a powerful firmware infrastructure with a Network on Chip (NoC) to which resources are connected. There are central resources such as FMC support or memory, and also user defined blocks. A user block was coded in VHDL language.

In order to increase reusability the design was divided into a platform independent PaSS Framework, and an application specific IFC1210 code. The framework consists of a package and generic building block definitions, such as timers, interlock trackers or input debounce elements.

The application specific code includes the safety logic, which is a Mealy state machine defining the configuration of the final elements based on present and past inputs, in addition to the specific memory interface, interconnection logic and instantiation of user configurable debug and visualization blocks.

Software

Many PaSS logic status and configuration variables are mapped via the on-board connectivity mesh in the IFC1210 to the EPICS driver and then exported through Ethernet to a local area network (LAN). A Graphical User Interface (GUI) was developed to set and get all 7000 published EPICS registers. It is organised in tabs; an overview is shown in Figure 2.

With the user interface one can visualise hardware and logical states of input, output and internal signals, display detailed information or bridge logical states of allowed signals. Also some statistic or debug information is displayed, like a chronological list of events, counters and timing statistics.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

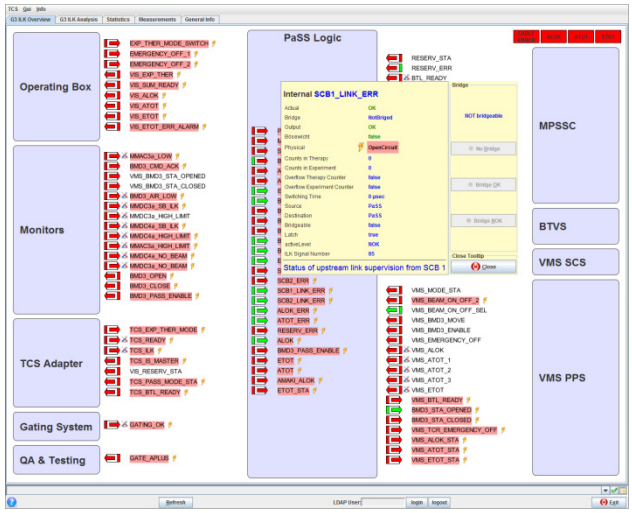


Figure 2: Interlock overview tab of PaSS GUI.

The GUI was programmed in Java using Apache Commons Logging, JavaX Mail, JCA and CosyLab CAJ libraries. It supports EPICS / ChannelAccess communication (directly with LPaSS IOC or through a ChannelAccess gateway), LDAP authentication (to authorize and log user actions), Mailing of important (user or program-triggered) actions and logging to files. Also the GUI is generic and gets all its application specific configuration parameters from an xml formatted file.

Automatic Generation of Configuration Files

A tool was created to generate a number of files based on a spreadsheet containing specific data of the project. It contains elements such as name and number of interlocks, bit width of the memory bus or desired extra visualization modules. It was programmed in Visual Basic and generates the following files: VHDL memory map block, interlock definition VHDL package, EPICS template with all registers and its location in memory, an xml file to configure the GUI and a memory definition header file for developing debug applications for the processor with Linux. In this way not only the system can be reconfigured fast after a modification, but it is highly portable to other treatment areas.

BUILT-IN FUNCTIONALITY

The powerful IFC1210 platform allowed placing extra functionality together with the basic safety logic in the design. This can be divided into the following groups.

General Infrastructure

Each input signal is automatically connected to a conditioning stage that takes care of debouncing in case of unstable sources like relays and decoding of the three wire logic to detect short circuits or broken cables. There is also a latch and bridge stages that modify the effective value of the signals according to the configuration package file. The physical, logic and effective values are all published via EPICS and available in the GUI. During experiment mode only the operator will be able to bridge certain signals values. The switching time between valid states of the physical signals is measured, compared to a timeout threshold and published via EPICS. This data can detect deterioration of the opto-coupler devices used in the three wire logic current loops and damaged pieces can be replaced before actually failing.

Interlock Analysis

An absolute timer with a 1 µs resolution is running permanently on the IOC and when an interlock occurs a timestamp is generated. All the interlocks triggered after the event and last few before are logged. This information can be consulted in a table at the GUI which contains the exact time, name, source and destination of each interlock triggered. This table is cleared together with the interlock status after the cause has been solved and the TCS issues a clear command.

Statistics

For each defined input, output or internal line the PaSS Framework instantiates automatically a counter of events during therapy mode and another one during experiment mode. The GUI has a tab where this information, together with settling time of the inputs and minimum, maximum and average time measurements of reaction times of logic elements are displayed.

Measurements

Various elements of the patient safety logic require the measurement of a time or the supervision of an event before a given timeout. There are several of such blocks which can be used for Quality Assurance (QA) purposes, configurable from the GUI, as shown in Figure 3

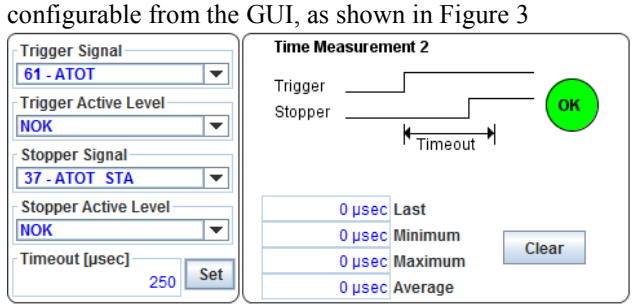


Figure 3: Configurable time measurement.

When the system is in experiment mode it is also possible to overwrite the timeout values of time measurements inside the safety logic itself. This can be used for example to verify during QA that an interlock is triggered if a certain element response is too slow. For the existing therapy areas this is time consuming, as it requires physically accessing the element and disconnecting it for the test. Now the test only requires configuring a few settings in PaSS using the GUI.

There is also a configurable number of interlock event counters which can be set to track a certain signal. This can be useful either for QA or to diagnose a problem when some error is occurring frequently.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

VERIFICATION AND VALIDATION

The verification process of the PaSS for the new gantry has the following steps: risk analysis, design specification, implementation and test specification, each performed by different people.

All the verification and validation steps are based on previous experience with existing areas, following the procedure to obtain authorization for treating patients by the regulatory bodies for Gantry 2 [7]. As it was the case in previous systems, a test stand is being built. This is a piece of hardware that interfaces each interlock line, generating stimuli for the inputs and checking the output for correctness. Every time that a modification is introduced in a PaSS, the system must go through the extensive automated test process.

Finally, after installation the whole PaSS system including final elements undertakes an extensive QA program. These tests are typically measurements of reaction times of final elements and the correct response of the system to several injected errors. Thanks to the built-in measurement capabilities of the PaSS, many such tests can be automated without the need of probing the elements with an oscilloscope.

RESULTS

The introduction of the new PaSS brought several advantages. There are qualitative improvements like extra functionality and more accurate debug tools. There are also quantitative gains both in development time, compared to similar past projects and for easing facility operation.

From the point of view of QA there are time savings to be expected. The yearly test for PaSS and final elements takes approximately ten days for the three existing treatment areas. By automating the measurement of the reaction time of the final elements the QA time is expected to be reduced by three days. Also the failure of switching elements can be predicted and a replacement planned if a performance drift is observed.

Furthermore the availability of an interlock analyser window in the GUI with detailed and deterministic information of past events can result in a faster reaction time to problems by trained personnel. In older implementations, the interlock analysis was performed with an inverted tree of likely events but could not determine an original cause with absolute certainty. The new platform allows tracking individual occurrences of events with a resolution of 1µs. In addition, physicists are provided real time statistics of the frequency of each interlock, helping to detect defects. With that information they are able to make more analysis and better identify trends.

Finally, a careful project planning and work tracking allowed us to have a log of the effort spent in the development. This could be helpful as a work estimate for similar projects and also allows comparing the development time with previous PaSS developments. In Table 2 the number of full time equivalent days of work in the new gantry safety system is shown. In row ”SCB HW” there are the days of work reported by the subcontractor in the design of the Signal Converter Box.

Table 2: Approximate Development Days

System	Development	Documentation
IOC FW	105	40
GUI	9	1
SCB FW (PSI)	60	10
SCB HW	192	Not available

As a reference, work effort was compared to a similar development: the PaSS of CPT’s fix beam line for eye tumour treatment. It was developed from scratch from 2008 to 2009 using FPGAs and EPICS. There are no accurate project management registers. The work effort, shown in Table 3, has been estimated from the date of the opening of the task in our repository to the date of commit of the documentation and source code. For the calculations it is assumed a dedication of 60% to the project and the vacation period is subtracted. Later commitments of bug fixes are excluded.

Table 3: Approximate Development Days

System	Development	Documentation
Old PaSS FW	310	55

From the comparison between the two projects it can be extracted that the use of a modular, reusable architecture based on ICF1210 IOC platform resulted in a time saving of approximately 40%.

CONCLUSION

A reusable, modular Patient Safety System was built to integrate a new commercial gantry in the existing infrastructure of the Proton Therapy System at PSI. By reusing existing technology it was possible to develop a highly sophisticated solution, highly customised, with restricted manpower and time. The separation of the design into generic and gantry specific parts would allow a fast deployment in other facilities, with only small adaptations being needed. Also the new detailed GUI, showing a deterministic log of events in case of an interlock can help the physicists save precious time identifying problems during patient treatment. Finally, by including built-in debug, visibility and measurement elements in the system it is possible to automate some QA tasks and to predict failures by ageing and deterioration of several components.

REFERENCES

[1] Varian Medical Systems, [Online]. Available: https://www.varian.com/oncology/solutions/proton-therapy.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

[2] E. Pedroni et al, "Commissioning and use of the PSI spot scanning isocentric system for proton therapy," in *Proceedings of the 14th International Conference on Cyclotrons and their Applications*, Cape Town, South Africa, 1995.

[3] E. Pedroni et al, "The 200-MeV proton therapy project at the Paul Scherrer Institute: Conceptual design and practical realization," *Med. Phys.,* no. 22, pp. 37-53, 1995.

[4] International Commission on Radiation Units and Measurements, "Prescribing, Recording, and Reporting Intensity-Modulated Photon-Beam Therapy (IMRT)(ICRU Report 83)," 2010.

[5] C. Bula, et al, "Report on Proton Therapy Safety Measures for Gantry 2," Internal report for regulatory body, Villigen, 2013.

[6] IOxOS Technologies, [Online]. Available: http://www.ioxos.ch/.

[7] O. Actis et al, "Gantry 2 Quality Assurance Manual," Internal report for regulatory body, Villigen, 2013.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

NSLS-II ACTIVE INTERLOCK SYSTEM FOR FAST MACHINE PROTECTION*

K. Ha#, S. Seletskiy, Y. Tian, Y. Hu, W. Cheng, O. Singh, D, Padrazo, A. Castiblanco, J. Mead, R. Smith, Y. Tang, W. Guimei, P. Ilinski, J. De Long, L. Dalesio BNL, Upton, NY 11973, USA
G. Shen, FRIB, East Lansing MI 48824 USA

Abstract

At National Synchrotron Light Source-II (NSLS-II), a field-programmable gate array (FPGA) based global active interlock system (AIS) has been commissioned and used for beam operations. The main propose of AIS is to protect insertion devices (ID) and vacuum chambers from the thermal damage of high density synchrotron radiation power. This report describes the status of AIS hardware, software architectures and operation experience.

INTRODUCTION

NSLS-II synchrotron radiation source produces very wide spectrum ranges from IR very hard x-ray. Insertion devices (IDs) and front end devices commissioning started in November 2014 and user operation started in February 2015. The 6 NSLS-II project beamlines consist of EPU, IVU, VUV IDs. The AIS is a key machine protection system that protects vacuum chambers and front end devices from synchrotron radiation power. NSLS-II storage ring (SR) has 792 m circumference and consists of 30 double-bend achromat cells. 180 SR RF beam position monitors (BPMs) are installed around the ring, and 23 ID BPMs are installed in the straight sections. They are all fully functional. The AIS was installed in January 2014 and was successfully commissioned. It has been used in daily operation since November 2014. Recent commissioning status and results of AIS were well described in [1]. NSLS-II AIS has a state-of-the-art architecture that include 30 cell controllers (CCs) and one AIS controller based on a Xilinx Virtex-6 FPGA. The requirement for AIS is to dump the stored beam using RF interrupt within 1ms. It prevents increase of chamber temperature when orbit is mis-steered. We achieved 22 µs global BPM data transfer latency around the ring through 1 to 31 CC nodes. Each of cell controllers contains 5 Gbps transceivers running home-made serial device interface (SDI) protocol. All calculation logic is implemented using FPGA. During beam commissioning, functional test was fully completed with stored beam (1 mA ~ 300 mA) with ID gap open/close, and with ID and bending magnet (BM) photon shutters open/close. The SDI link 10 kHz packet communication and angle offset calculation results were very stable and fast enough for commissioning and user operation. In addition, the measured actual beam dump time is about 1 ms and system hardware latency time is about 200 µs.

*Work supported by DOE contract No: DE-AC02-98CD10886.
#kha@bnl.gov

SYSTEM OVERVIEW

Our novel architecture of AIS is shown in Figure 1. AIS employed a 1 AIS controller, 30 CCs, 203 BPMs, 1 DCCT, 8 front end PLCs. In each cell, local SDI link connected 6 RF BPMs and additional 3 ID BPMs. AIS controller is installed in cell 23 rack group D. Each cell has its own PLC for the integration of equipment protection system (EPS) logic, for ID control and photon shutter controls. This PLC provides ID gap status, BM photon shutter status and ID photon shutter status to the local CC. Each CC communicates with its neighbor cell via 40-m multi-mode fibre optics cables. A ring topology is established by the 30 CCs with SDI protocol. The BPM position data, PLC data, and DCCT status data are delivered to the AIS controller at 10 kHz rate.

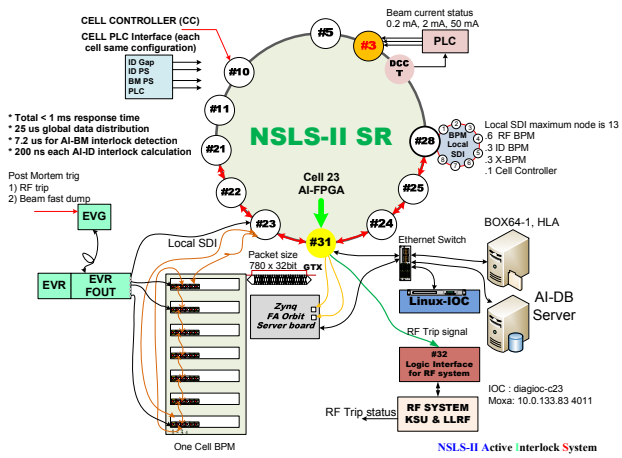


Figure 1: AI system installed layout. (Cell23: AIS and IOC installed, Cell3: DCCT controller installed, Project beamline (Cell 3, 5, 10, 11, 23, 28, 8, 18), and ABIX beamline (16, 17) IDs installed)

The AIS specification is developed following the NSLS-II synchrotron radiation protection requirements [2]:

1. Support multiple IDs and BMs.
2. The AIS controller must cover up to 64 IDs.
3. 10 kHz system monitoring and calculations.
4. Easily extensible for additional IDs which will be installed in the future beamlines.
5. Total system response time is less than 200 µs (BPM to RF trip).
6. Redundant communication.
7. Maximum fast beam dump latency time is less than 1 ms.
8. Slow beam dump time is less than 10 ms (LLRF power ramp down).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

9. Calculate angle and offset simultaneously.
10. Web-based parameters configuration and password protection.
11. Configurable operator enable/disable.
12. Diagnostic functions, such as post-mortem, can be added.
13. Time synchronization for the entire system.

Figure 2 shows the in-house designed AIS controller board and the hosting 1 U chassis (430 cm x 36 cm). AIS controller uses exactly the same hardware as the CC. The only difference is the FPGA firmware. We used same digital front end board for BPM, CC and AIS controller. However, the analogue front end circuit boards are different between these units.

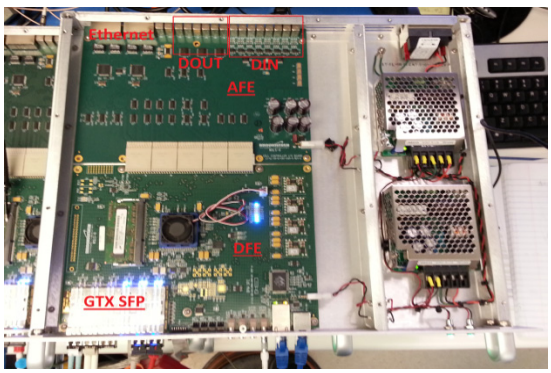


Figure 2: AIS controller

The main technical details of the AIS controller are: FPGA Virtex-6 (xc6vlx240tff1156-1), 2-Gbye DDR-3(MT8JF245664HZ-1GM1) , 5-Gbps GTX, 1-Gbps Ethernet, 16 channel 50-ohm TTL inputs, 12 channel 50-ohm TTL outputs, 16-bit 4 channel DAC.

Post-mortem (PM) Function

For a particle accelerator, PM function is widely used to diagnose the source of a beam dump, which can be caused by RF, power supplies, and other systems. The AIS controller plays the most important role for the NSLS-II PM function.

First of all, the PM trigger is generated by the AIS controller. When the AIS controller detects an abnormal beam orbit (will be discussed in details in the FGPA implementation section), or detects any system status faults (BPMs, PLCs, DCCT etc), it decides to dump the beam via RF system. In the meantime, the AIS controller sends a trigger signal to one of the distribution bus inputs of the event generator (EVG). A global PM timing event is then distributed through timing system to all BPMs, CCs, power supplies, RF systems and other systems to record the PM waveforms. We measured that the latency from the PM trigger to the BPM embedded event receiver is 5.9 µs. The latency includes AIS → EVG → global fiber fanout → 400 m fibre cable → local fiber fanout → BPM embedded event receiver.

Secondly, the AIS controller itself records the 10 kHz data for all BPMs in DDR memory as waveforms for 2 second before the beam dump and for 1 second after the beam dump. These waveforms show how the orbit changed during beam dump, and provide information to investigate which system caused beam dump.

Beam Position Monitor (BPM)

In-house designed high performance sub-nm resolution (200nm for 10Hz data) BPMs were installed and commissioned before March 2014 [3]. During the commissioning, we updated firmware several times for bug fixing and additional functions such as PM and fault detection functions. BPM provides PM waveforms for beam dump diagnostics. These waveforms include ADC raw sum (117 MHz, 256 k points), turn-by-turn x, y, sum (378 kHz, 32 k points), FA x,y (10 kHz, 10 k points).

For the AIS, the most important data from a BPM is the FA data. Each BPM has an embedded event receiver and this enables all the BPMs' FA data be synchronized within 8 ns. Each BPM has dedicated fibre optics to deliver the 10 kHz FA position data to the local CC. The total communication latency for the local 13 BPMs is 2.5 µs.

Cell Controller (CC)

The CC was originally designed for the fast orbit feedback system. Each CC has 16 50-ohm digital inputs and 12 50-ohm digital outputs. The input ports collect the following AIS-related signals and they are sent to the AIS controller via the SDI link:

- ID gap status (Open/Close)
- ID photon shutter (Open/Close)
- BM photon shutter (Open/Close)
- Canting magnet status (Normal/Fail)
  DCCT beam current level & status (0.2 mA, 2 mA, 50 mA, system fail, heartbeat)

DCCT

Beam current is measured by a DCCT which has a 50Hz low pass filter. The analogue output of this DCCT is sent to a PLC. The PLC compares this analogue signal with three pre-defined current levels of 0.2mA, 2mA and 50mA. The results are then sent to 3 digital inputs of CC at cell 3. The DCCT fault status and heartbeat signal are also sent to the digital inputs of the same CC. These digital inputs signals are delivered to the AIS controller via the SDI link.

Front End PLC

Front end PLCs collect various front end device signals, such as ID gap status, BM photon shutter status and ID photon shutter status. These signals are sent to the digital inputs of CC. In addition, the PLC calculates the overall status for the local front end using equipment-protection logic. The result bit is also sent to the digital input of CC. All the digital input signals of CCs are delivered to the AIS controller via SDI link.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

RF

The RF system receives two separate trip signals from the AIS controller. The first one is a fast dump for transmitter trip which controls the Klystron input amplifier PIN diode. The fast dump delay is about 10 µs. The second trip signal is a slow dump for low level RF ramp down in about 10 ms.

FPGA IMPLEMENTATION

For AIS controller FPGA design, Xilinx PlanAhead software package is used for project integration. Verilog HDL is used as the design language. System generator is used for angle/offset calculations. Figure 3 shows FPGA block diagram with auxiliary devices.

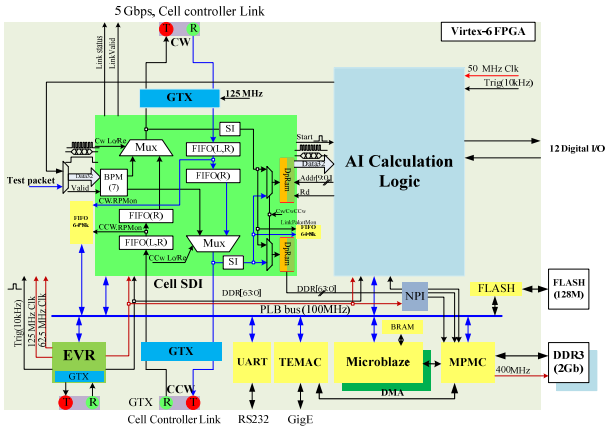


Figure 3: FPGA internal block diagram

The AIS controller obtains the data of equipment protection, front ends system, DCCT, cell controller status and BPM position at every 10 kHz from the SDI link. The RF system faults and power supply faults make a real beam orbit distortions. The overall faults are:

- BPM fault : PLL unlock, ADC saturation
- RF fault : low level RF, transmitter
- Power supply driving abnormal current: dipole, quads, sextuple, skew, correctors
- EPS/PPS malfunction detection
- BM BPMs positions out of envelop
- ID BPMs positions and angles out of envelop
- DCCT fault

AI-BM Level Comparator

AI-BM interlock logic is engaged when the beam current is larger than 50 mA. The AIS controller compares the reference orbit and the 10 kHz x/y position data of the 180 RF BPMs (6 BPMs per cell). If the difference between them exceeds acceptance envelope (+/- 5 mm in horizontal plane and +/- 3 mm in vertical plane) at any BPMs, the AIS controller generates interlock signal to trip the LLRF system and dump the beam. At the same time, the AIS controller sends the PM trigger to timing system and latches all the status data and all the calculation results.

AI-ID Angle and Offset Calculation

In order to protect the sensitive IDs and x-ray front end devices, AI-ID uses a more complicated algorithm. It requires an angle and offset calculation at each ID by using two neighbor BPMs position data. The basic angle calculation concepts are described in [2]. Here is the formula to calculate the offset and angle in H plane:

Angle = ((x2-x2Offset)-(x1- x1Offset)) * S_MRAD;
Offset = ((x1- x1Offset)*(s2-s3)+(x2- x2Offset)* S_MM;

where
x1 is BPM1 x position, x2 is BPM2 x position
s1 is BPM1 button location from center
s2 is BPM2 button location from center
s3 is centre position
x1Offset is BPM 1 horizontal geometry offset
x2Offset is BPM 2 horizontal geometry offset

The above calculation is carried out in the FPGA of AIS controller. To minimize the calculation in FPGA hardware, the two parameters, S_MRAD and S_MM, are calculated using Microblaze following the equations:

S_MRAD = (1/(s2-s1))
S_MM = ((s3-s1)/(s2-s1))

If the angle or offset are out of tolerance, the AIS controller generates beam dump signal to RF system and also generates PM trigger to timing system.

SDI Protocol for GTX Fibre Communication

At NSLS-II, a novel communication protocol called SDI is designed to support fast orbit feedback (FOFB) system and AIS. It is stable, robust and proven to have good performance during the commissioning and user operations. For SDI, the data rate is 5 Gbps and 8b/10b data encode is used. Each CC collects total of 26*32bit data packages from the local BPMs. The total global SDI data size is 26*32bit*30 = 780*32bit =3120bytes.

FOFB reads the global SDI data and uses it for feedback calculation at 10 kHz rate. The AIS controller reads this SDI data for angle and offset calculations, and for system fault detection. The global SDI data is synchronized with timing system. An SDI package has a flexible user-defined simple structure including header, data and CRC. The data packet has variable length and can be defined in Microblaze initialization routine. The SDI total communication latency was measured to be about 22 µs all of the 30 cells with the total packet size of 3120 bytes.

SOFTWARE

Software part of AIS consists of IOC and Web interface, relational database and CSS display.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

EPICS IOC

At NSLS-II, we use IBM Linux servers to host EPICS IOCs. In each cell, there is one EPICS IOC for the control of local BPMs, and one for FOFB. The AIS EPICS IOC is located in cell 23 rack group-D. EPICS IOC database provides an alarm status for the following faults:

- IOC to FPGA TCP/IP communication failure (major)
- RF dump signal (major)
- AI-ID interlock (major)
- AI-BM interlock (major)
- ID BPM fail (major)

We developed special soft IOC driver called portable streaming controller (PSC) for the TCP/IP communication between the EPICS IOC and the FPGA. The PSC uses user-defined simple protocol and driver is able to quickly transfer large waveform data between EPICS IOC and FPGA. One advantage of the PSC is that it utilizes different TCP/IP ports for FPGA to IOC streaming and IOC to FPGA streaming [4].

Web Interface and DB

Since the AIS is an equipment safety-related system, all its parameters are password protected. All AIS parameters are configured by a web based client program and it requires password for login and modifications. Only authorized people can modify the configuration and download it to FPGA memory. MySQL is used as back-end relational database to save the configuration parameters. Data service is implemented using the Django framework.

PM-Waveform Archiving

Waveform archiving software is used by CA library and saved to HDF5 file. A client program continuously monitors PM status PV. If the PM is detected, the client program reads all waveform PVs and saves them on remote file system. Currently the PM client program is archiving waveforms of RF system, power supplies, BPMs, CCs, and the AIS.

CSS Display

Control system studio (CSS) is used as high level control and monitor program. OPI page provides two different pages for AIS: one expert page and one general user page. The expert page includes all control and monitor functions while the general user page only provides system monitoring.

COMMISSIONING AND OPERATION

During commissioning we leaned important things that problem is passible by human mistake and firmware timing error. For avoid of these errors from unexpected conditions we modified epics database and BPM, AI firmware. All previous-cycle data is latched as internal registers to provide details when an interlock is detected or other failure event happens. PM waveform data are stored in local DDR-3 memory and used to provide diagnostics functions for failure analysis and event analysis. Figure 4 shows measured superconducting RF cavity field and ADC raw sum (117 MHz) signals during RF trip and beam dump.

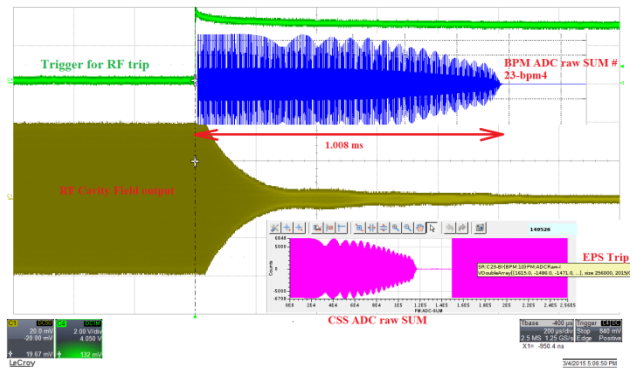


Figure 4: RF Fast dump latency measurement (acting time is 20 µs, total beam loss time is 1 ms)

SUMMARY

In 2009, we started the hardware and firmware design for BPMs, FOFB and AIS with SDI protocol. We achieved remarkable performance: 22 µs global SDI communication latency, 46 µs latency from BPM to AIS output signal for RF dump, and 5.9 µs of global PM trigger latency. We confirmed there is no error in SDI communication between 30 CC nodes and one AIS controller. In September 2014 we started AIS commissioning with up to 150 mA beam current and ID operations. In September 2015 we confirmed all designed functions with 300 mA beam current and top-off operation. All system functions and reliability are proven during the commissioning and operation in the last two years. The AIS will include more beamlines when the ABXI and NEXT beamlines are built until 2018.

ACKNOWLEDGMENT

The authors give great thanks to the RF and diagnostics group for support this project. A special thanks to M. Maggipinto and C. Danneil for their strong support. Also very special thanks to M. Davidsaver who developed the PSC driver.

REFERENCES

[1] S. Seletskiy et al., "Commissioning of Active Interlock System for NSLS II Storage Ring," TUPMA057, these proceedings, Proc. of IPAC15, Richmond, VA, USA

[2] Petr Ilinski, Technical specification for Active Interlock Envelops and Active Interlock System Settings for NSLS-II

[3] J. Mead et al., "NSLS-II BPM Commissioning Update," WECYB2, Proceeding IBIC2014, Monterey, CA USA

[4] http://irfu.cea.fr/Meetings/epics/ presentations/ Thursday/pscdrv-201410.pdf

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

MACHINE PROTECTION SYSTEM FOR THE KOMAC 100-MeV PROTON LINAC

Young-Gi Song, Kyung-Tae Seol, Han-Sung Kim, Dae-il Kim, Sang-Pil Yun, Hyeok-Jung Kwon, Yong-Sub Cho, Korea Multi-purpose Accelerator Complex, Korea Atomic Energy Research Institute, Korea

Abstract

A Machine Protection System (MPS) is one of the important systems for the 100-MeV proton linear accelerator of the Korea Multi-purpose Accelerator Complex (KOMAC). The MPS is required to protect the very sensitive and essential equipment during machine operation. The purpose of the MPS is to shut off the beam when the Radio-Frequency (RF) and ion source are unstable or a beam loss monitor detects high activation. The MPS includes a variety of sensors, such as beam loss, RF and high voltage converter modulator faults, fast closing valves for vacuum window leaks at the beam lines and so on. The MPS consists of a hardwired protection for fast interlocks and a software protection for slow interlock. The hardware-based MPS has been fabricated, and the requirement has been satisfied with the results within 3 μs. The Experimental Physics and Industrial Control System (EPICS) control system has been also designed to monitor and control the MPS using a Programmable Logic Controller (PLC). This paper describes the design and implementation of the MPS for the 100-MeV proton linear accelerator of the Korea Multi-purpose Accelerator Complex (KOMAC).

KOMAC FACILITY OVERVIEW

The Korea multi-purpose accelerator complex (KOMAC) 100-MeV proton linac has been developed and has been installed at the Gyeong-ju site. Figure 1 shows a schematic layout of the KOMAC 100-MeV proton linac and beam lines [1].

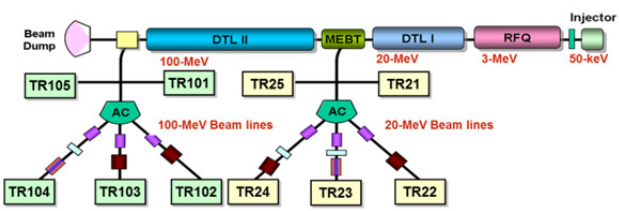


Figure 1: KOMAC proton linac consists of an ion source, a Low-Energy Beam Transport (LEBT), a Radio-Frequency Quadrupole (RFQ), a Draft Tube Linac (DTL), a Medium-Energy Beam Transport (MEBT), beam-lines, and a Target Room (TR) for 20-MeV and 100-MeV beam.

The KOMAC consists of low-energy components, including a 50-keV ion source, a low-energy beam transport (LEBT), a 3-MeV Radio-Frequency Quadrupole (RFQ), and a 20-MeV Drift Tube Linac (DTL), as well as high-energy components, including seven DTL tanks for the 100-MeV proton beam. The KOMAC includes 10 beam lines, 5 for 20-MeV beams and 5 for 100-MeV beams. The peak beam current and maximum beam duty of the 20-MeV linac are 20 mA and 24%, respectively. The peak beam current and maximum beam duty of the 100-MeV linac are 20 mA and 8%, respectively. There are 4 high-voltage convertor modulators. Each modulator drives 2 or 3 klystrons. The peak output power is 5.8 MW, and the average power is 520 kW with a duty of 9%. The pulse width and the repetition rate are 1.5 ms and 60 Hz, respectively.

MPS OVERVIEW

The radiation from the beam loss and faults of the linac components can cause substantial damage to the devices. The KOMAC active protection system needs to minimize the beam loss radiation and ensure the safe operation of the machine [2]. The purpose of a machine protection system (MPS) is to turn off the beam and the sub-systems when an interlock occurs. The MPS consists of two parts: a hardwired-based fast interlock system using an analog circuit and a software-based slow interlock system with a Programmable Logic Controller (PLC). The interlock systems are designed as main unit for machine protection and are fabricated with analog circuits. The PLC includes software logic that checks the result of the machine protection system on an interlock signal and also turns off components related to the interlock. Figure 2 describes a flow chart of machine protection using hardwared and software based interlock system.

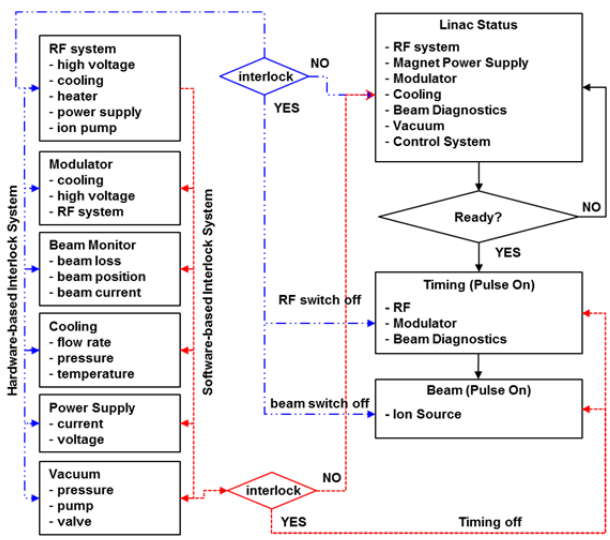


Figure 2: Block diagram of interlock system for machine protection with hardware and software based interlock system.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The basic concept is that the main interlock system turn off the beam and be connected with distributed local interlock systems for sensitive devices, such as the High Voltage Convertor Modulator (HVCM), the Radio Frequency (RF) system, beam loss, the cooling system, and the beam extraction power supply. A local interlock system collects signals from sensors and report to interlock systems for an RF and an ion source. The RF interlock system acts as the interface to a RF switch system. The beam interlock system also receives input from beam loss and position monitoring circuits. A MPS control screen monitors all interlock signals of the machine components, including the vacuum and the timing systems, as well as sensitive devices, and turns off and prevents a beam.

MPS DESIGN

The MPS is essential devices for machine protection by turning off beam or RF. If the essential equipment is to be protected, an MPS must be designed to identify a variety of device failures around the linac and the beam lines. The machine operation must be done in a reliable protection system. The goal of the machine protection system is to turn off the RF system or the beam of a microwave ion source within a few microseconds when some failure of the RF system or beam loss occurs during the beam operation mode in which the beams are switched to beam lines. Table 1 describes the interlock lists of the machine protection system.

Table 1: List of Interlock Signals for Machine Protection of the KOMAC Linac

List	Interlock signal
High-power RF source	Klystron heater, magnet, and ion pump Low level RF Waveguide circulator and RF window arc
RF cavity	Cavity vacuum, RF arc Cooling water (flow rate, temperature, RCCS, wall cooling)
High-voltage power supply	Equipment fault Personnel fault
Beam	Beam loss, beam position
Beam line	Fast closing valve Safety block

The block diagram of the interlock system, which is developed using a fast-trip direct analog circuit, is shown in Figure 3. The interlock system consists of fast analog interlock modules like comparators and latches, an auto-reset module, a Voltage Standing Wave Ratio (VSWR) module for only the RF interlock unit, and a power supply [3]. The response time of the fabricated interlock system was measured to be within 3 μs. The response time satisfies the requirement for the KOMAC machine protection, which states that the beam must be stopped within a few milliseconds when any device failure occurs during 60 Hz beam operation.

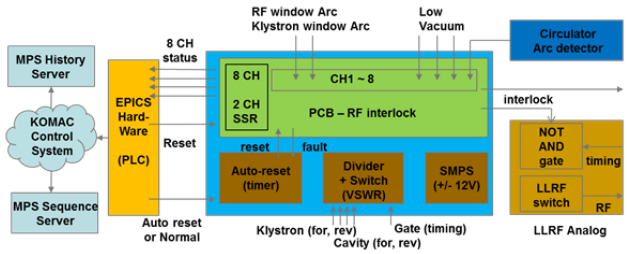


Figure 3: Block diagram of the signal path of interlock system developed by using a fast trip direct analogue circuit.

Local interlock systems are installed into each klystron and RF rack of klystron galley, as shown in Figure 4, to prevent a beam when a device failure is activated. Each local interlock system sends the interlock signal to the interlock system for the ion source, and the beam is shut off by turning off the extraction power supply of the ion source. It is also possible to shut off the RF power of the RFQ to decelerate the beam in the linac. When the interlock signals of the RF system and the HVCM occur, the local interlock systems also send the interlock signal to a switch in the low-level RF system to shut off the RF power to each cavity [4].

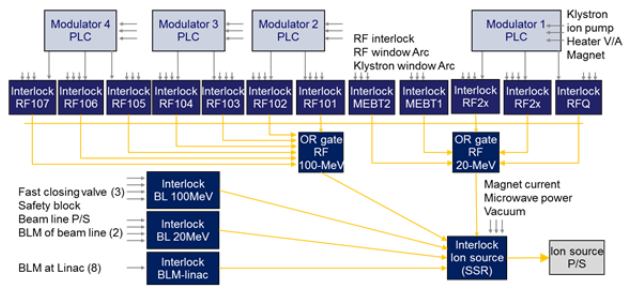


Figure 4: Schematic layouts of the hardware based MPS to protect the linac components from the high-power RF system, low-level RF, beam radiation, and modulator fault. The MPS shuts off the beam-extraction power supply of the ion source.

The interlock systems are essential devices for personnel safety and machine protection by turning off beam. Also, in order to support sequential operation based on interlock signals, software-based monitoring system is designed using the Experimental Physics and Industrial Control System (EPICS) framework and industrial PLC [5]. The PLC collects local interlock signals from local MPSs as well as vacuum and cooling components. The PLC I/O chassis is distributed into the linac gallery and communicated with a main PLC CPU system through ControlNet. An EPICS Input Output Controller (IOC) for the PLC is integrated with the KOMAC main control systems through the EPICS Channel Access (CA). Figure

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

5 shows a user interface screen for the interlock status and alarm.

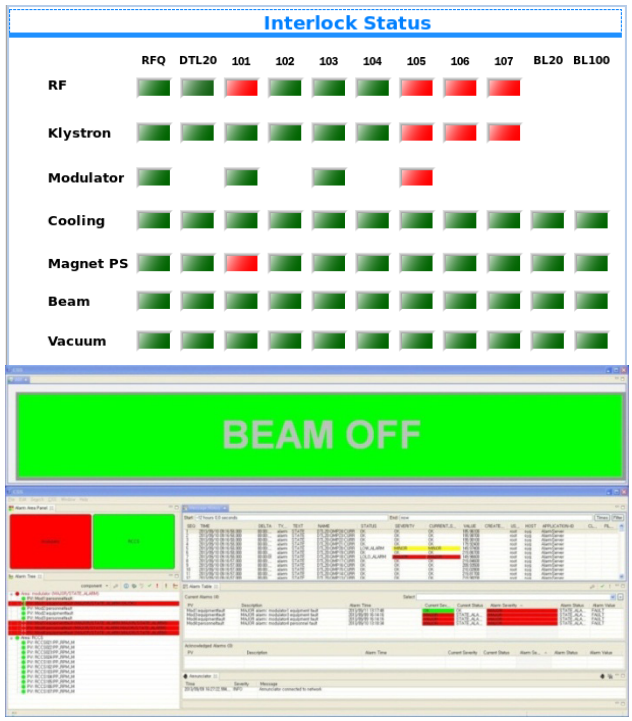


Figure 5: User interface screen for the interlock and alarm status.

The beam permission in the linac and beam lines is controlled by local interlock systems, Personnel Protection System (PPS), and Target Room (TR). One of the failures will activate a beam stop.

There is upgrade plan for KOMAC MPS that is under development using fully programmable system based on Field Programmable Gate Array (FPGA). The new MPS is one module type for all measurements. The interlock sensor interface is totally flexible. The MPS utilizes individual processors which are independent from other hardware.

CONCLUSION

A hardwired based interlock system was developed to protect critical devices from beam-induced damage, such as excessive beam losses and faults at high-power components. The local interlock system for a fast interlock has been fabricated, and its response time was within 3 μs. This response time satisfied the requirement for machine protection. The requirement specifies that a beam must be prevented within tens of microseconds during beam operation. A beam can be accelerated under conditions that protect both machine and personnel using the MPS.

ACKNOWLEDGMENT

This work has been supported through KOMAC operation fund of KAERI by MSIP (Ministry of Science, ICT and Future Planning).

REFERENCES

[1] Y.S. Cho et al., “The KOMAC Accelerator Facility”, Proceedings of IPAC’13, Shanghai, China (2013).
[2] SNS report, SNS Machine Protection System, 2001.
[3] K.T. Seol et al., “Design of Machine Protection System for the PEFP 100 MeV LINAC”, Proceedings of IPAC’12, New Orleans, USA, (2012).
[4] Y.G. Song, “Interlock system for machine protection of the KOMAC 100-MeV proton linac”, J. Korean Phys. Soc. 59, 577 (2013).
[5] Experimental Physics and Industrial Control System (EPICS), URL:http://www.aps.anl.gov/epics

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SAFETY INTEGRITY LEVEL (SIL) VERIFICATION FOR SLAC RADIATION SAFETY SYSTEM

F. Tao, E. Carrone, J. Murphy, K. Turner
SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA

Abstract

Safety Integrity Level (SIL) is a key concept in functional safety standards. SIL is a performance measure on how reliable a safety system is in performing a particular safety function. To comply with standards, during the system design stage, SIL verification must be performed to demonstrate that the SIL achieved meets/exceeds the SIL that was assigned during risk assessment. Unlike industrial applications, where safety systems are usually composed of certified devices or devices with long usage history, safety systems in large physics laboratories are less standardized and more complex in terms of system architecture and devices used. In addition, custom designed electronics, with limited reliability information, are often employed. Verifying SIL for these systems requires in-depth knowledge of reliability evaluation. In this paper, it is demonstrated how to determine SIL using SLAC radiation safety systems (Personnel Protection System (PPS) and Beam Containment System (BCS)) as examples. PPS utilizes commercial safety rated devices, while BCS contains customized electronics. Choice of standards, methods of evaluation, reliability data gathering process (both from industry and from hardware development) are also discussed.

INTRODUCTION

At SLAC National Accelerator Laboratory, there are three protection systems deployed to mitigate radiation risks: Machine Protection System (MPS), Beam Containment System (BCS) and Personnel Protection System (PPS). While MPS is focused on protecting equipment from getting damaged, the other two systems are critical to protect personnel and the environment from getting a radiation dose. In comparison with the MPS, PPS and BCS have very rigorous configuration control policies in place and follow all due diligence in engineering practices. For these reasons, these two radiation safety systems meet all definitions of safety-critical control systems.

Functional safety standards started in 1990's and now there are several critical standards have been developed and adopted worldwide. Such as IEC61508 [1] and corresponding sector specific standards IEC61511 [2] (process), IEC 62061 [3] (machinery). ISO also published a machinery safety standard ISO13849 and its relationship with IEC62061 is described in a IEC/ISO joint technical report ISO/TR23849 [5]. While industries have started following functional safety standards in implementing safety-critical control systems two decades ago, the natural question that arises is, are the same standards applicable to accelerator safety systems, and if the answer is yes, how are the standards applied so that we can learn from industries' long time experience with the engineering design and operation of safety systems.

As found in any functional safety standard, two key concepts are safety lifecycle and safety integrity level. The former describes a series of activities in system engineering and operation to make sure all risks are identified and properly mitigated. The latter is a measure of the reliability performance each safety function within the safety system, such that the design be precisely performance based and is less conservative than the traditional description-based approach.

In all functional safety standards, after the conceptual design stage, the SIL of the safety function must be verified to make sure it meets or exceeds the SIL assigned in the Safety Requirements Specification (SRS). In the case the safety system/function SIL level is not met, the safety system/function must be re-designed. For example, the following figure is from IEC 61511.

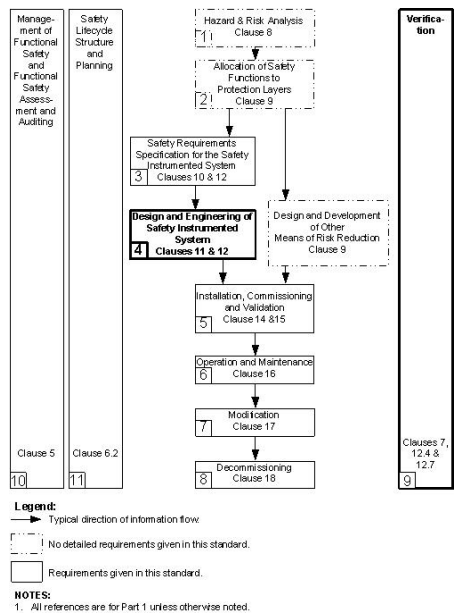


Figure 1: Safety System Lifecycle Phases.

Though PPS and BCS are both classified as safety-critical systems with the purpose to mitigate radiation risks, different philosophies and technologies are used. The majority of the PPS devices are commercial off-the-shelf (COTS), their reliability data can be obtained and the system level reliability block diagram (RBD) can be

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

quickly built. On the contrary, BCS has many accelerator unique sensors such as beam loss monitors, beam current monitors etc. The system needs to response to these sensor signals in a very short time, 100uS for the LCLS-II BCS, and bring the system into the safe state. Commercial off-the-shelf products do not meet the required shut off times nor are their interfaces compatible with unique sensors found in accelerators. These reasons contribute to the need for custom electronics designed hardware.

For these reasons, the SIL verification for these two systems face different challenges and need to utilize different methodologies.

PPS SIL VERIFICATION

The Personal Protection System (PPS) is responsible for keeping people away from beam. The system is composed of an access control systems and a safety interlock systems. At SLAC, this configuration is implemented with a 3 PLC architecture, one access control PLC responsible for access related functions as well as acting as a communication bridge between EPICS and safety controllers. For safety interlocks, there are two redundant stand-alone safety PLCs in an A/B chain configuration to independently monitor the safety interlock conditions. A typical PPS installation is shown in the following figure.

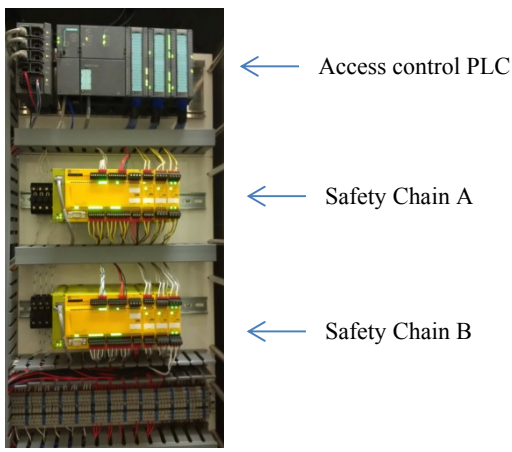


Figure 2: A Typical PPS Installation.

Access control functions are generally classified as non-safety critical, so usually no SIL number is assigned or verified during the design stage. However, if other systems such as Oxygen Deficiency Monitoring (ODM) system need to use "access control system" as one protection layer, the this system can be regarded as one protection layer with SIL1 capability if only the rigorous configuration control is in place.

Compared with access control functions, those interlock functions implemented within the PPS safety PLCs are more critical. Their major functions are to detect that the beam operation safety boundary is secured during beam operation, and that there is no excessive radiation in occupiable areas. Typical inputs for these functions are micro-switches and area radiation monitors whereas the outputs are stopper/RF permits.

Figure 3 shows the RBD for E-stop interlock in the photon area. In this example, inputs are Emergency Off buttons and the outputs are two solenoid valves. Pilz PNOZmulti safety controller can de-energize the solenoid valves and 2 beam stoppers will move in due to the gravity force.

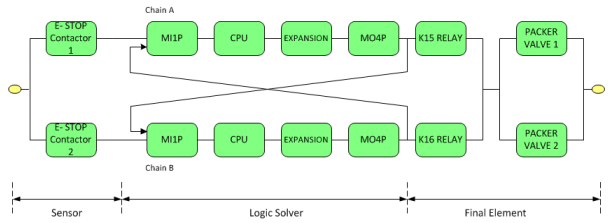


Figure 3: RBD for PPS E-stop.

As this is not a standard architecture that appears in any functional safety standard, its reliability cannot be calculated by using the standards or commercial software. However, standards do not expect a very accurate reliability calculation, but accentuate that the results must be conservative such that the safety system is not under-designed. Therefore, we can use the "cut set" concept in reliability engineering and quickly obtain the performance bounds of the above configuration:

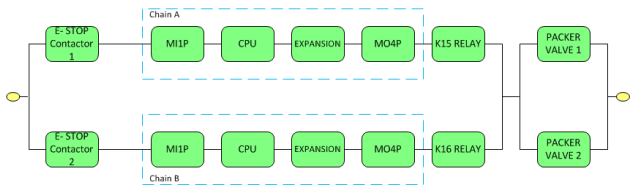


Figure 4(a): equivalent RBD with lower reliability.

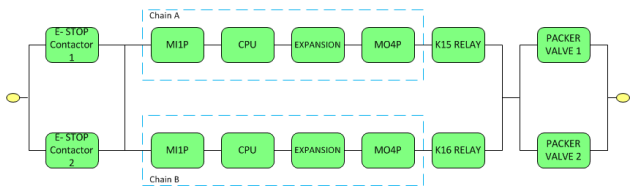


Figure 4(b): Equivalent RBD with higher reliability.

The two RBDs shown in Figure 4 are standard configurations, and their reliability can be calculated using the standard 1oo2 configuration formula. Since the configuration is redundant, the common cause failure dominates the reliability performance. In this case, with the common cause factor being considered, the difference between two bounds is small and we can simply use the result associated with Figure 4(a) as approximation.

The common cause/mode failure portion is usually represented as $\beta\lambda_{DU}$, where λ_{DU} is the dangerous undetected failure rate of a single channel. Common configuration is when two or more redundant channels are

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

identical, if this is not the case, the λ_{DU} can be replaced with the geometric average for two or more channels. On the other hand, in the existing formulae, if there are multiple redundant channels, common cause or common mode failures are less likely to occur at the same time and the system reliability is improved. To properly give the credit for this configuration, the modified Beta- method is given in [6] as:

$$\beta(MooN) = \beta C_{MooN}$$

Table 1: Common Cause Factors for Different Voting Logics

M\N	N=2	N=3	N=4
M = 1	C_{1oo2}= 1	C_{1oo3}=0.5	C_{1oo4}=0.3
M = 2		C_{2oo3}=2.0	C_{2oo4}=1.1
M = 3			C_{3oo4}=2.9

PPS interlock to electron stoppers are quite often triple or quadruple, and the modified Beta-method will be useful in the reliability calculation.

Another challenge for PPS SIL verification is to obtain the reliability data for some large accelerator unique equipment, such as modulator (providing the power to klystron) and Variable Voltage Substation (VVS) (providing the power to modulators).

To analyse the reliability performance for PPS functions interlock to these devices, we have to look into the detailed schematics of those devices to create the RBD and perform the SIL verification. For example, a control reliability analysis for a VVS was performed and it was found that the configuration is vulnerable to control power loss:

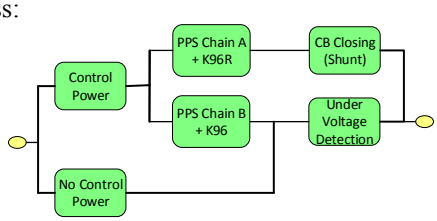


Figure 5: RBD for VVS interlock to PPS.

To calculate the reliability of such a scheme, we will need to substitute in the RBD into "PPS Chain A" and "PPS Chain B" to get the final results. A noticeable shortcoming of such a configuration is its dependence on control power. A simple solution to improve the overall system reliability would be power line monitor and alarming. So that the alarm will notify operators the power loss, and corresponding remedy actions can be taken to mitigate the risk.

For power system devices, their reliability analysis method as well as failure rates can be found in IEEE 493 standard. However, if the device is special, customized or has been modified, then site specific reliability data is more valuable.

In the VVS interlock case, the PPS Chain A controls a 2000A circuit breaker, whose failure rate can be obtained from IEEE 493, but the Chain B controls a SLAC modified device which is a customized modification. With the aid of SLAC CATER (Comprehensive Accelerator Tool for Enhancing Reliability), past 20 years of service records for that device can be found. Based on that data, the failure rate of chain B "under-voltage detection" mechanism can be calculated. This highlights the importance of keeping records of site specific operational data, which is valuable in system reliability evaluation.

A standard interface to modulators has been implemented as well. Commercial safety circuit breakers will be used to meet the PPS requirements on reliability and safety performance.

These breakers are designed to meet ISO 13849 machinery safety standards with manufacturer provided device failure rate data. The auxiliary contacts comply with IEC60947 standard, such that the contact can always provide the true status feedback to PPS for setting access to accelerator, which is also a safety instrumented function.

It should mention that when IEC 61508 was first published in 1998, there was no detailed explanation on how those equations in Part 6 were developed. It caused confusion and people made mistakes when trying to extend the results to non-standard configuration. With the Recent publications of technical reports and a research monograph [6-8], practitioners in this field will have better understandings on reliability modelling of safety systems.

BCS SIL VERIFICATION

The BCS for the LCLS-II project will adopt a hybrid architecture to meet safety, reliability and response time requirements. For those interlocks needs slow response times, sensor inputs directly connect to Safety PLCs. But for those fast sensors that require fast response times, customized designed electronics have to be developed and deployed to achieve these speeds.

Illustrated in Figure 6, the PLC architecture is not implemented in the same way as with the PPS. A partial reason for this is to reduce the system hardware cost but still follow the recommended configuration from the PLC manufacturer. With this PLC configuration, the system is still SIL-3 capable as suggested by the PLC safety manual.

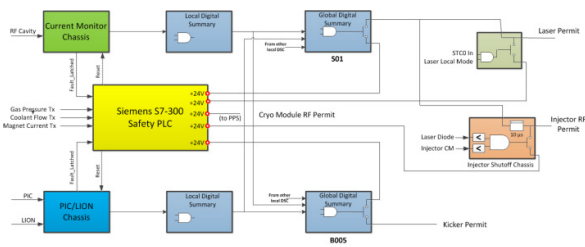


Figure 6: LCLS-II BCS Diagram.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

For customized electronics, reliability data is not immediately available therefore the reliability data has to be estimated from the electronic design schematics. For this task, IEC 61508 has to be followed to make sure that other design requirements are met before the design is considered suitable to be used in SIL rated safety applications. In the design process, design for reliability should be established at beginning in addition to functional requirements. Failure Modes Effects and Diagnostics Analysis (FMEDA) should be carried out to get quantitative information of the design. Then Diagnostics Coverage (DC), minimal fault tolerance and Safe Failure Fraction (SFF) has to be calculated and comply with requirements from IEC 61508. Once the structural constraints are met, further details on reliability performance can be quantified. There are many data sources on electronic parts available, such as [9-11]. Hardware design engineers should combine these sources to get the reliability information requested by IEC 61508. Additionally, SIL verification can still follow the methodology given in standards, technical reports as if the data is provided by "manufacturer".

CONCLUSION

SIL verification is an important step in functional safety standard compliance. The main purpose is to evaluate the reliability of each safety function to make sure it meets the targeted risk reduction requirement. For safety systems composed of commercial off-the-shelf devices, we can build up the reliability block diagram and fill in the reliability data to get the result. Site specific reliability data is more valuable for site specific devices. For customized designed electronics, a rigorous IEC61508 development process needs to be followed to make sure the pre-determined SIL capability is obtained as well the reliability predication information of the overall design.

REFERENCES

[1] IEC 61508, "Functional Safety of Electrical /Electronic /Programmable Electronic Safety-related Systems", 2nd Ed, IEC, 2010.
[2] IEC 61511, "Functional safety - Safety instrumented systems for the process industry sector", IEC, 2004.
[3] IEC 62061, "Safety of Machinery- Functional Safety of Safety-related Electrical, Electronic and Programmable Electronic Control Systems", IEC, 2005.
[4] ISO 13849, "Safety of Machinery – Safety-related Parts of Control Systems", ISO, 2006.
[5] ISO/TR 23849, "Guidance on the application of ISO 13849-1 and IEC 62061 in the design of safety-related control systems for machinery", ISO, 2010.
[6] ISO/TR 12489, "Petroleum, Petrochemical and Natural Gas Industries – Reliability Modelling and Calculation of Safety Systems", ISO, 2013.
[7] ISA-dTR84.00.02, "Safety Integrity Level (SIL) Verification of Safety Instrumented Functions", ISA, Jan. 2015.
[8] M. Rausand, "Reliability of Safety-critical Systems", John Wiley&Sons, 2014.
[9] Telcordia SR-332, "Reliability Prediction Procedure for Electronic Equipment", Issue 3, Telcordia, Jan. 2011.
[10] IEC TR 62380, "Reliability data handbook-Universal model for reliability prediction of electronics components, PCBs and equipment", IEC, 2004.
[11] "Reliability Modelling: The RIAC Guide to Reliability Prediction, Assessment and Estimation", RiAC, 2010.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE LMJ TARGET DIAGNOSTICS CONTROL SYSTEM ARCHITECTURE

S. Perez CEA/DIF, Bruyères le Châtel, 91297, Arpajon, France
T. Caillaud, CEA/CESTA, Le Barp, 33114, France

Abstract

The French Laser Megajoule (LMJ) is, behind the US NIF, the second largest inertial fusion facility in the World. More than 30 diagnostics will be installed and driven in a huge and complex integrated computer control system. The aim of this paper is to describe an architecture based on the TANGO open source software for the very low level control system, Python language for the development of drivers and the French commercial PANORAMA© software as the main high level SCADA.

This choice leads to guaranty the evolution of the middleware software architecture of this facility supposed to be operated during dozen of years with the capability of using many instruments including sustainability.

INTRODUCTION

Since it definitively abandoned nuclear testing, France has relied on the Simulation Program to guarantee the operational performance and safety of its nuclear deterrent weapons throughout their lifetime.

Successful simulation requires both:

- Qualified computer codes that integrate laboratory-validated physics models to simulate weapon functioning;
- Teams of qualified physicists to use these codes.

In this respect, the Megajoule Laser (LMJ [1]) plays a vital role, as it is used to validate the numerical codes and certify the skills of French physicists.

On October 23, 2014, French Prime Minister Manuel Valls declared the facility operational after starting up the first experiment.

Target diagnostics are a key for numerous physical data acquisition. CEA will develop dozen of these equipments during next twenty years. Each target diagnostic will be dedicated to one or several kind of measurements like X-ray, visible, UV or particles like neutron…

During the life cycle of the LMJ Facility, CEA needs to implement new kind of equipments compliant with a stabilized control command system.

This paper describes the command control architecture used for target diagnostics and the reasons why we insure sustainability for such a huge modular installation.

We will describe the 2 layers of the Target Diagnostics Control System (TDCS) and particularly the use of TANGO for Layer 0, which guaranty modularity and life time expectancy, and the French SCADA PANORAMA E² [2] for Layer 1 dedicated for every LMJ command control subsystem.

Maintenance and qualification tools will also be described. The use of Open Source Software like TANGO, Python and QT will allow the capability for diverse contractors to insure all future developments.

USING A TARGET DIAGNOSTIC

In 2014-2015, three different target diagnostics have been installed: two X-ray imaging systems (different ranges of energy and waveform) and a complex diagnostic used for Hohlraum temperature measurements including an absolutely calibrated broadband X-ray Spectrometer, a Gating Spectrometer and a time resolved Imaging System of the emitting area.

This paper will focus on a « simple » X-ray imaging system. This diagnostic can actually be compared to a giant microscope. Made of 4 parts (alignment beams, a telescopic motorized arm, filters and a framing camera), the command control configures the equipment, then focus it using alignment and the arm, in order to acquire data from an optical camera (Fig. 1).

Figure 1: an X-Ray Target Diagnostic.

Each part of the diagnostic has to be driven by the command control and, as these functions should be reused in future target diagnostics, a modular command control architecture must be designed.

THE LMJ COMMAND CONTROL ARCHITECTURE

The LMJ command control architecture is driven by the 4 classical component layers, as shown in Figure 2 :

- Layers 2 and 3 are devoted to the common control system (administration, main supervisory, prediction and tuning system [4] [5], sequences [6]…)
- Layer 1 is set for the main subsystems command controls (target and laser diagnostics, synchronization [7]…) and interfaces between them.
- Layer 0 is the main layer for equipment communications. It includes drivers, communication protocols as well as maintenance and qualification tools.

There is one Layer 1 for the all TDCS and as many as necessary drivers included in Layer 0 for each defined

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

target diagnostic. PANORAMA E² is the Framework used for Layers 1 to 3 (Fig. 2).

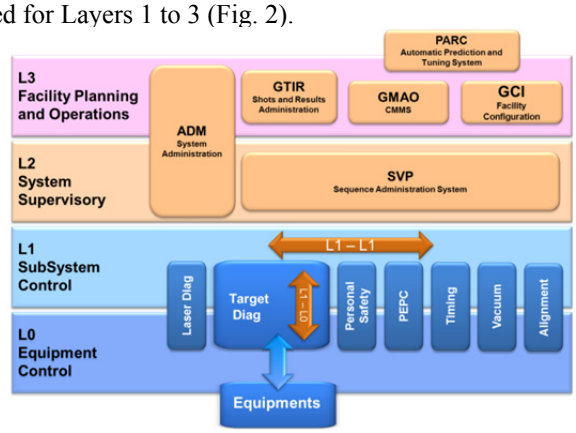


Figure 2: The LMJ Command Control Architecture.

Within that configuration it seems necessary to use a modular Layer 0 architecture and established PANORAMA Layer 1 software that accepts futur Target Diagnostics as new plugin elements.

LAYER 0

A way to make Layer 0 modular is to develop hardware to software abstraction layer between each equipment and the command control. This kind of architecture has been used by Microsoft with DirectX© or by National Instrument MAX© system. Using an abstraction layer allows equipment sustainability: Layer 1 software does not have to be modified even if the low level equipment changes. Only the equipment driver has to be updated but the low level interface will remain the same (Fig. 3).

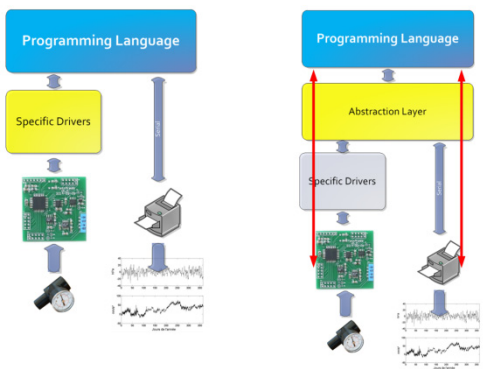


Figure 3: An Abstraction Layer Architecture.

TANGO as Layer 0

TANGO is a software architecture that have the following characteristics:

- Uses an abstraction layer,
- Has been used from several years in huge installations (mostly European synchrotrons),
- Supports different operating systems (Linux and Windows),
- Allows modularity and, by the way, includes a lot of instrument drivers,
- Can be programmed in different languages (C++, Java, Python),
- Part of a large community,
- Available as an open source architecture [3].

Next figure shows how Tango architecture matches our needs for modularity (Fig. 4).

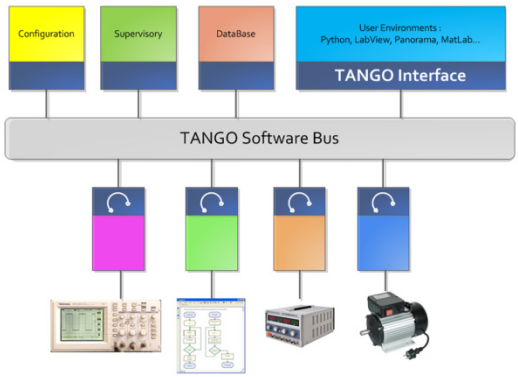


Figure 4: The Tango Architecture.

TANGO framework comes with several tools that simplify low level development phases.

In TANGO, a driver is usually called a Device Server (DS). The DS communicate with each associated equipment and is able to deliver information to the TANGO software bus in different ways (push, pull, event…).

A DS can also be a single process software whose methods are shared between other DS (barycenter and matrix computations for instance).

Each DS is made of:

- Methods (a typical list of functions that will be available for Layer 1 or for other DS)
- Properties (items configurations like motor characteristics, IP addresses…)
- Attributes (values changed by the equipment like arm positions, motor currents, specific acquired datas…)

One main TANGO development tool is Pogo, used to generate both DS framework and DS documentation with a choice of 3 different languages for writing the drivers.

The whole driver skeleton and interfaces are generated; the developer « only » have to fill up the communication protocol between hardware equipment and the corresponding methods.

14 specific « States », are used for states machines in Pogo. They are used to define the real condition of the equipment (for instance « MOVING » for an arm, « OPEN » and « CLOSED » for an obturator…). These states are also used in the Layer 1 control software or by other DS (Fig. 5).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

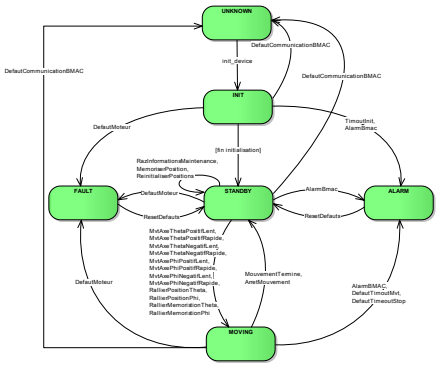


Figure 5: State machine example.

The simplest way to get information from an equipment is to :

- Use the specific method :
  - The method speaks to the equipment,
  - The equipment answers and the method fills up one or several attributes,
- Layer 1 or another DS client reads the attribute value.

This works. But, actually, when more than one client try to access the equipment before the end of each request and if the equipment (or the DS) is not compliant with some design rules, a deadlock can occurs and timeout messages are sent by to the clients (Fig. 6).

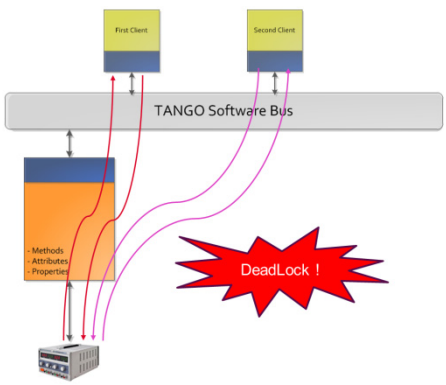


Figure 6: Deadlock.

One way to avoid this situation is to configure the TANGO polling mode.

This specific configuration (which also can be used for computer hardware architecture evaluation) can be applied to each single method.

When activating the polling mode, the DS is, by the way, in charge of calling the method at a configured period and filling up the attributes in a buffer.

Depending on the configuration, when the attribute changes or exceeds a specific range, an event is sent to the clients. Each client gets directly the information from the buffer and not using the method.

If a client tries to call a DS'method, the method is not used, but the reading directly comes from the attribute's buffer (Fig. 7).

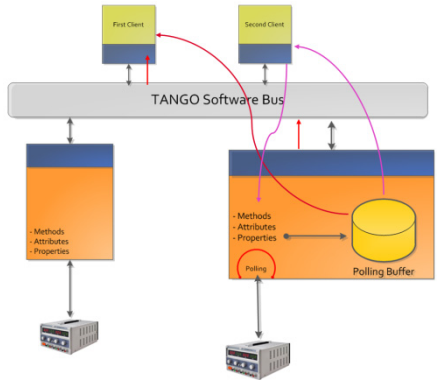


Figure 7: Polling activation.

Using a Modular Architecture for Optical Cameras and Siemens PLCs

Usually, complex instruments (optical cameras, motor controllers with slots) are made of sub-equipments.

The optical camera used in our first target diagnostics uses 5 sub-equipments whose drivers can also be shared with future cameras.

For framing cameras and streak cameras, we developed the following drivers:

- Andor CCD
- Agilent power supply
- Kentech high voltage pulse generator
- One specific electronic board for each camera

No doubts that there will be no needs for Layer 1 to access some very low level methods of sub-equipment. To make this point easier for level 1 developer, we defined the following naming convention[1] :

- Each instance of sub-equipment DS will be called BN_NameOfDriver_xxx
- Each instance of the Layer 1 connected DS will be called *HN_NameOfDriver_yyy*

For framing cameras, BN driver instances will be :

- BN_CCD_001
- BN_Agilent_001
- BN_GXD_001
- BN_ElecCIIX_001

The high level DS connected to Layer 1 will be :

- HN_CIIX_001

Actually, the *HN* driver does the main job as the *BN* driver does the more complex and dirty one!

The following Figure shows the interface organization for framing cameras (Fig. 8).

[1] Where *BN* stands for "Bas Niveau" i.e. "Low Level" and *HN* stands for "Haut Niveau" i.e. "High Level" ; *xxx* and *yyy* are numbers

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

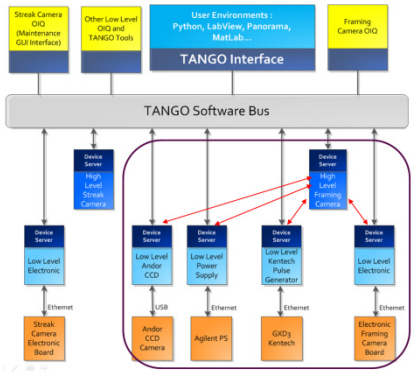


Figure 8: High level and Low level drivers.

Managing PLCs is quite a bit more complicated.

S7-300 Siemens PLC's are used to manage pumps, valves and gauges. These systems are necessary for the hardware security equations implementations which are very closed to the sensors.

Nevertheless, as Layer 1 and Layer 2 need to access to individual components state (in order to forbid non wanted actions), *HN* and *BN* conventions have been used to write the driver in the following way:

- Low level driver for PLC access (controller, manager and look up tables)
- High level access for individual sensors components (valves, gauges, pumps…)

This configuration, even if it allows full access to individual components, warranties that by the use of PLC equations; methods can or cannot be applied. Level 1 should be able to open a valve, but, if pressure equations written in the PLC do not complain, the *HN_Valve* DS method will reject the command and will reply a verbose error (Fig. 9).

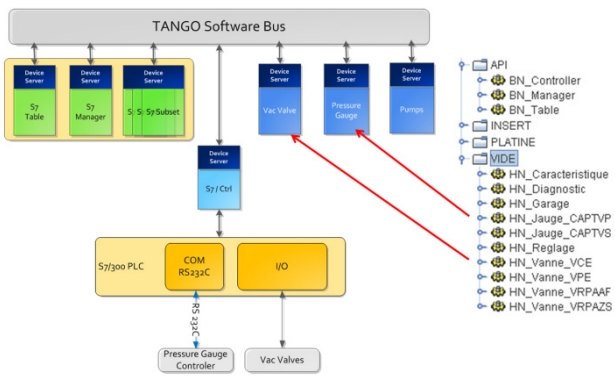


Figure 9: PLC driver configuration.

Maintenance and Qualification GUI Tools

In order to test, control and maintain Layer 0, we will be using Jive (a TANGO native tool for DS implementation usage) and some specific GUI developed in QT/Taurus TANGO framework. Main screen gives access to the Layer 0 TDCS and configuration menus driven by *HN* DS. These tools are used for both qualification and maintenance in locations where Layer 1 is not available.

A very low level interface let the maintenance operator analyze specific configurations like motors interfaces or optical cameras. In LMJ, these actions are granted by Layer 1 (Fig. 10).

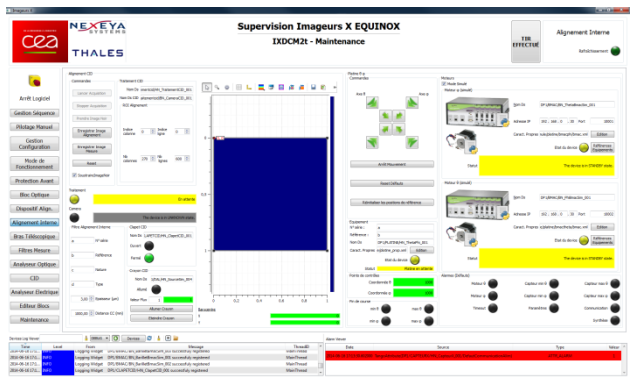


Figure 10: Maintenance GUI Tool.

Real and Virtual Modes

Every command control software is test in our ICSS integration platform [8]. Each subsystem uses specific software simulators that reproduce the equipment responses to methods.

For target diagnostics, we developed two DS *BN* modes by using the same Pogo architecture:

- *Real mode* is directly connected to the equipment,
- *Virtual mode* gives responses from specific files.

Two properties for the *HN* DS are used :

- A Boolean that tags mode type,
- A property that gives the name of the *BN* driver mode.

In both cases, the interface between *HN* Layer 0 and Layer 1 remains the same.

This configuration gives also the developer, the ability to test a full TDCS without the need of all real equipments. For each virtual mode, XML files deliver data and for some specific equipments, like motor controllers, an external software can generate defaults states. Actually, each *BN* DS exists in both real and virtual mode but only the real mode is loaded inside the facility command control.

For PLCs, the virtual mode is located inside the controller. This software emulates PLC's cards at the low level interface. There is no change for *HN* DS but the configuration.

LAYER 1

PANORAMA for Layer 1

Layer 1 for TDCS is developed using the French SCADA PANORAMA E² [2]. This LMJ requirement is necessary to insure all subsystems interfaces and protocol access to Layers 2 and 3.

PANORAMA comes with a generic graphical and VbScript editor. A generic framework and libraries

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

written for LMJ are delivered for developments but, as PANORAMA is mostly appropriate for automation, there was a need to interface this high level SCADA to a low level instrumentation architecture like TANGO.

Codra developed this new TANGO to PANORAMA interface (*binding*) for CEA. This interface creates a bi-directional link between each TANGO DS object and PANORAMA objects. In that way, a PANORAMA software developer does not have to know anything on the low level TANGO architecture (Fig. 11).

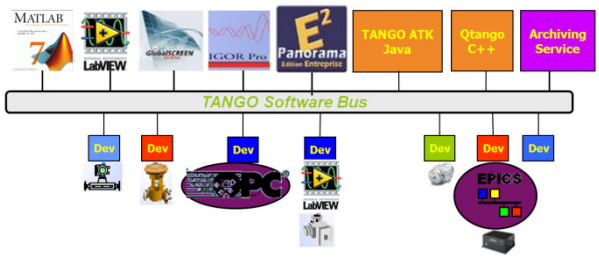


Figure 11: PANORAMA Binding.

A specific editor manages target diagnostics TANGO DS libraries and scripts written in VbScript to respond to Layer 2 sequences orders.

This binding is open source and available for free at www.TANGO-controls.org [3].

By using TANGO for layer 0, PANORAMA for layer 1 and this new binding between these 2 layers, CEA has now the 3 main tools for developing LMJ TDCS GUI.

A generic GUI is devoted to exchange with top layers. In that way, future target diagnostic Layer 1 GUIs will be like new plugins for the main interface (Fig. 12).

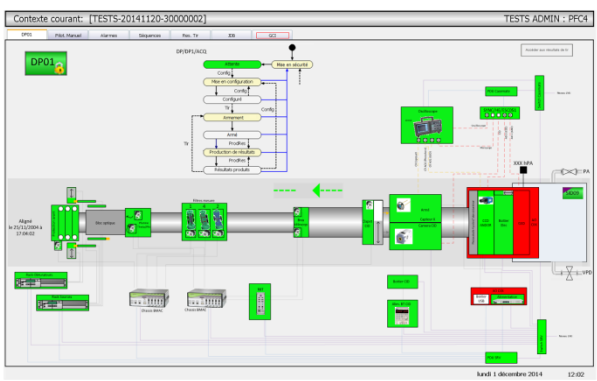


Figure 12: XRay Layer 1 interface.

MANAGING CONTRACTORS

Each part of a target diagnostic has been developed by an external contractor upon CEA's requirements. In order reduce interface problems, we chose to give full responsibility for the hardware contractor to take in charge DS developments, based on the full knowledge of his materials.

If this worked in some cases, it was not sufficient in others, due to the misunderstanding of TANGO architecture (which was clearly new challenge for some industrials) and the knowledge of interfaces between sub systems of a target diagnostic.

We wrote a "TANGO design rules guide" for new developers which is, by now, used in the community but was quite fresh during the beginning of our project.

Managing contractors was mostly a race between requirements, interfaces and time scheduling.

The best advice should be to keep a developing team that works on a low level DS skeleton, quite functional, and make it fully industrialized once tests are sufficient. Externalizing process should only be efficient at this very moment.

SUMMARY

This paper describes the TDCS architecture based on TANGO for the low level equipment driving, PANORAMA for the high level SCADA and the new TANGO to PANORAMA binding that makes the bidirectional glue between both layers.

The hardware to software abstraction layer insures the installation to keep software interfaces while equipment upgrade. TANGO makes it.

Layer 1 architecture uses a common interface which allows future target diagnostics software to be plugged in.

All these specifications have been chosen to guaranty modularity and life time expectancy for :

- adding new target diagnostics
- managing heterogeneity
- using new kind of equipment for target diagnostics
- helping upgrade computers and equipments

Reference [9] gives a full explanation on the LMJ facility control system status report.

REFERENCES

[1] www-lmj.cea.fr
[2] uk.codra.net/panorama
[3] www.tango-controls.org/download/binding
[4] MOC3O06 : PARC : How to computerise Laser Setting on the Megajoule Facility, ICALEPCS 2015, by S. Vermersch, CEA/CESTA, Le Barp, 33114 France.
[5] MOPGF075 : PARC : An Automated Laser Setup System for the Laser Megajoule Facility, ICALEPCS 2015, by J.P. Airiau, CEA/CESTA, Le Barp, 33114 France.
[6] TUB3O04 : The LMJ System Sequences Adaptability, ICALEPCS 2015, by Y. Tranquille-Marques CEA/CESTA, Le Barp, 33114 France.
[7] THCOCA05 : Laser Megajoule Timing System, ICALEPCS 2013, by P. Raybaut, V. Drouet, J.J. Dupas, J. Nicoloso, CEA/DIF, Bruyères le Châtel, 91680 France.
[8] MOCOBAB03 : The Laser Megajoule ICCS Integration Plateforme, ICALEPCS 2013, by J.P. Arnoul, J. Fleury, A. Mugnier, CEA/CESTA, Le Barp, 33114 France.
[9] FRA3O02, The Laser Megajoule Facility Control System Status Report, ICALEPCS 2015, by J. Nicoloso, CEA/DIF, Bruyères le Châtel, 91680 France.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

EXTREME LIGHT INFRASTRUCTURE, BEAMLINES – CONTROL SYSTEM ARCHITECTURE FOR THE L1 LASER*

J. Naylon#, T. Mazanec, Institute of Physics, Prague, Czech Republic, A. Greer, C. Mayer, Observatory Sciences Ltd., Cambridge, UK, M. Horáček, B. Himmel, M. Drouin, K. Kasl, J. Horáček, P. Škoda, P. Bakule and B. Rus, Institute of Physics, Prague, Czech Republic

Abstract

The ELI-Beamlines facility aims to provide a selection of high-energy and high repetition-rate TW-PW femtosecond lasers driving high intensity XUV/X-ray and accelerated particle secondary sources for applications in materials, medical, nuclear and high-field physics sectors. The highest repetition rate laser in the facility will be the L1 laser, producing 1 kHz, 20 fs laser pulses of 200 mJ energy. This laser is based entirely on picosecond chirped-pulse parametric amplification and solid-state pump lasers. The high repetition rate combined with kW pump powers and advanced technologies calls for a highly automated, reliable and flexible control system. Current progress on the L1 control system is discussed, focussing on the architecture, software and hardware choices. Special attention is given to the LabVIEW-EPICS framework that was developed for the ELI Beamlines lasers. This framework offers comprehensive and scalable EPICS integration while allowing the full range of LabVIEW real-time and FPGA embedded targets to be leveraged in order to provide adaptable, high-performance control and rapid development.

INTRODUCTION

The four laser sources (L1-L4) are currently under development and will be installed in the facility in time for a 'first light' demonstration in 2016-17. The L1 laser's high repetition rate, 5 TW peak power and excellent beam quality is aimed at experiments based on high-harmonic generation (XUV), X-ray and keV betatron radiation in the molecular, biomedical and materials sciences [1].

The resulting L1 control system architecture and design is currently in a fairly final state with most hardware and software components operational and tested as part of an integrated solution. The aim of this paper is to summarise the chosen architecture and highlight some of its advantages for similar control system projects.

ARCHITECTURE DEVELOPMENT

At the start of control system development, the type and quantity of I/O was estimated by reference to similar laser systems. Requirements were also collected regularly from the laser scientists. The estimate of the scale of the I/O requirements was the basis of the architecture:

Table 1: L1 Control System Scale

Device	Parameters	Approx. qty
Cameras	20 Hz, 2 M pixel	60
High speed cameras	1 kHz 0.5 M pixel	6
Stepper motors	Simple, open loop	100
Piezo actuators	Small, open loop	20
Piezo steppers	Mostly open loop	20
Laser relative energy	*e.g.,* photodiode, kHz	40
Digital I/O channels	Mostly 24 V logic	30
General analogue I/O	≤kHz, 16 or 24-bit	30
Temperature sensors	RTDs	20
Flow sensors	Cooling water, pulse	20
Complex instruments	Serial, USB, Ethernet	6
Simple devices	Serial, Modbus, *etc.*	15

Over the course of development other project-specific factors have also been significant in determining the final choice of implementation:

- The high laser repetition rate and expense of key laser optics necessitates a real-time control system with sufficient intelligence to provide machine protection;
- Some laser instruments are highly specific and available from few vendors. Generally the only integration for these is via LabVIEW;
- The laser is based on new and cutting-edge technology, so control system requirements change frequently;
- The budget, time and effort required for laser control systems are often underestimated;
- Human resources, particularly software developers, are limited due in part to competition from the rapidly growing IT sector locally;
- Being a new facility there is little established experience. Any solution must be easy and quick for new staff to learn and work with;
- Strict tendering rules and laws restrict procurement; therefore reliance on a few key suppliers for most of the control system components justifies the time and expense of preparing a framework contract.

EPICS was chosen as the integration framework as it is widely used, stable and requires minimal programming. Use of LabVIEW was essential and is fast, flexible and easy to learn. This was chosen to be the sole software development language for reasons of maintainability and to streamline training. The resulting structure of the control system (Fig. 1) is based on a simple 3-layer scheme appropriate to the scale and suitable for an industrial/machine-control system.

* Work supported by the European Regional Development Fund and the European Social Fund under Operational Programs ECOP and RDIOP. The project is hosted by Fyzikální ústav AVČR, v.v.i.
#jack.naylon@eli-beams.eu

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The services layer provides database-driven configuration and data archival. A flexible HMI server supports flexible configurations of in-lab displays and centralised status panels. A hierarchical state machine model governed by a top-level sequencer ensures subsystem modularity and facilitates integration and automation. Finally, the interface gateway provides network security and integration to the rest of ELI-BL.

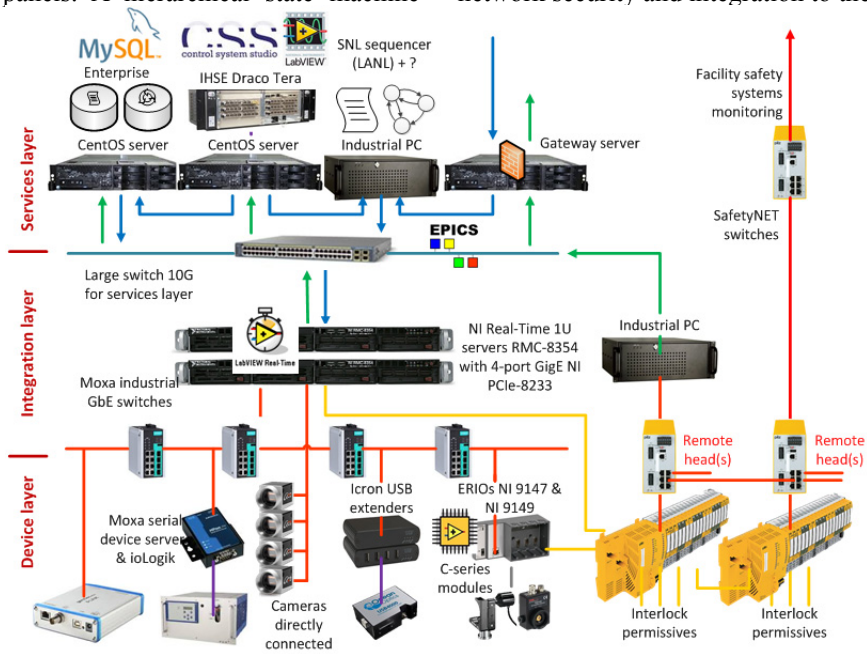


Figure 1: Control system architecture for the L1 laser – implementation view.

ARCHITECTURE IMPLEMENTATION

Selection of hardware and software solutions to implement this control architecture is now largely finalised. Advice from National Instruments and from Lawrence Livermore National Laboratory (contractors for the L3 laser at ELI-BL) was instrumental in selecting an implementation. Ongoing collaboration with LLNL on the High-Repetition-Rate Advanced Petawatt Laser System (HAPLS) [2] and with National Energetics-EKSPLA on the 10 PW L4 laser [3] has resulted in the standardisation of much of this control system architecture across all of the laser beamlines – a major achievement for integration and assurance of the future maintainability of the facility.

Due to the large number of cameras the NI RMC-8354 controller was chosen. This product is normally marketed as a remote PXIe chassis master. It is based on a 1U server-class computer with NI's Pharlap-based operating system. The one x16 PCIe slot is suitable for mounting a 4-port frame grabber (NI PCIe-8233), and the powerful platform is useful for image processing at the 20 Hz framerate. Each IOC has a dedicated link to the MSS via its serial port; modem lines are used as a deterministic digital interface to signalise faults and trigger a safe-state. The RMC-8354 has limited I/O options; therefore the device layer was restricted to Ethernet with the use of Ethernet-to-serial (Moxa NPort) and USB-over-Ethernet (Icron RG2304GE-LAN) interface adapters. This has the advantage that IOCs can be largely hardware-independent, supporting virtualisation.

The machine safety system (MSS) and personnel safety system (PSS) are controlled by a SIL-3 rated (IEC 61508) PLCs (Pilz PSS 4000) offering reliability and flexibility at reasonable cost. Although SIL-3 is not required for machine safety, use of the same platform for both systems simplifies training and improves maintainability.

The MSS is interfaced via Modbus to EPICS integration layer using Base records on Linux. Standard records are used to interface most simple support devices such as chillers and PSUs. PSUs with analogue control such as those for Pockels Cells are controlled using simple remote I/O modules (Moxa ioLogik) using Base records. This approach avoids unnecessary LabVIEW development when the application is straightforward.

More complex low-level I/O is handled by NI C-series modules on a programmable FPGA expansion platform (Ethernet RIO). The C-series range is sufficient for all of the analogue and digital I/O requirements of the L1 laser. This solution was chosen as it is more modular, cheaper, and easier to use than higher-density platforms such as MicroTCA or PXIe which are better suited to higher performance I/O and/or higher channel counts. The Artix-7 FPGA is flexible to handle changing laser requirements and powerful enough for sophisticated feedback systems (such as laser power stabilisation, fast beam-pointing correction and temporal jitter cancellation).

NI provides two other RIO expansion systems, MXIe and EtherCAT. An expansion system is appropriate because the RMC-8354 is the real-time host. ERIO is the lowest cost interface for C-series modules. For 'hard-real-time transfer of data at sub-ms rates only EtherCAT RIO and MXIe RIO are suitable. However, trials showed that ERIOs have sufficient determinism to guarantee 1-2 ms transfers over the device layer network with an RMC-8354 host. This is sufficient for all applications in L1.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

SOFTWARE IMPLEMENTATION

A LabVIEW framework was established to support the integration of ERIO controls, instrument drivers and camera image processing. The architecture is simple and robust, well-suited to machine-safety-centric design and real-time software (Fig. 2).

At its heart is a hierarchical state machine model. Each IOC has its own state machine definition to abstract its component processes. Below this, every hardware device driver, feedback algorithm, analysis task, *etc.,* is contained within a process, based on a queued state machine. Messages are strictly produced only by a single central sequencer. When the IOC is sent an external state transition request (via a PV), the sequencer translates this into process messages executed in the appropriate order. Failure of a process causes the sequencer to revert all steps in the transition, returning the IOC to a known state. Laser automation is therefore handled by distributing state transition requests to IOCs; this can be done without detailed knowledge of all IOC processes.

Processes inter-communicate via notifier references only. All upward data flow is via a hierarchical current value table implemented using variant attributes. The LabVIEW IOC core is designed so that multiple instances can run on the same RMC-8354 server independently. This facilitates the modularisation of software IOCs as a reusable functional unit.

This architecture is simple and fairly inflexible, yet is easy and quick to use. LabVIEW object orientated programming (LVOOP) is avoided in this framework. Much more sophisticated frameworks exist, such as the GSI CS-Framework based on the Actor Framework [4], but these were considered too complex for this application and difficult for novice LabVIEW developers to learn and use. One benefit of inflexibility here is that it forces processes to be well-designed and self-sufficient. Since individual processes are easy to unit-test, this lends greater confidence to IOC integration.

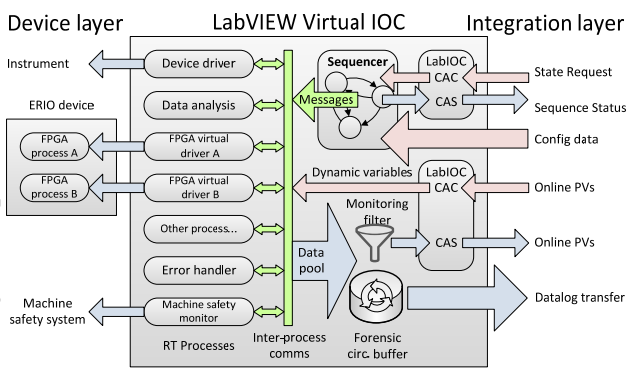


Figure 2: Block diagram of the LabVIEW-based IOC framework.

LABVIEW-NATIVE CHANNEL ACCESS SERVER AND CLIENT - LABIOC

A vital component of the control system is an EPICS channel access client (CAC) and server (CAS) that is compatible with the RMC-8354 platform. It was decided that the only acceptable implementation would be in native LabVIEW to ensure the longevity and ease of integration of the solution, regardless of any later upgrades to the Real-Time OS or server platform.

Many solutions for LabVIEW-EPICS interface have been developed previously [5]. The majority are incompatible with Pharlap and are not LabVIEW-native. Initially, NI's built-in solution was considered. However, this requires Network Shared Variables to be used. These have been found by us to be slow, unstable and have poor scalability and are avoided in our control system. Furthermore, the server only implements a partial record that does not have many important fields. In early testing this caused problems for Control System Studio clients. There were also concerns about the longevity of the solution and that the solution is not open-source.

Due to the critical nature of this package we opted for a commercial open-source solution. Observatory Sciences Ltd. (OSL) [6] were approached to deliver a LabVIEW-native server and client package. This LabIOC package is based on an existing LabVIEW client for EPICS developed by OSL. As well as L1, the package will be used in the other laser systems, particularly L4, making it key to the successful integration of the ELI-BL's lasers.

The package was released in May 2015 and is now undergoing comprehensive testing and bug fixes in collaboration with National Energetics. The package implements a LabVIEW-native CAC and CAS and supports standard record types (ai/ao, bi/bo, longin/longout, mbbi/mbbo, stringin/stringout, and waveform). On the LabVIEW side, automatic type conversion to I8, I16, I32, SGL, DBL, STR, Boolean, and arrays of these types is handled automatically via convenient polymorphic VIs.

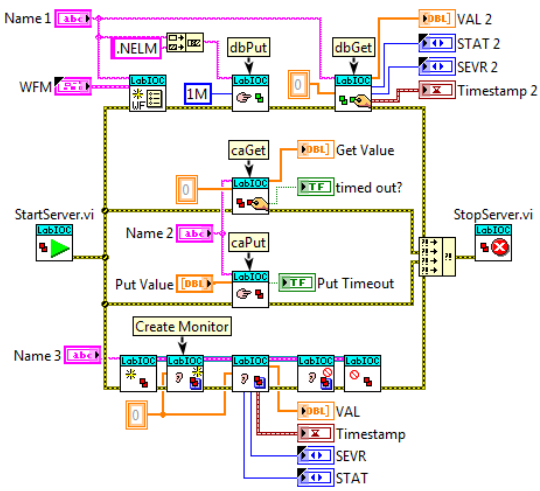


Figure 3: Example of the LabIOC LabVIEW interface.

A major advantage of the package is that multiple software can IOCs run in parallel. Instances share a single UDP port on the NI run-time engine by setting the flag "SocketSetReuseAddr=TRUE" to the LabVIEW .ini file options. However, this feature has the consequence of

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

making the library take up rather a lot of memory on the server – possibly limiting its use on smaller real-time targets such as cRIOs. We are working with OSL to streamline and optimise the package.

LABIOC PERFORMANCE ANALYSIS

To evaluate the performance of the LabIOC package, two RMC-8354 servers were connected by a 1 GbE Moxa switch. Identical versions of LabIOC were run on both and a variety of measurements performed. Access times were analysed (Fig. 4) to verify the determinism of the software on the real-time system.

The analysis shows that server-local variable access is quick and fairly deterministic, as expected. Access time decreases exponentially and appears bounded at no more than three times the median vales of 60 and 330 microseconds for Get and Put, respectively.

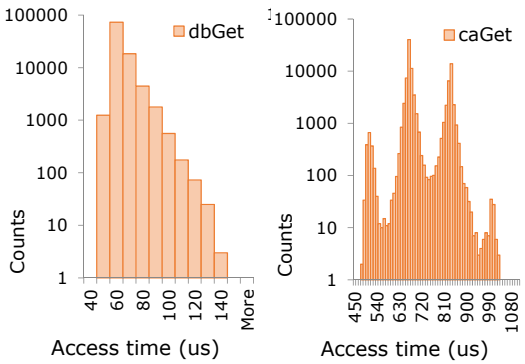


Figure 4: Server and client access time histograms for a single, local PV of DBL type (ai record).

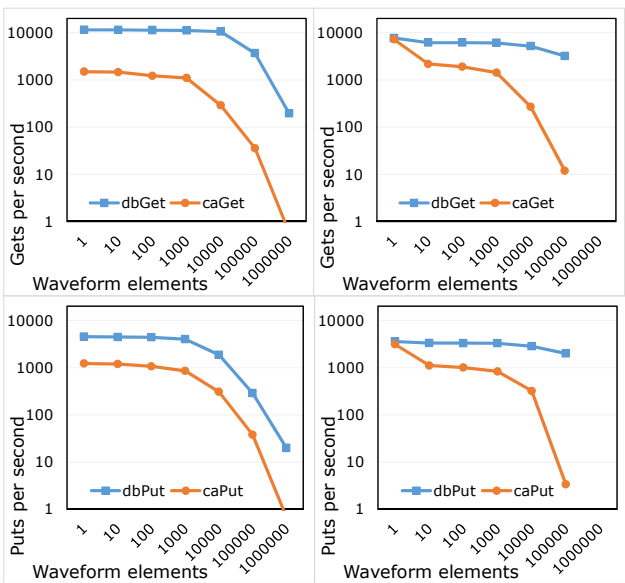


Figure 5: Server and client access frequencies. Top right: Gets per second for a single waveform; top left: Gets per second for 1000 channels of waveforms; bottom left Puts per second for a single waveform; bottom right Puts per second for 1000 channels of waveforms. All PVs are DBL/ai type.

Measurements of maximum read/write frequencies (Fig. 5) show that LabIOC is slightly slower than typical figures for CA (~2k caGets/sec for single element). Results for single-channel and 1000-channel are similar below 1M elements – showing library to be robust and scalable.

Generally these measurements show that the overhead associated with the LabIOC library is reasonable and meets the requirements for the ELI-BL lasers. Tests are ongoing.

CONCLUSION

Control system requirements, architecture and implementation for the L1 beamline have been discussed. Central to the control system was the development of a LabVIEW-native EPICS server on the Real-Time RMC-8354. The package has been tested and found to be a satisfactory solution for the integration challenge.

Future development will focus on integration of LabIOC into the software framework. This will provide integration of the hierarchical state machine within each IOC with the top-level sequence engine to enable laser automation. Laser diagnostics can also be displayed via CSS panels, archived, and plotted using existing tools – removing a significant software development burden.

Currently, the main drawback of the solution is that it does not support the transfer of camera image data with the required frequency and fidelity (smaller 'preview' images can still be transferred). Laser beamlines have highly camera-focussed control systems and a LabVIEW-EPICS integrated solution for high-performance camera image transfer would be a welcome future development.

REFERENCES

[1] B. Rus et al. "ELI-Beamlines laser systems: status and design options," Proc. SPIE 8780, *High-Power, High-Energy, and High-Intensity Laser Technology; and Research Using Extreme Light: Entering New Frontiers with Petawatt-Class Lasers,* 87801T (May 7, 2013); doi:10.1117/12.2021264.

[2] LLNL Science and Technology Review, "Lighting a New Era of Scientific Discovery" [Online]. Available: str.llnl.gov/january-2014/haefner [Accessed 2015]

[3] News Feed, "National Energetics and EKSPLA awarded contract to build 10 Petawatt laser system" [Online]. Available: http://bit.ly/1NCg6uJ [Accessed 2015]

[4] GSI Wiki, "CS - A Control System Framework for Experiments" [Online]. Available: http://bit.ly/1ZwNU00 [Accessed 2015]

[5] A. Zhukov; "Pure LabVIEW Implementation of EPICS Communication Protocol", NI Week 2012, [Online]. Available: decibel.ni.com/content/docs/DOC-28575 [Accessed 2015].

[6] Observatory Sciences Ltd., Cambridge, UK [Online]. Available: http://www.observatorysciences.co.uk/ [Accessed 2015].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REMUS: THE NEW CERN RADIATION AND ENVIRONMENT MONITORING UNIFIED SUPERVISION

Adrien Ledeul, Gustavo Segura Millan,
Riku-Pekka Ilari Silvola, Bartlomiej Styczen, Daniel Vasques Ribeira,
CERN, Geneva, Switzerland

Abstract

The CERN Health, Safety and Environment Unit is mandated to provide a Radiation and Environment Monitoring SCADA system for all CERN accelerators, experiments as well as the environment. In order to face the increasing demand of radiation protection and continuously assess both the conventional and the radiological impact on the environment, CERN is developing and progressively deploying its new supervisory system, called REMUS - Radiation and Environment Monitoring Unified Supervision.

This new WinCC OA based system aims for an optimum flexibility and scalability, based on the experience acquired during the development and operation of the previous CERN radiation and environment supervisory systems (RAMSES and ARCON). REMUS will interface with more than 70 device types, providing about 3,000 measurement channels (approximately 500,000 tags) by end 2016.

This paper describes the architecture of the system, as well as the innovative design that was adopted in order to face the challenges of heterogeneous equipment interfacing, diversity of end users and non-stop operation.

INTRODUCTION

Radiation Protection and Environment Monitoring of CERN facilities and experiments are essential for three reasons:

- Safety of the workplace and of the environment.
- Reporting to authorities the nature and the quantities of emitted ionizing radiation.
- Reporting to authorities in case of pollution of the environment.

In order to fulfil those missions, CERN has set up diverse monitoring equipment across CERN area and its immediate surroundings.

Since 2005, RAMSES (RAdiation Monitoring System for the Environment and Safety) [1] has been in charge of the data acquisition, control/command and supervision of 50 device types, allowing the supervision of 1,500 channels. The former radiation monitoring system, ARCON (ARea CONtroller), is also still in charge of the majority of the monitoring in PS (Proton Synchrotron) accelerator complex. Additionally, proprietary supervisors such as Berthold MEVIS handle the data acquisition and supervision of devices that have not been integrated into RAMSES Supervision. Numerous devices are also operated without any remote supervision.

In 2012, the development of a new SCADA system, REMUS (Radiation and Environment Monitoring Unified Supervision) began. This system aims to:

- Cover a larger set of equipment than RAMSES, in order to unify the supervision and integrate stand-alone devices, most of them being COTS (Commercial Off-The-Shelf) products.
- Provide a reliable, scalable and cost-effective system.
- Use common CERN software, WinCC OA (WinCC Open Architecture) [2], formerly known as PVSS, and JCOP framework (Joint COntrols Project) [3], for the development and support of the new SCADA.
- Reduce the delay and the cost of adding new devices to the supervision.
- Provide light and fast clients, adapted to the needs of diverse end-users.
- Reduce the overall maintenance and user support effort necessary to maintain the system in operation.

REMUS founded its evaluation of scalability and performance on the study performed by CERN engineering department [4]. The probed technologies are commonly employed in large SCADA systems supervising LHC (Large Hadron Collider) experiments.

For its development, REMUS applies the same International Standard IEC 61508 [5] that was set up during the development of RAMSES.

REQUIREMENTS AND SCOPE

The REMUS project aims at developing a universal software for supervision, control and data acquisition for the entire suite of monitoring stations covering all radiological and environmental parameters that can potentially be affected by the operation of the facilities of CERN.

The functions of the system include:

- Logging of all measured values and system events coming from the instrumentation.
- Providing real-time measured values, alarms and operational states of the devices through customizable user interfaces composed of synoptic and widgets.
- Publishing alarms and faults of the devices to a remotely accessible user interface and to the CERN central alarm system, LASER (Lhc Alarm SERvice) [6].
- Publishing a selected set of measurements to other systems via the CERN Data Exchange system, DIP (Data Interchange Protocol), built on the top of DIM (Distributed Information Management) [7].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- Logging of a selected set of measurements to CERN shared MDB (Measurement DataBase) [8].
- Remote sending of commands and operational parameters, while recording their changes.

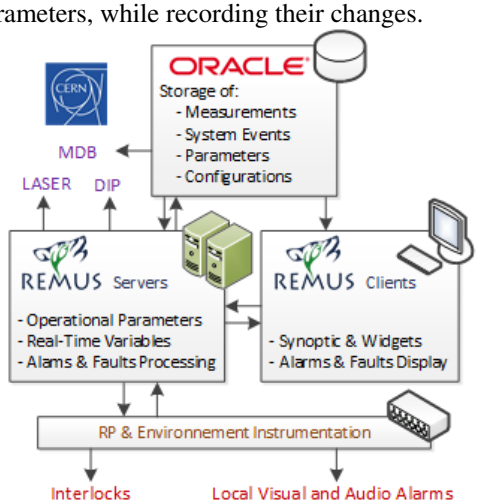


Figure 1: REMUS Functional Diagram.

The system shall monitor the instrumentation 365 days a year, 24/7 and ensure the logging of measured values and system events in case of network outages.

Local safety functions, such as interlocks and visual and audio alarm devices are kept to a low level, independent of the supervision.

HANDLING HETEROGENEOUS DEVICE TYPES INTEGRATION

Devices connected to REMUS come from various manufacturers. Uniform protocols and architectures can therefore not be reasonably expected from the 70 device types that need to be interfaced. Furthermore, many of those devices were not yet selected from the market at the time the development of REMUS began.

Model and Basic Concepts

One approach would be to encapsulate those diverse protocols into heavy middleware infrastructures, but this could lead to a degradation of the performance and has the disadvantage of adding complexity to the architecture, therefore increasing the maintenance cost. In addition, it creates dependencies to the infrastructure and makes the full system harder to be deployed in a different context, for example in a different laboratory for the supervision of temporary experiments.

The innovative approach chosen by REMUS makes an abstraction of this complexity by using a generic model, applicable to any kind of instrumentation, in order to provide the SCADA a uniform set of instruments to monitor (See Fig. 2).

REMUS proposes 3 basic concepts:

- Channel: An abstract entity representing one and only one point of measurement at a specific location, connected to a device.
- Device: A piece of equipment of a determined type (Electronic boards, PLCs, Ionization chambers …) holding its own parameters, eventually connected to other devices and channels.
- Monitoring Station (MS): An encapsulation of a set of devices and associated channels, connected to the system through a driver.

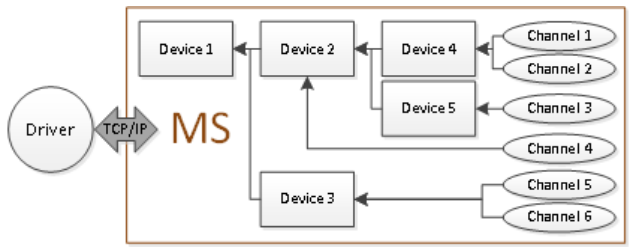


Figure 2: REMUS Instrumentation Model.

Each device type to interface is then modeled in an external Oracle Database using those three concepts, providing a uniform set of equipment to the system. This approach is flexible enough accommodate any radiation and environment monitoring device in the market. It homogenizes the software, makes most of source code reusable and simplifies the development process, from analysis to tests and validation.

Run-time Declaration of New Devices

Using the model previously described, declaring a new device of a known device type in the supervision is equivalent to creating a new instance of the model previously defined, and associating it to a driver. This process is done directly through a WinCC OA User Interface. The new instance is created in the REMUS Oracle Database that REMUS SCADA uses as a reference for the creation of its internal variables, which will hold all the necessary data coming from the equipment in real time. This entire process of instantiation can be executed during run-time and does not requires any restarting of the REMUS servers. Run-time declaration, creation and commissioning of a new device is a function of the system, tested and validated as such, allowing REMUS to be maintained in normal operation 365 days a year, 24/7.

This mechanism, associated with an adapted access control policy, allows users to declare new devices of a known device type in the system, without any intervention of the REMUS maintenance team, reducing the user support effort significantly.

DIVERSITY OF USERS

One of the particularities of the system is its diversity of users. Indeed, REMUS shall provide interfaces for accelerator operators of the different CERN accelerators and experiments, radiation protection engineers, environmental engineers, physicists & maintenance teams.

Furthermore, instrumentation is frequently moved, dismantled or installed, due to the nature of CERN and its

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

large scope of experiments. Thus, user interfaces need to be adapted to each category of user and flexible enough to allow run-time evolution.

REMUS presents to the users a tree-structured set of synoptic representing CERN surface and underground areas. Widgets representing monitoring stations and channels can be displayed on the synoptic. Several versions of widgets are available for each device type in order to display the appropriate level of information to the final users.

REMUS Applications

In order to provide each user the most suitable interface, REMUS allows the definition of several sets of synoptic and widgets, called *Applications*. Applications represent a sub-set of the supervised instrumentation, with a specific access control. Advanced users having the appropriate access rights can create their own applications for the needs of their team, choosing the synoptic, channels and monitoring stations to be displayed.

REMUS applications are stored in xml files, editable through a user friendly interface, the *Application Editor*, developed by the REMUS team. This feature, added to the run-time definition of new devices, allows advanced users to add a new device to the supervision and make it accessible to the appropriate users without having to go through a heavy procedure, and without intervention of the REMUS maintenance team.

The outcomes of this strategy are positive in three ways:

- Users can focus on the devices they are interested in (only a sub-set of the total instrumentation). For example, some of the applications only display the monitoring of a specific accelerator or experimental area.
- The performance of the clients is improved, as only the variables displayed in the current application are fetched from the servers.
- The maintenance effort is separated among several *Application Administrators*, close to the needs of the users, who are not manpower of the REMUS maintenance team.

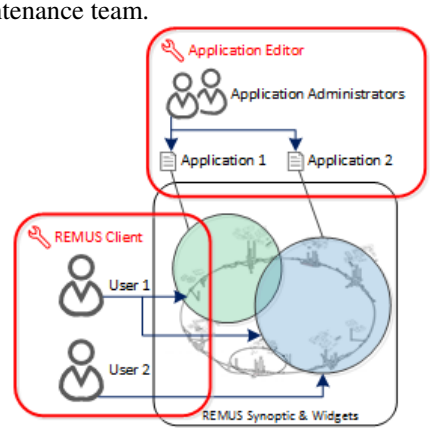


Figure 3: REMUS Application definition and usage.

ARCHITECTURE

As a safety system used by CERN not only during accelerators operation, but also during shutdowns, REMUS needs to be as resilient as possible. The architecture put in place (See Fig. 4) allows the continuous availability of the SCADA, during maintenance operations, deployments, or in case of failure of one of its components.

Connection of the Devices to WinCC OA

REMUS implements five ways to connect devices to the SCADA, depending on the protocol implemented on each device type:

- WinCC OA native driver
- JCOP driver
- In-house developed driver on top of WinCC OA API
- File exchange system
- OPC Server + OPC Client

The choice of driver/middleware type depends on the capacity of the device, as well as the cost of development and maintenance.

Data Acquisition & Archiving

The devices connected to REMUS provide about 170 measurements per second (700 at project completion). For performance reasons, the approach that was chosen is to generate homogeneously formatted measurement files, at the lowest possible level (Device, Driver or WinCC OA control managers), and then inject them to the REMUS Oracle database through SQL*Loader.

This mechanism allows devices and intermediate file servers to store the measurement data in case of network outages in order to ensure the continuity of the data logging.

REMUS Clients

Since WinCC OA has the advantage of being Windows and Linux compatible, REMUS proposes 2 types of clients (Remote UI in WinCC OA terminology) for the users. A load balanced cluster of terminal servers running under Windows Server, accessible from CERN network through remote desktop connections offers a flexible and quick access solution. In addition, Control Rooms can easily access REMUS through their consoles, running under Scientific Linux and already configured to run all the WinCC OA applications developed at CERN.

Redundancy

All the servers used in REMUS architecture are redundant in order to ensure the continuity of the service. WinCC OA integrates a hot standby redundancy feature, including a configurable switching mechanism based on error status evaluation. The less degraded server will be selected as the active server in case of failure of one of the WinCC OA components.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

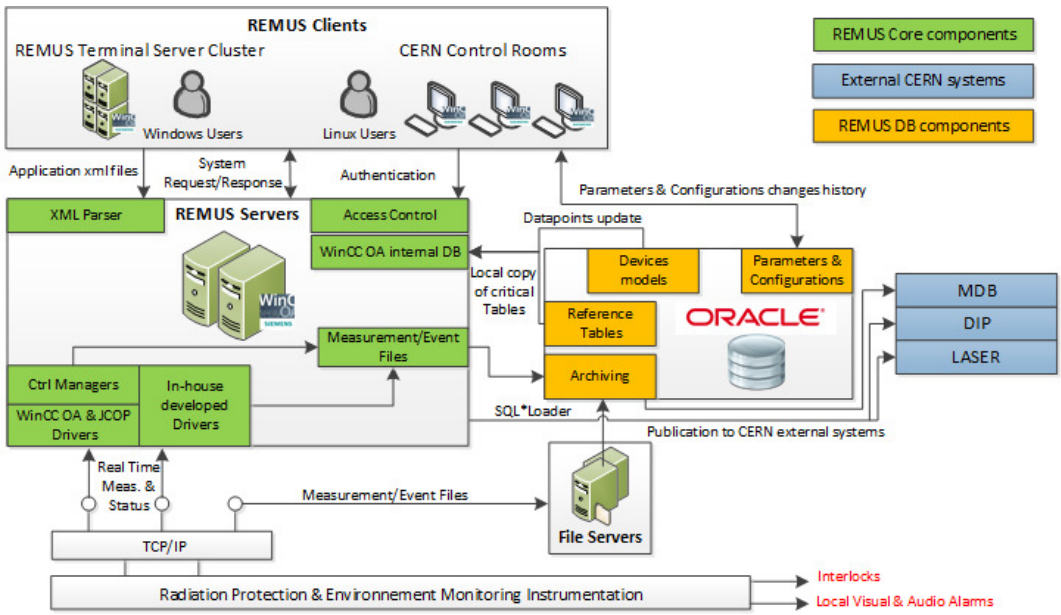


Figure 4: REMUS Architecture.

Oracle Database Failure Handling

Aside from the archiving of measurements and system events, REMUS uses an external Oracle database in order to store device models and history of device configuration and parameterization changes. It also uses Oracle tables as a reference for the list of available measurement units, email addresses, phone numbers to send notifications to etc.

In order to provide the full SCADA functionality in case of unavailability of the Oracle database, REMUS stores critical information into the WinCC OA internal database, based on RAIMA technology [9]. In addition, all statements that should have been executed during this unavailability, like the changes of a device parameters, are stored temporarily in the same internal database and executed when the Oracle database connection is re-established.

CONCLUSION

Relying on technologies widely used and probed across CERN, REMUS achieves its main objective to unify all Radiation Protection and Environment Monitoring systems into one reliable, scalable, flexible and cost-effective solution.

Implementing innovative concepts that handle issues revealed during the development and operation of previous supervisory systems used in the CERN Health, Safety and Environment Unit, REMUS offers a long term prospect.

In continuous operation and development since 2013, the project has received very positive feedback from its users. Its light architecture and easy deployment have opened the possibility of collaboration with other laboratories and industrial partners, who have been interested in the project since its early stage.

ACKNOWLEDGMENT

We would like to thank all the members, past and present, of the REMUS team as well as colleagues from Radiation Protection, Environment, Engineering and Control departments for their fundamental contribution to the success of the REMUS project.

REFERENCES

[1] G. Segura Millan, D. Perrin, L. Scibile, “RAMSES: The LHC Radiation Monitoring System for the Environment and Safety”, ICALEPCS 2005, Geneva, Switzerland.
[2] ETM.at website: http:// http://www.etm.at
[3] O. Holme, M. Gonzalez Berges, P. Golonka, S. Schmeling, “The JCOP Framework”, ICALEPCS 2005, Geneva, Switzerland.
[4] P.C.Burkimsher, “Scaling up PVSS”, ICALEPCS 2005, Geneva, Switzerland.
[5] International Standard IEC 61508 “Functional safety of electrical / electronic / programmable electronic safety-related systems” (E/E/PE) system.
[6] F. Calderini, B. Pawlowski, N. Stapley, M.W. Tyrrell, “Moving towards a common alarm service for the LHC era”, ICALEPCS 2003, Gyeongju, Korea.
[7] C. Gaspar, M. Dönszelmann, Ph. Charpentier ,“DIM, a portable, light weight package for information publishing, data transfer and inter-process communication” Computer Physics Communications, vol. 140, pp. 102– 109, 2001.
[8] C. Roderick, R. Billen, M. Gourber-Pace, N. Hoibian, M. Peryt, “The CERN accelerator Measurement Database: On the road to federation”, ICALEPCS 2011, Grenoble, France.
[9] RAIMA.com website: http://raima.com

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

THE VIRTUAL EUROPEAN XFEL ACCELERATOR

R. Kammering, W. Decking, L. Froehlich, O. Hensler, T. Limberg, S. Meykopff, K. Rehlich, V. Rybnikov, J. Wilgen, T. Wilksen, DESY, Hamburg, Germany

Abstract

The ambitious commissioning plans for the European XFEL require that many of the high-level controls are ready from the beginning. The idea arose to create a virtual environment to carry out such developments and tests in advance, to test interfaces, software in general and the visualisation of the variety of components. Based on the experiences and on the systems that are already in operation at the FLASH facility for several years, such a virtual environment is being created. The system can already simulate most of the key components of the upcoming accelerator. Core of the system is an event synchronized data acquisition system (DAQ). The interfaces of the DAQ system towards the device level, as well as to the high-level side is utilising the same software stack as the production system does. Thus, the software can be developed and used interchangeably between the virtual and the real machine. This allows to test concepts, interfaces and identify problems and errors at an early stage. In this paper the opportunities arising from the operation of such a virtual machine will be presented. The limits in terms of the resulting complexity and physical relationships will also be shown.

THE IDEA

Lessons learned from the fast successful start up of the Linac Coherent Light Source (LCLS) at the SLAC National Accelerator Laboratory showed that one needs to have not only all hardware related software ready and checked from the first hours on, but also all foreseen high level software.

Since most high level software is not directly acting on the hardware layer itself but needs a vast amount of infrastructure to be working properly, testing high level software can only be done with this infrastructure up and running. However, having the high level software ready and tested prior to having the first beam is in contradiction. To still be able to accomplish this task, the idea of having a test environment for this, came up within our group which is in charge of creating high level controls and applications. For a general overview of the concepts and architecture of the foreseen high level software for the European XFEL see [1].

FROM FLASH TO XFEL

The FLASH facility can be seen as the *small brother* of the European XFEL. On the one hand due to the use of the same hardware technologies, but also in the sense of layout, beam physics and creation of the FEL pulses.

Similarities

FLASH has always been used for prototyping and testing hardware to be later be used at the European XFEL. To name some examples of the major hardware systems, there are the:

- µTCA crate standard
- Timing System
- Machine Protection System

Thus FLASH serves already since several years as the test bed for the interplay between the new hardware systems and the software layers for processing and providing this data to the operators and experts. But the European XFEL has, roughly speaking, ten times the extend compared to the FLASH facility. Table 1 shows a comparison of some hardware and characteristics for both machines.

Table 1: FLASH vs. XFEL

System	FLASH	XFEL
Crates	~ 30 VME, µTCA crates	~ 200 µTCA crates
BPMs	~ 40	~ 460
Data rate to DAQ	< 100 Mbytes/s	>> 100 Mbytes/s
Length	~ 300 m	~ 3000 m

The shear number of devices to be installed in the European XFEL and the thereby resulting data rates require a strong data reduction and pre-processing already on the lower hard- and software layers. But beside the need for data concentration at an early stage, the operation of such a large scale facility as the European XFEL require precise synchronization of many different hardware devices.

Further did lessons learned from the operation of the FLASH facility show that one needs to work more in terms of physical meaningful values instead of reading *raw* data directly from the hardware devices.

Even though many ideas and concepts for high level applications can be lend from the FLASH facilty, one needs to take the increased data rate into account. A central concept to overcome this, is to use the, at the FLASH facility already since years used, data acquisition system.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE DAQ SYSTEM AS THE CORE OF THE CONTROL SYSTEM

The overall structure and concepts of the control system will not be discussed here but can be found at e.g. [2]. For details about the status of the controls of the European XFEL see [3].

The data acquisition system of the European XFEL's control system is of major importance for the foreseen high level applications. The overall architecture and concepts of the DAQ system itself have been discussed in e.g. [4]. To enlighten the central role of the DAQ system within the scope of the virtual XFEL these aspects will be discussed here in more detail.

The big amount of data produced by the front-ends require data reduction before the data can be made available to the higher levels and finally to the display level. But beside the pure need for data reduction also the data from various different sources needs to be synchronized. Further more did lessons learned from the FLASH facility have shown that one needs to think much more in terms of *physics entities* instead of passing the digitized data simply up to the user or operator level.

To accomplish this all data from the various front-ends is pushed up to the DAQ system using multicast transmission. This allows for having multiple instances of the DAQ system without the need to re-send data multiple times over the network.

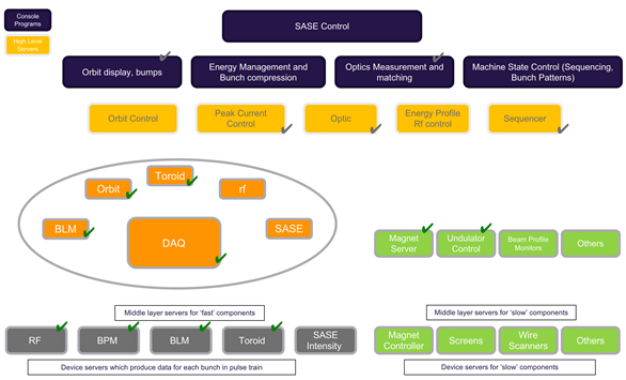


Figure 1: Overview of the XFEL control system architecture (with checkmarks marking *ready* components as to mid 2014).

The overall structure of the control system follows the typical three layered architecture as shown in figure 1.

The DAQ system serves in this sense as the place to attach middle layer processes to do data concentration and pre-processing. As shown in figure 1 are there multiple of these middle layer services attaching to the DAQ system.

THE VIRTUAL XFEL

With having already numerous server processes ready, as depicted in figure 1, we have been able to produce a closed loop through the control system architecture simulating a realistic data flow for the beam position monitors (BPMs) as shown in figure 2.

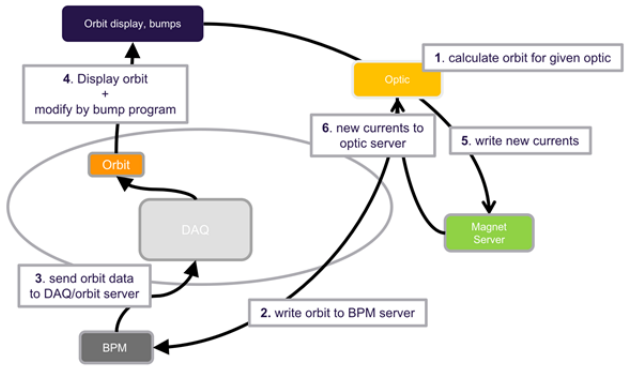


Figure 2: Simulated closed loop data flow for the case of the beam position monitors.

Starting with the test for the BPMs we where able to test not only the network infrastructure but also the processing speed of the attached middle layer servers (in this case the orbit server). After proper configuration and adjustments, a continuous data throughput of roughly 160 Mb/sec had been archived allowing to run with the foreseen nominal repetition rate of the European XFEL of 10 Hz. Figure 3 shows a betatron oscillation within the horizontal plane, introduced by a disturbance (modification of a magnet current) in the injector region of the European XFEL.

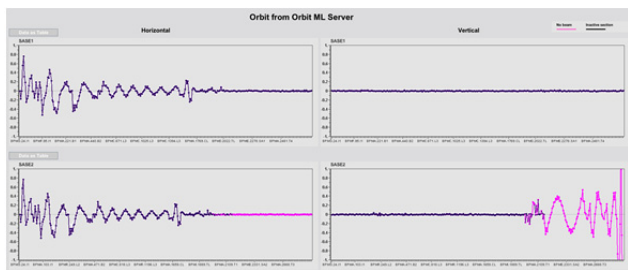


Figure 3: Betatron oscillation introduced within the data flow of the beam position monitors simulating the closed loop.

After accomplishing this first test, which had been the case of the original idea, we started to extend the system step by step by further front-end server processes and corresponding middle layer processes. Figure 4 shows a jddd [5] panel showing all available processes of the virtual XFEL as to date of mid 2015.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

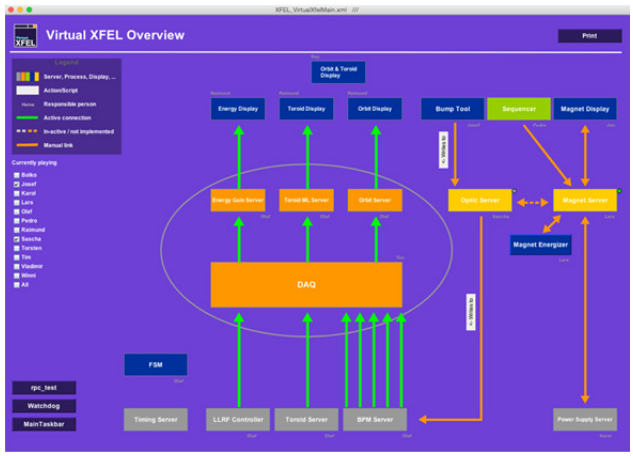


Figure 4: The jddd based control panel of the virtual XFEL (status mid 2015).

Up to date the following hardware systems have been integrated in the simulation of the European XFEL:

- Beam position monitors (BPMs)
- Toroids
- LLRF partial vector sums
- Beam loss monitors (BPMs)
- Magnets

SUMMARY

Starting as a pure test for simulating the data throughput, it soon showed up that the virtual XFEL can be used for much more than this. Especially the test and debugging of all processes in need of synchronized shoot data could be done within this simulation, thus allowing to test handling of complex bunch patterns foreseen to be run at the European XFEL.

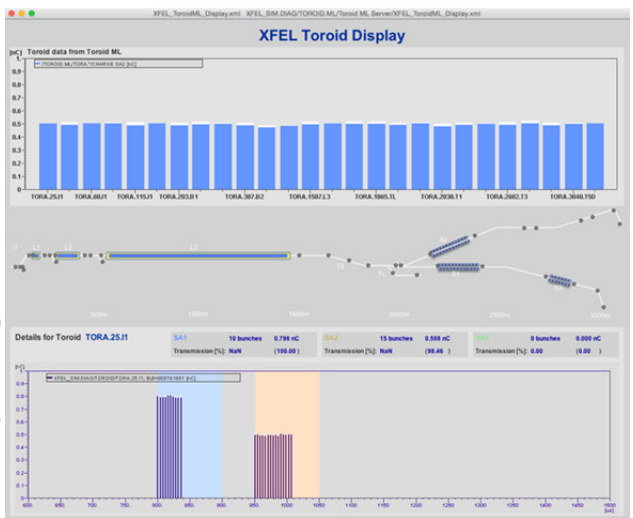


Figure 5: Transmission display developed at the virtual XFEL and nowadays used in standard operation at the FLASH facility.

But even further it showed up that this environment is perfectly suited to do development, testing and debugging of complex graphical user interfaces (see figure 5).

Beside using the virtual XFEL for testing software, configurations created here allow to port the software unchanged from the virtual to the real machine. This is archived due to the reason that the software above the front-end level is completely unchanged and thus only minimal modification of the naming is needed to do the porting.

CONCLUSIONS

The virtual XFEL showed to be of much more use than original intended. Nearly no extra work needed to be done to get this simulation working. Higher level software developed here can be used unchanged later on at the real machine and even configurations can be ported one to one with only minimal modifications.

Thus in contrast to the original idea, it has been decided to keep the hardware for the virtual XFEL dedicated for the simulation and instead build up *a copy* of the system for the real machine.

Working now already for more than a year with this simulated accelerator it showed up that beside all benefits of the virtual XFEL, it also behaves like a real machine. Keeping this virtual machine up and running is also causing some extra work (somewhat it really *feels like a real machine*). Thus a good balance of what is the simulation to be used for and where the limits are being needed here. Thus did it turned out to be in strong contradiction to do basic software developments while someone else is working on optics simulations in parallel.

Nonetheless did the tremendous success of this project showed that setting up an accelerator simulation can sometimes be done without any big extra effort, while providing an enormous profit and thus allowing to be well prepared much ahead of time before the real hardware is in place.

REFERENCES

[1] B. Beutner et al, "High level software for the European XFEL", ICALEPCS'15, Melbourne, Australia, 2015

[2] G. Grygiel, O. Hensler, K. Rehlich, "DOOCS: a Distributed Object Oriented Control System on PC's and Workstations", PCaPAC96, DESY, Hamburg, October 1996

[3] A. Aghababyan et al, "The large scale European XFEL control system: Overview and status of the commissioning", ICALEPCS'15, Melbourne, Australia, 2015

[4] K. Rehlich 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, pp. 256-260

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

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INTEGRATING CONTROL APPLICATIONS INTO DIFFERENT CONTROL SYSTEMS*

M. Killenberg†, M. Hierholzer, C. Schmidt, DESY, Hamburg, Germany
S. Marsching, aquenos GmbH, Baden-Baden, Germany
J. Wychowaniak, Łódź University of Technology, Łódź, Poland

Abstract

Porting complex device servers from one control system to another is often a major effort due to the strong code coupling of the business logic to control system data structures. Together with its partners from the Helmholtz Association and from industry, DESY is developing a control system adapter. It allows writing applications in a control system independent way, while still being able to update the process variables and react on control system triggers. We report on the status of the project and the experience we gained trying to write portable device servers.

INTRODUCTION

With embedded systems becoming more and more powerful, the algorithms in the devices which are accessed via control systems are becoming more and more advanced. Especially on MicroTCA [1] systems the hardware usually features a powerful multi-core CPU with several GB of RAM. The MicroTCA.4 [2] extension brings trigger and clock lines, as well as large rear transition modules which can be used for demanding analogue control applications.

Many particle accelerators which are currently being build are using or will use MicroTCA.4 for control of the radio frequency (RF) in the accelerator, for instance FLASH [3] and the European XFEL [4] hosted at DESY, Hamburg, ELBE [5] at Helmholtz-Zentrum Dresden-Rossendorf, or FLUTE [6] at KIT, Karlsruhe. The complex RF control applications shall be reused across the different accelerators, while all the facilities are using different control systems. It turned out that porting the software to a different control system is a major effort because the code is strongly coupled to the original control system. This lead to the idea to have an adapter layer between the device library, which implements the algorithms, and the control system, which provides the communication protocols and integration into the facility's control infrastructure. This adapter shall be part of the MicroTCA.4 User Tool Kit (MTCA4U) [7, 8], a collection of libraries to facilitate the implementation of control applications.

REQUIREMENTS

The main task of the adapter is to allow application code to access process variables which are communicated to the outside world in a control system independent way. For this, the adapter has to use the functionality which is provided by the control system, like communication protocols or the addressing scheme.

The part of code which is device *and* control system dependent has to be minimal, zero if possible. This type of code is causing the huge workload when porting and maintaining applications for multiple control systems.

Abstraction is easy to achieve if data is simply copied back and forth between two domains, but this comes with a performance penalty. So an additional requirement to the adapter is to avoid unnecessary copying, especially of large data structures like arrays. In addition, the adapter should be thread safe and usable in real time application.

As a starting point and to check if the abstraction is working, the adapter is tested with two control systems: DOOCS [9] and EPICS [10]. DOOCS (used at DESY for FLASH and the European XFEL), has an object-oriented data model written in C++. EPICS 3, one the most widely used control systems for particle accelerators and used at FLUTE, has a channel-based C API. We intentionally used two conceptually different control systems, hoping that the abstraction needed to work with these two should also allow other control systems to be used without too much modifications in the design.

DESIGN CONCEPT

The first implementation of the adapter focuses on process variables. It provides data structures for scalars (8, 16 and 32 bit signed and unsigned integers, single and double precision floating point), strings, and arrays of the numerical data types. Each process variable is identified by a unique name which describes its function inside the device code ("TEMPERATURE" for instance for a device with a temperature sensor). The name does not contain information where the device is installed and in which context it is used. This part depends on the control system and the facility, and is added in the control system specific part of the adapter, not in the device part.

The original idea to avoid copying, especially for large arrays, was to have a single instance of the data. This would be stored in a control system dependent type, an instance which always has to be there to work with the particular control system. The adapter would provide a wrapper, which would be used inside the business logic. But it turned out that this approach is not viable. DOOCS and EPICS have different locking schemes for their variables, and only the control system side could know when it is safe to access

* This work is supported by the Helmholtz Validation Fund HVF-0016 "MTCA.4 for Industry".
† martin.killenberg@desy.de

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

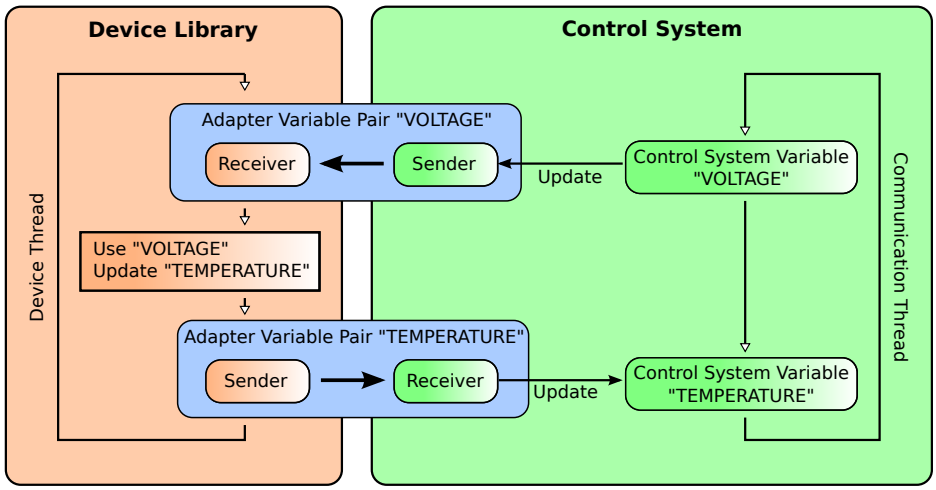


Figure 1: The update flow using the control system adapter.

the variables, but not the control system independent device part.

The solution is to have a control system independent instance on the device side which can be accessed at any time, and the control system variable on the other side, which can have a control system lock (Fig. 1). These variables have to be synchronised, which means that the abstraction at this point requires one copy which cannot be avoided.[1] As not all control systems allow a variables to be input and output at the same time, it was decided to restrict process variables to be unidirectional ("control system to device" or "device to control system").

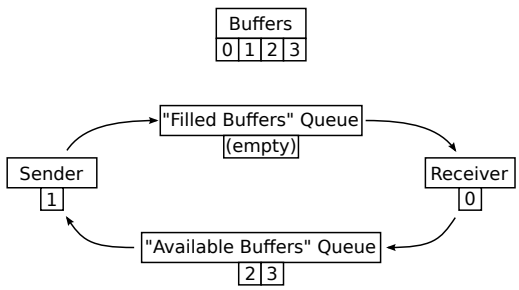


Figure 2: For arrays a process variable pair with sender and receiver features two lock-free queues and at least four pre-allocated buffers.

As one of the requirements for the adapter is to allow real time threads in the device library, the variable on the device side has to be lock-free. To implement this, the adapter's process variable is always a sender-receiver pair (Fig. 2), using lock-free queues for transfer. As dynamic memory allocation is not allowed in a real time thread, the mechanism is working with a pool of pre-allocated buffers for arrays. Sender and receiver each hold the reference to one buffer at all times, which allows the business logic to access and modify the data at will, except while sending or receiving. The data being transferred is always the reference to a buffer, not the buffer itself, which avoids unnecessary copying of large data structures.[2]

When receiving, the receiver will pop the head of the "filled buffers" queue. If it could get (the reference to) a new buffer, it will push the now outdated buffer to the "available buffers" queue, so the sender can reuse it. In case there are no updated buffers available, the receiver will hold on to the current buffer, as it contains the most up-to-date information that is available on the receiver side.

Before actually sending, the sender will pop the head of the available buffers queue to be sure it has a new buffer which it can fill after sending (at all times there must be at least one buffer on each the sender and the receiver side). After that, the buffer to be send is pushed to the "filled buffers" queue. Both receiver and sender first pop the head of the queue where they retrieve the next buffer, and then push the buffer which has been processed for use by the other side. As this can happen at the same time, it means there have to be at least four buffers.

In contrast to a triple buffer, which is a common scheme in real time applications, this approach does not require a dirty flag for the buffer, and the receiver does not have to swap back to keep the latest available buffer in case no updated buffer is found. As a further advantage, the number of buffers can simply be increased, allowing a longer queue in case there are fluctuations on the receiver side, but it is fast enough to catch up with a certain backlog.

If the receiver is too slow to process data at the rate at which the sender is producing it, data will be lost. This cannot be avoided. In our implementation with a fixed number of buffers it means the "available buffers" queue is empty. In a naive implementation, the sender would keep its current buffer to overwrite it (buffer 3 in Fig. 3). However, it is not a good solution to stop sending because this would result

[1] A copy can be avoided if the control system type allows swapping of the internal buffer.

[2] For scalars copying the value is not more expensive than copying the pointer. In this case the values are directly copied and no additional buffers are needed.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

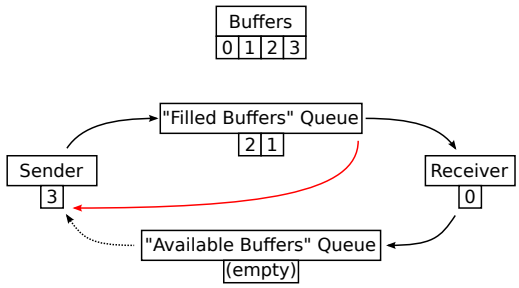


Figure 3: If the queue of available buffers is empty, a further send call would pop the head of the "filled buffers" queue to be overwritten (buffer index 1 in this picture), and perform the send operation (buffer index 3 goes to the queue). Like this, the information in the queue can be updated, whether the receiver is active or not.

in newer data being discarded in favour of older data that is already in the queue. If for instance the receiver was down for several minutes, the information in the queue would be several minutes old when it is received, while newer data has been overwritten. Instead, the oldest information which has not yet been received should be dropped. So in case no free buffers are available, the sender will pop the head of the "filled buffers" queue (buffer 1 in Fig. 3) to be overwritten, and send buffer 3, which has just been filled. Like this, the data in the queue is being updated even if the receiver is not active.

Creation of a Process Variable

Not only the device business logic has to be independent from the control system side, also the amount of device-specific control system code should be minimised. For this reason the creation of process variables is automated as much as possible in the adapter (Fig. 4). The device is calling the create function of the adapter, giving the name and the direction ("device to control system" or "control system to device"). Depending on the direction, a sender or a receiver is returned to the device side. The other partner of the process variable pair is stored in a list. After all process variables have been created by the device, the control system calls a function which creates the instances of the control system variables for all process variables known to the adapter. This function does not depend on the device logic, which improves the decoupling. It is, however part of the control system specific part of the adapter and looks different for each control system. The level of abstraction which can be achieved here may vary.

STATUS AND OUTLOOK

The current implementation of the control system adapter provides process variables in a common, control system independent part. A control system specific, but device independent part has to be added to the adapter for each target control system. Bindings for DOOCS and EPICS have been written and are ready to use.

As "proof of concept", some example devices with a couple of scalars and arrays have been created and tested using either DOOCS or EPICS, and have afterwards been ported to the other control system. This porting worked smoothly in both directions. As expected, the device code itself did not have to be modified. The amount of device-specific code is very small on the control system side. The example devices are down to three lines of device-dependent control system code, for DOOCS as well as for EPICS, independent from the number of process variables.[3]

To make use of additional control system features like variable limits or histories, these currently have to be implemented as device-specific code on the control system side. This introduces additional work when porting and maintaining the code for multiple control systems. It is currently being discussed how the adapter can be extended to allow the definition of the features on the device side, but use the functionality from the control system. Especially features which are expensive to implement like archiving and history should not be duplicated in the adapter.

CONCLUSIONS

The MTCA4U control system adapter provides an interface to use process variables in a device library without introducing a coupling to a particular control system. The implementation is lock free and transfers pre-allocated buffers without copying. This step was necessary to allow operation in control systems with different locking mechanisms. It also enables the device logic to use process variables in a real time thread.

The adapter consists of a common part, which implements the decoupling, and a control system specific part, which provides the particular bindings. As a proof of concept, adapters for DOOCS and EPICS have been implemented. An extension for OPC-UA is planned. As next steps, the abstraction of additional control system features like variable limits, engineering units and histories are intended.

All software is published under the GNU General Public License and available in the respective software repositories [11–13].

REFERENCES

[1] PICMG®, "Micro Telecommunications Computing Architecture, MicroTCA.0 R1.0" (2006).

[2] PICMG®, "MicroTCA® Enhancements for Rear I/O and Precision Timing, MicroTCA.4 R1.0" (2011/2012).

[3] C. Schmidt et al., "Real time control of RF fields using a MicroTCA.4 based LLRF system at FLASH", 19th IEEE Real-Time Conference, Nara, Japan (2014).

[4] M. Altarelli et al., "XFEL : The European X-Ray Free-Electron Laser : Technical Design Report", DESY-2006-097, DESY, Hamburg (2007).

[3] The device configuration files for the control system had to be written in addition.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

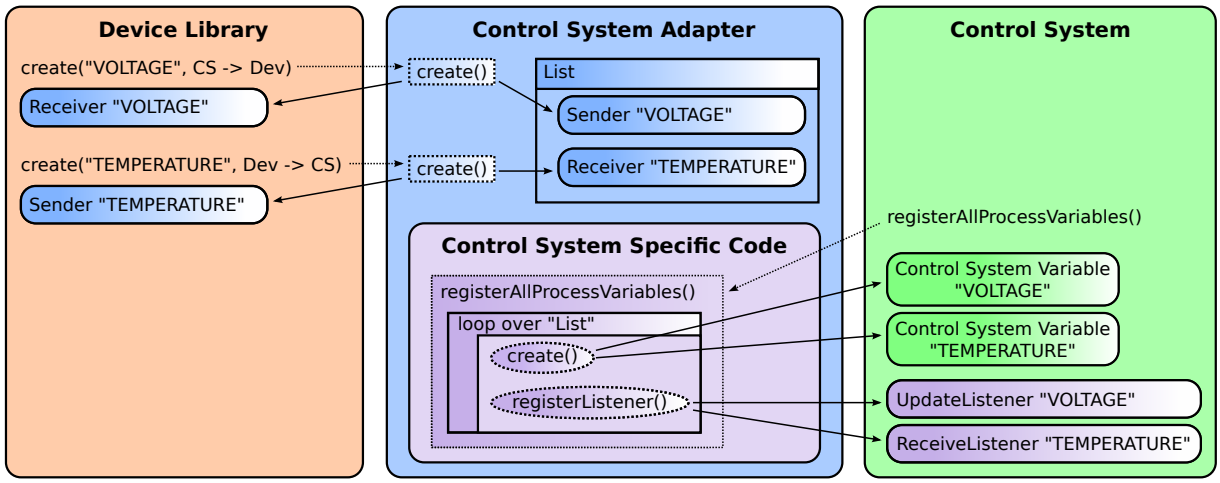


Figure 4: The creation of process variables is requested on the device side. The instantiation on the control system side is automated as much as possible inside the adapter.

[5] F. Gabriel et. al., "The Rossendorf radiation source ELBE and its FEL projects", Nucl. Instr. Meth. B 161-163, 1143 (2000), http://dx.doi.org/10.1016/S0168-583X(99)00909-X

[6] S. Marsching et al., "Status of the FLUTE Control System", WPO013, PCaPAC2014, Karlsruhe, Germany (2014).

[7] M. Killenberg et al., "Drivers and Software for MicroTCA.4", WEPGF015, *These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

[8] MTCA4U—The DESY MicroTCA.4 User Tool Kit, Subversion Repository, https://svnsrv.desy.de/public/mtca4u

[9] The Distributed Object Oriented Control System (DOOCS), http://doocs.desy.de/

[10] Experimental Physics and Industrial Control System (EPICS), http://www.aps.anl.gov/epics/index.php

[11] MTCA4U Control System Adapter, Subversion Repository, https://svnsrv.desy.de/public/mtca4u/ControlSystemTools/

[12] MTCA4U EPICS Adapter, Subversion Repository, http://oss.aquenos.com/svnroot/epics-mtca4u/

[13] MTCA4U DOOCS Adapter, Subversion Repository, https://svnsrv.desy.de/public/mtca4u_applications/DOOCS_Adapter/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE TANGO CONTROLS COLLABORATION IN 2015

Andrew Götz, Jean-Michel Chaize, Emmanuel Taurel, Pascal Verdier,
Jean-Luc Pons, Tiago Coutinho, Faranguiss Poncet, Reynald Bourtemburg, ESRF, Grenoble, France
Gwenaelle Abeille, SOLEIL, Gif-sur-Yvette, France, Sandor Brockhauser,
Lajos Jeno Fulop, ELI-ALPS, Szeged, Hungary, Alejandro Vázquez-Otero, ELI-BEAMS, Prague, Czech Republic
Mihail Octavian Cernaianu, IFIN-HH, ELI-NP, Bucharest - Magurele, Romania
Cristina Knapic, Ricardo Smareglia, INAF, Trieste, Italy
Igor Khokhriakov, Helmholtz-Zentrum Geesthacht, Geesthacht, Germany

Abstract

This paper presents the latest news from the TANGO collaboration. TANGO is being used in new domains. The three ELI pillars - ELI-Beamlines, ELI-ALPS and ELI-NP in Czech Republic, Hungary and Romania respectively have selected TANGO for many of their control systems. TANGO will be extended with new features required by the laser community. These features will include nanosecond time-stamping. The latest major release of TANGO V9 includes the following features - data pipes, enumerated types, dynamic commands and forwarded attributes. A prototype REST API (mtango) has been developed. The collaboration has been extended to include the new members and to provide a sustainable source of resources through a collaboration contract. A new website (http://www.tango-controls.org/) has been designed which has improved the communication within the community. Finally the updated roadmap is summarised.

THE COLLABORATION

New Sites

The TANGO Controls collaboration has continued to grow. New sites since the last ICALEPCS in 2013 are ELI-BEAMS, ELI-ALPS, ELI-NP, VIRGO EGO and SKA to mention a few. In 2015 two new TANGO-based synchrotrons have been commissioned – Solaris (Poland) and MAX IV (Sweden).

Student Trainings

The number of students attending classes at high schools, universities and training courses has increased. TANGO courses are now available as part of a Bachelor's degree in Grenoble (France), and as part of a Master's degree offered by the University of Szeged (Hungary). TANGO is used for training courses at the EPFL in Lausanne (Switzerland). More students learning TANGO means it is possible to find trainees for TANGO related projects. A group of French students submitted a TANGO project running on a Zynq platform to the Open Hardware 2015 Xilinx student competition. The IAEA has sponsored 3 training courses on TANGO as part of a further education programme in Northern Africa.

TANGO V9

The TANGO approach to development is to continue improving the code base by making regular major releases. The latest major release is version 9. It has been in beta test at the ESRF since October 2014. The official source code release 9.1.0 was made in October 2015.

Why is it a major release? Firstly because of the addition of a new communication channel – pipes. Secondly because there is a new device interface (Device_5Impl). Thirdly because of new implementations in Java for device servers and in Python with a more Pythonic interface. As always the new release is backward compatible with all previous versions. The following sections summarises the new features which have been added to TANGO V9.

Enum Attributes

A common need in control systems is to allow only a restricted set of values for certain attributes e.g. SINGLE, MULTI, HYBRID filling patterns. This is usually solved in programming languages using enumerated types. This needs special treatment when traversing the network. Version 9 has implemented enumerated attributes to make programmers and users lives easier.

Forwarded Attributes

Attributes are the well defined data exchanged between device servers and clients. In a hierarchical distributed control system system it is common to build hierarchies of device servers. When doing so it is very useful to be able to make some of the attributes of sub-devices available as attributes of other higher level devices. In the past this could only be done by manually re-programming the passing of the attribute of the sub-device in the higher level device. Version 9 has imported the notion of forwarded attributes which require no programming. They only need to be declared in the database. Less code means more reliability.

Pipes

TANGO supports a wide set of data types. For exchanging data these are mostly made up of scalars or arrays of scalars of the same type. This is sufficient in most cases. However there are some cases which have a need to exchange data of varying types coming from the same source. A typical example of this is when returning scanned data from an experiment scanning server. In this case it is useful to mix data types on the same communication channel. A new data communication channel, called pipes, has been introduced to address this problem. Version 9 introduces Tango Pipes for sending and receiving blobs of data of mixed types. Blobs of blobs are supported too.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Polling Thread Improvement

In previous version of TANGO the polling thread which is the main mechanism for generating events, accessed device attributes one by one. This could lead to longer than necessary readout times. Version 9 has modified the polling thread to read multiple attributes in one call. This is more efficient, leads to better performance and fewer errors.

Dynamic Commands

Previous versions of TANGO allowed on dynamic attributes. Dynamic means they are created at runtime as opposed to statically at compile time. Prior to Version 9 dynamic commands were only possible in Java. Version 9 has added the possibility of defining dynamic commands in C++ and Python device classes too.

New Java Device Implementation

For a long time the TANGO Java server implementation was frozen. In 2010 it was un-frozen and completely rewritten using the latest features of Java. Support was added for annotations at compile or runtime. This allows any POJO to be turned into a TANGO device. The following annotations are supported :

@Device: class
@Attribute: field
@Command: method
@State: field
@Init: method
@DeviceProperty

The latest version of the TANGO Java server and client api support all the new features of TANGO V9 (cf. above).

High Level PyTango API

The Python implementation of Tango is based very closely on the C++ API. This approach is not very natural for Python programmers. Consequently a Tango Enhancement Proposal (TEP1 [1]) was proposed with the following rationale - "*It would be nice if non tango experts could create tango device servers without having to code some obscure tango related code. It would also be nice if the tango programming interface would be more pythonic. The final goal is to make writing tango device servers as simple as possible*". TEP1 has been implemented as the High Level API (HLAPI) [2] in PyTango 8.1.2. The HLAPI has been implemented as a template in Pogo to generate pythonic Python Tango device servers. The HLAPI is based on PyTango binding. Consequently programmer's have access to all the low level features of PyTango if needed. Here is a simple example on how to write a *Clock* device server using the high level API:

```
import time
from PyTango.server import run
from PyTango.server import Device, DeviceMeta
from PyTango.server import attribute, command

class Clock(Device):
    __metaclass__ = DeviceMeta
    time = attribute()

    def read_time(self):
        return time.time()

    @command(din_type=str, dout_type=str)
    def strftime(self, format):
        return time.strftime(format)

if __name__ == "__main__":
    run((Clock,))
```

mtango REST API

Growing demand in web and mobile applications that would communicate with TANGO devices was the reason of a request for a new API. A prototype has been designed and implemented following the REST semantics. The API simplifies the access to devices from the browser or mobile clients. Web/mobile clients can be written in JavaScript, Java, Python or any other language that supports http. Security is implemented via basic http authentication and integrated with the TANGO Access Control (TAC). Refer to the documentation [3] for more information. See Fig. 1 for a schematic of the architecture.

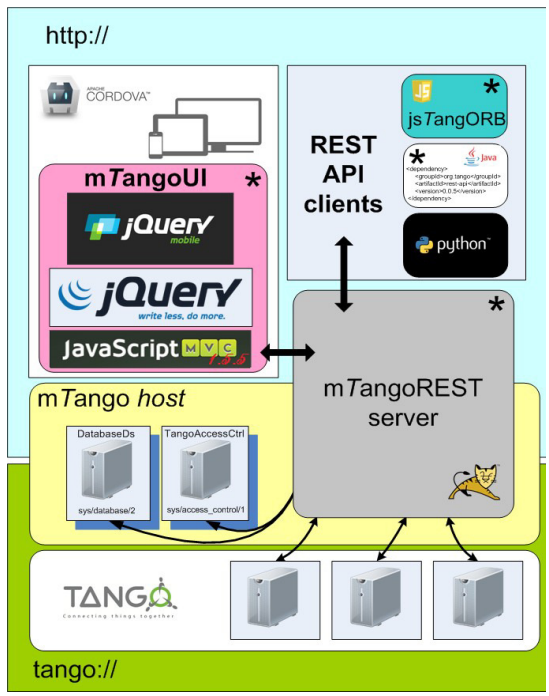


Figure 1 : mtango REST API architecture

This prototype is a starting point for a new TANGO REST API specification (see roadmap section below).

New Java Client Library

A new Java library – ezTangORB [easyTangORB] - has been added. This library facades the standard Java Tango library. ezTangORB provides convenient abstractions to

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

most commonly used features: reading/writing attributes, commands execution and handling events subscription.

NEW WEBSITE

A new web site http://tango-controls.org has been developed. The website contents was designed with the help of a communications specialist while the layout was done by a company specialised in web design. New communication tools like discussion forums, news and event pages are now extensively used. By allowing live exchange between beginners and specialists, these new tools foster knowledge sharing and help newcomers to learn TANGO. The next step is to boost code sharing via the website by setting up a market place (see Roadmap section).

NEW MEMBERS

The TANGO community has continued to grow into new domains – lasers, astronomy, gravitational wave and radio astronomy etc. How TANGO will be used in a selection of the new larger projects are described briefly below.

ELI-BEAMS

ELI-Beamlines [4] will be a high-energy, repetition-rate laser pillar of the ELI (Extreme Light Infrastructure) project. It will be an international facility for both academic and applied research, planned to provide user capability since the beginning of 2018. The main objective of the ELI-Beamlines project is the delivery of ultra-short high-energy pulses (between 10 and 150 fs) for the generation and application of high-brightness X-ray sources and accelerated particles.

The Central Control System (CCS) design is based on TANGO, whilst the Laser Controls (LC) will use a combination of EPICS and LabView. The CCS will talk to the LC using the already existing Tango2Epics Gateway. Matlab and Labview will be extensively used in the experimental chambers for controlling local equipment, though all of them will be integrated into the CCS through adequate Tango device servers.

ELI-ALPS

The primary mission of the ELI-ALPS [5] research facility (Hungary) is to make a wide range of ultrashort atto-second light sources accessible to the international scientific community user groups. Laser driven secondary sources emitting coherent extreme-ultraviolet (XUV) and X-ray radiation confined in atto-second pulses is a major research initiative of the infrastructure.

ELI-ALPS will use TANGO Controls in several places. The control systems and the gateways of the beam transport and secondary source systems will be based on TANGO, as well as the gateways of the turn-key laser source systems delivered by external suppliers. The integration of these systems will be also based on TANGO as well as the central control system.

ELI-NP

The ELI-NP facility [6] will have a dedicated major control system for the HPLS (High Power Laser System), developed in TANGO and a second dedicated control system for the Gamma Beam (GBS) developed in EPICS. The facility requires a modern implementation of a distributed monitoring and control system for the experiments that are proposed in the eight available experimental areas E1-E8.

In the HPLS based experiments, multiple TANGO control systems will be developed and deployed, one for each experiment, for redundancy, safety and better maintenance. In the case of the GBS based experiments, one EPICS framework will handle all the experiments. Additionally, the Laser Beam Transport System which routes the laser beams into the experimental areas will be controlled using TANGO in order to provide a unified interface with the HPLS and the experiments.

In the first stage of the development, the TANGO device servers will only implement access to the HW parameters of the devices and not the logic (e.g. only the means to send commands and receive the status from a motor and not the logic to perform an automatic alignment). Due to a long period of testing, development and fine tuning will be needed to implement the logic. This will be made in the HMI layer and not inside the device servers. As developments advance and stable working solutions emerge, the maths and logic from the supervisory HMI and client HMI will be progressively shifted from LabVIEW and Matlab to the device servers themselves.

Various TANGO implementations have already started at ELI-NP, for interfacing equipment that will be used in the experiments (motor controllers, CCD cameras, etc) and performance tests are being carried out.

INAF

A new use of TANGO was recently implemented by INAF- Trieste Observatory [7] of Trieste to handle the Distributed Large Binocular Telescope Data Archive and the INAF Radio Archive. Using the features offered by TANGO, a configurable infrastructure to ingest and replicate astronomical data over geographically distributed sites was developed. Since the INAF institute joined the TANGO Controls collaboration, the TANGO control system has triggered interest in several astronomical fields, including instrumental control in more complex infrastructures of Radio Astronomy. INAF has foreseen to hold a number of internal TANGO trainings in the near future.

SKA

The Square Kilometer Array (SKA) [8] is currently the most challenging astronomical project world wide. It will be composed of several thousand antennas arrays and dishes distributed over two sites (South Africa and Australia). It presents several technological very challenging issues. The SKA community has shown strong interest in the TANGO control system for the implementation of several parts of the project. A large part

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

of the consortium in charge of controls met in Trieste March 2015 to discuss a common control system infrastructure. TANGO is currently considered the most complete, well suited and modern framework capable of fulfilling all the requirements. Two trainings were organized in order to learn and understand the paradigms and features of TANGO. Presentations, meetings and a webinar on TANGO are planned within the SKA community.

ARCHIVE DATABASE

A large project called HDB++ has been started to improve the archive database in TANGO. The new database supports SQL and NoSQL type databases. It offers orders of magnitude more in time resolution. The HDB++ project is described in detail in two other papers in these proceedings ([9] and [10]).

COLLABORATION CONTRACT

To date the TANGO collaboration is governed by a Memorandum of Understanding. Institutes who have signed the memorandum form the Executive Committee (EC). Major decisions about TANGO are taken by the EC. In order to ensure the sustainability of TANGO the MOU will be replaced by the TANGO Controls Collaboration in 2016. The collaboration will be governed by a so-called collaboration contract. Institutes who sign the contract will commit to financing the development of TANGO and participate in the decision making process. Two types of members will be recognised – core members and contributing members. All members contribute financially to maintaining TANGO. Core members contribute in addition code to the core. The new organisation does not change the free open source nature of TANGO i.e. TANGO remains freely available and open source for everyone. The new organisation will ensure the sustainability of TANGO by financing resources to work on tasks of common interest like the roadmap.

ROADMAP

TANGO is a mature and reliable toolkit to build distributed control systems for installations which need to run for the next 10 to 20 years. For this reason TANGO needs a roadmap which will ensure its future evolution and stay a good choice for the coming years. At the TANGO meeting in May held at Solaris in Krakow (Poland) the community gave their input on the essential features of the current roadmap. Input was provided via email and discussed during an interactive session. The results presented in [11] are summarised here.

(1) **Improve Documentation** – re-factor and consolidate documentation, write a cook book of recipes and concepts, (2) **Move to Git** – move source code repository from svn to git, (3) **Remove CORBA completely** – introduce and abstract transport layer and replace the underlying synchronous synchronous CORBA protocol with ZMQ, (4) **Grow the community**, (5) **REST API** – implement a REST API for TANGO, (6) **Web browser application** – implement a device browser for the web, (7) **Secure encryption** – implement a secure encrypted protocol for public networks, (8) **Database performance** – improve to remove bottlenecks for memorised attributes, (9) **Device class Marketplace** – implement a marketplace to improve code sharing, (10) **Long Term Support** – provide upgrades for older versions, (11) **Tango Virtual Machine** – upgrade virtual machine to latest versions, (12) **Auto-generate Unit tests** – Pogo to generate unit tests automatically, (13) **SysML support** – add support for using SysML to specify device servers, (14) **Replace Boost.Python** – replace boost.python in PyTango with a lighter weight solution.

A number of the features on the roadmap have already been implemented or are in the process of being implemented e.g. 5, 8, 11. Others (2, 9, and 12) will be addressed in the short term. The remainder require more resources and will be implemented as part of the next major release. The Collaboration Contract will be extremely helpful in finding resources to implement the roadmap.

CONCLUSION

TANGO Controls is a growing collaboration for building distributed control systems with a clear roadmap for the future. The new members of the collaboration will provide a big boost to the community in terms of new applications and developments. The new collaboration contract will help ensure the sustainability of TANGO for the next years.

ACKNOWLEDGEMENT

The TANGO Controls community for their ideas, input and innovative use of TANGO.

REFERENCES

[1] Tango Enhancement Proposal 1 - TEP1: http://www.esrf.eu/computing/cs/tango/tango_doc/kernel_doc/pytango/latest/tep/tep-0001.html#pytango-tep1
[2] PyTango High Level API – HLAPI: http://www.esrf.eu/computing/cs/tango/tango_doc/kernel_doc/pytango/latest/server_api/server.html
[3] mtango REST API documentation: https://bitbucket.org/hzgwpn/mtango/wiki/browse/
[4] ELI-BEAMS: http://www.eli-beams.eu/
[5] ELI-ALPS: http://www.eli-alps.hu/
[6] ELI-NP: http://www.eli-np.ro/
[7] INAF – Trieste: http://www.oats.inaf.it/
[8] The Square Kilometer Array – SKA: https://www.skatelescope.org/
[9] L.Pivetta et al., "*HDB++: A New Archiving System for TANGO*", WED3O04, ICALEPCS 2015, these proceedings.
[10] R.Bourtembourg et al., "*How Cassandra Improves Performances and Availability of HDB++ Tango Archiving System*", WEM310, ICALEPCS 2015, these proceedings.
[11] Tango Roadmap presentation at Tango meeting - http://ftp.esrf.eu/pub/cs/tango/meetings/2015_may_solaris/Andy_Gotz_Roadmap_presentation.pdf

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

RECENT ADVANCEMENTS AND DEPLOYMENTS OF EPICS VERSION 4

G. White, M. Shankar, SLAC, Menlo Park, California, USA
A.N. Johnson, S. Veseli, ANL, Argonne, Ilinois, USA
A. Arkilic, L.B. Dalesio, M. Davidsaver, M.R. Kraimer, N. Malitsky, B.S. Martins, BNL, Upton, Long Island, New York, USA
M. Sekoranja, Cosylab, Ljubljana, USA
D.G. Hickin, DLS, Oxfordshire, England
T. Korhonen, ESS, Lund, Sweden
G. Shen, FRIB, East Lansing, Michigan, USA
R. Lange, ITER Organization, St. Paul lez Durance, France
S.M. Hartman, K. Kasemir, ORNL, Oak Ridge, Tennessee, USA

Abstract

EPICS version 4 is a set of software modules that add to the base of the EPICS toolkit for advanced control systems. Version 4 adds the possibility of process variable values of structured data, an introspection interface for dynamic typing plus some standard types, high-performance streaming, and a new front-end processing database for managing complex data I/O. A synchronous RPC-style facility has also been added so that the EPICS environment supports service-oriented architecture. We introduce EPICS and the new features of version 4. Then we describe selected deployments, particularly for high-throughput experiment data transport, experiment data management, beam dynamics and infrastructure data.

EPICS BASE AND EPICS VERSION 4

The Experimental Physics and Industrial Control System (EPICS), is a software framework for high-performance distributed control and data acquisition in large scientific instruments, such as accelerators and telescopes.

The "base" of EPICS is software for supervisory control, closed loop control, archiving, alarm management, timing and other aspects of front-end processors and device facing hardware. Typically hosted in an embedded system processor such as RTEMS or VME, this software and its host are collectively known as the IOC (Input/Output Controller) in an EPICS control system. IOCs are optimized for low-latency I/O. They control and/or monitor a collection of devices such as actuators and measurement diagnostics. Each IOC node contains a memory resident real-time database.

The IOC database is a set of "smart" records, which are interconnected in a *data flow* pattern. They're smart in that their field values may come directly from hardware, or a result of processing that was dependent on the type of record. The records may contain "device support" code, to interface the processing to physical devices through device drivers. Much more information can be found on the EPICS base at [1, 2].

EPICS base and the software extensions built on top of it have proven very successful for the control aspects of scientific instruments, providing excellent low level I/O, DAQ and optimal control.

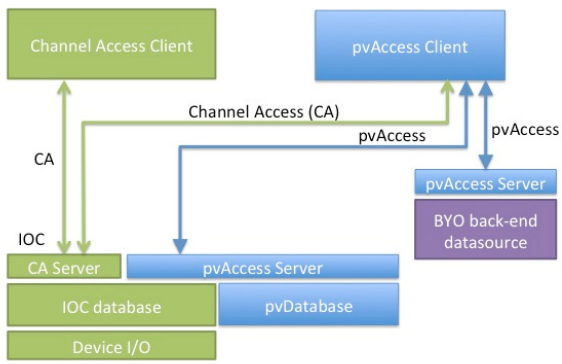


Figure 1: The basic architecture of EPICS version 4, showing the classic "base" components of EPICS in green, and components added by version 4 in blue. A new protocol, pvAccess, transports potentially complex data types encoded by pvData. pvAccess connects clients to IOCs, which may include a new database type, pvDatabase to support complex data processing services (and is the basis of the high-performance detector streaming applications described here), and to middleware services that can themselves connect to enterprise data stores, the web, etc. Note that hardware device I/O remains under the IOC database.

EPICS Version 4 for New Controls Problems

Modern control systems of large instruments call for more science to be done in the control system itself, than is possible with the EPICS base software alone. High output detectors call for pipelined data processing; physics applications deal with systems of process variables and their values rather than one process variable (PV) at a time; process variables may be structured, multi-modal, or require significant metadata associated with values in order to be understandable (such as images, or timing related metadata to give context when reviewing archives).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

EPICS version 4 is a system of software modules that extend EPICS base to address these emerging requirements.

Core Modules of EPICS Version 4

Apart from EPICS base, the two core modules of EPICS version 4 are "pvData" and "pvAccess."

The pvData module is the high-performance structured data backbone of EPICS version 4. It provides an introspection interface for the dynamic creation and management of structured data types and arrays. Although types can be created on the fly, EPICS version 4 also defines a standard set of data types oriented towards scientific controls data (called "Normative Types" [3]). The types there defined include matrices, key-value sets, tables, histograms, continua, and images, among others both generic and application specific.

pvAccess is the network protocol of EPICS v4. Like the equivalent protocol in version 3, "Channel Access", named endpoints (PVs) are discovered by broadcast, while data I/O is by TCP. In addition to the expected get, put, and put/get methods one would expect, pvAccess supplies a Remote Procedure Call (RPC) method, and a method to deal efficiently with exchanging only the changed elements of large arrays. Working in concert with pvData, the two include memory management and efficient encoding and deserialization, to minimise copy and wire transactions for even large complex data I/O. For instance, in the case of a PV defined by a complex structure, a client can subscribe only to the fields of the structure of interest to that client [4].

The RPC channel method enables a client to pass arguments along with a PV value request, in which case the value is computed with respect to the arguments.

Notably, the same PV may be accessed by either an asynchronous get method, or an RPC method. In this way, a get-value operation on one PV, can be either the familiar "monitor", or the "compute value subject to given parameters and return result." In either case, the processing path (and so potentially the PV's value) would be different. This RPC facility is being used extensively at SLAC, and is being adopted also at ESS, for beam dynamics (see Figure 2), infrastructure, and directory service data.

```
$ eget -s XCOR:IN20:491:RMAT -a b BPMS:IN20:525
0.669544 0.694654 0        0        0        0
-0.57085 0.901274 0        0        0        0
0        0        1.333434 0.966896 0        0
0        0        0.358411 1.009578 0        0
0        0        0        0        1        0
0        0        0        0        0        1

$ eget
pva://mccas0.slac.stanford.edu/XCOR:IN20:491:RMAT?b=BP
MS:IN20:525

$ eget QUAD:LI24:900:TWISS
    energy 5.00512
    psix 37.7625
    alphax 13.6562
    betax -2.78671
    etax -0.00698294
    etaxp 0.00107115
    psiy 31.9488
    alphay 116.762
    betay 5.2592
    etay 0
    etayp 0
    z 2438.72
```

Figure 2: Examples of three *normative types* used with the new pvAccess command *eget*. The first shows an RPC request for the response matrix between two accelerator devices (sometimes called "rmat A to B"), in LCLS. The PV is "XCOR:IN20:491:RMAT". An argument named "b" whose value is the name of the second device was also given. Since the returned data self identified as normative type *NTMatrix*, eget displayed the data appropriately. The second example is of eget being given the same request in URL form. In fact, in both the first and second examples, the server received its request in the form of the normative type *NTURI*. The last example shows use of *NTNamedValue* to get and display the Courant-Snyder parameters of a quadrupole.

RECENT ADDITIONS TO EPICS VERSION 4

The pvAccess system is now codec based; the communication layer is decoupled from the transport layer, allowing alternative transport layer protocols such as one using zeroMQ. A codec for the former transport layer of pvAccess is provided.

Array handling has been greatly enhanced in the C++ implementations of the core modules. Reference counted shared vectors help minimize copy operations. Fixed and bounded arrays have been added in order to more easily facilitate high-performance array management. These complement the existing unbounded arrays. Stride length can be specified in put and get operations.

The Normative Types have been extended to include full dynamic typing (essentially an "Any" in the CORBA sense).

A plug-in architecture for implementing security schemes has been added, along with the plugin for Channel Access security.

The command line tools have been extended to better support time stamp, alarms, and correct display of enumerated types. All of the command line tools now support both pvAccess and Channel Access operations. A new tool, *pvlist* gives information about pvAccess servers on the network, including lists of the PVs they host.

Extensive API changes have been made to give greater flexibility for channel callbacks and converting from one introspection data type to another.

New Data Processing Database

The new "pvDatabase" module of EPICS v4 implements a framework for a memory resident database of records defined in terms of pvData structures. Like the IOC database of classic EPICS, the records of pvDatabase can process on I/O events; unlike the IOC the records may be of any structure the engineer wishes, and may pull in data from any pvAccess-ible data source, plus Channel Access. pvDatabase images may be standalone, or hosted within an IOC, where they might interface directly to base records, asynchronous device driver support (asynDriver), or detector control (areaDetector). pvDatabase is then useful for complex optimal control tasks, data assembly, and preprocessing. Combined with pvAccess streaming, it can be used as the basis of a data processing pipeline.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

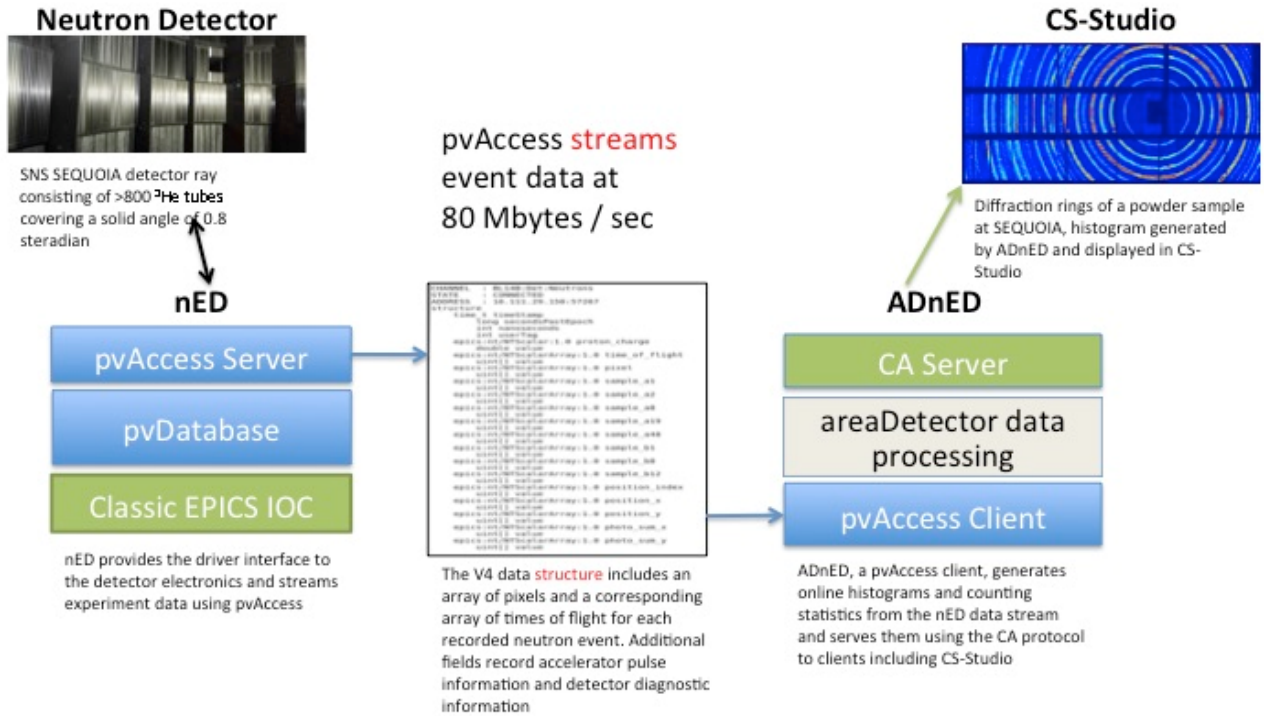


Figure 3: Outline dataflow for neutron detector data transport at SNS. An EPICS version 4 IOC (*nED*) takes data from *areaDetector* – a well-known detector data system – assembles it into a structured package using pvData, and transports it to EPICS v4 clients over pvAccess. One client in particular, *ADnED*, disassembles the data into classic PVs for consumption by existing display tools such as CS Studio. *Thanks to S. Hartman and K. Kasemir, ORNL*

SELECTED DEPLOPYMENTS

The primary uses of EPICS version 4 have so far been found to be in high-performance detector data fan out, and in middleware data services.

EPICS v4 is used at SNS in data acquisition and processing of neutron scattering experiments [4], see figure 3. Neutron scattering is a technique complementary to X-ray scattering used in the understanding of the structure and properties of materials. At SNS, detector image pixel data from an experiment is combined with meta-data such as probe pulse id and charge, into a pvData structure, which then fully describes an experiment observation event. The event data is then streamed by pvAccess to clients for processing and analysis. Clients may subscribe to the fields of the event structure of interest to them. The primary client – effectively an EPICS v4 plugin for areaDetector – includes statistical processing and publication of selected fields as histograms by Channel Access, for consumption by existing GUI tools.

Diamond and BNL have also developed an areaDetector pipeline based on EPICS version 4, and the BNL framework is being distributed with areaDetector [5, 6]. Figure 4 shows a CS-Studio image from an Eiger detector image processed with the BNL pipeline.

Both BNL and SNS have thoroughly investigated EPICS v4 network performance. Their findings are that pvAccess' network efficiency is constant, well-behaved, and capable of delivering at or near 95% of the nominal maximum bit rate on 1 Gb/s or 10 Gb/s Ethernet, with no CPU saturation [4, 5].

BNL have also developed an interesting experimental beam-line data management system on EPICS version 4 [7]. The raw experiment data is dissociated from the metadata for storage and management, so that different and appropriate technologies can be used for the two different data categories. In particular, since the experiment data may itself be very differently structured from one experiment to the next, MongoDB is used for the store, and EPICS v4's flexible data typing can be used for high-performance transport. One particular EPICS v4 service, the "databroker," can interface to all experiment and metadata to provide a single interface to all.

FRIB and BNL have been using a PV configuration management tool named *MASAR* for some time. MASAR uses pvAccess to deliver sets of CA PV values to clients.

A collaboration of SLAC and ESS, are using EPICS version 4 for beam dynamics modelling services and device infrastructure data. Now EPICS (base and v4) provides all the data required for commissioning applications – both device PV values, plus optics, response matrices, device type lists etc. Services for beam synchronous data and mass archive data are planned.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

CONCLUSIONS

EPICS can be easily integrated into an EPICS installation to supplement existing device support, DCS and SCADA. It effectively extends EPICS to support structured scientific data and complex processing. Early users have found it particularly useful for high-throughput detector data and for integration of beam dynamics and accelerator enterprise data, into the controls system.

THANKS

The authors wish to thank all users and testers of EPICS version 4 for their excellent feedback and guidance. Questions and feedback can be given on EPICS *tech-talk*.

REFERENCES

[1] EPICS Version 4 project website: http://epics-pvdata.sourceforge.net
[2] EPICS version 4 source code repositories, https://github.com/epics-base
[3] G. White, L. Dalesio, M. Rivers, M. Kramer, D. Hickin, EPICS V4 Normative Types, http://tinyurl.com/l3pypbc
[4] K. U. Kasemir, G. S. Guyotte, M. Pearson, EPICS V4 Evaluation for SNS Neutron Data, WEPGF105, these proceedings, ICALEPCS 2015, Melbourne Australia.
[5] B. Martins, M. Davidsaver, areaDetector EPICSv4 modules, EPICS Meeting spring 2015, Michigan State, Michigan, U.S.A., http://tinyurl.com/os29bla
[6] D. Hickin, EPICS V4/ areaDetector Integration, areaDetector, Diamond, D.L.S., Didcot, Oxfordshire, U.K. (2014), http://tinyurl.com/pngcfb3
[7] A. Arkilic, NSLS-II Data management Framework, EPICS Meeting spring 2015, Michigan State, Michigan, U.S.A., http://tinyurl.com/os29bla

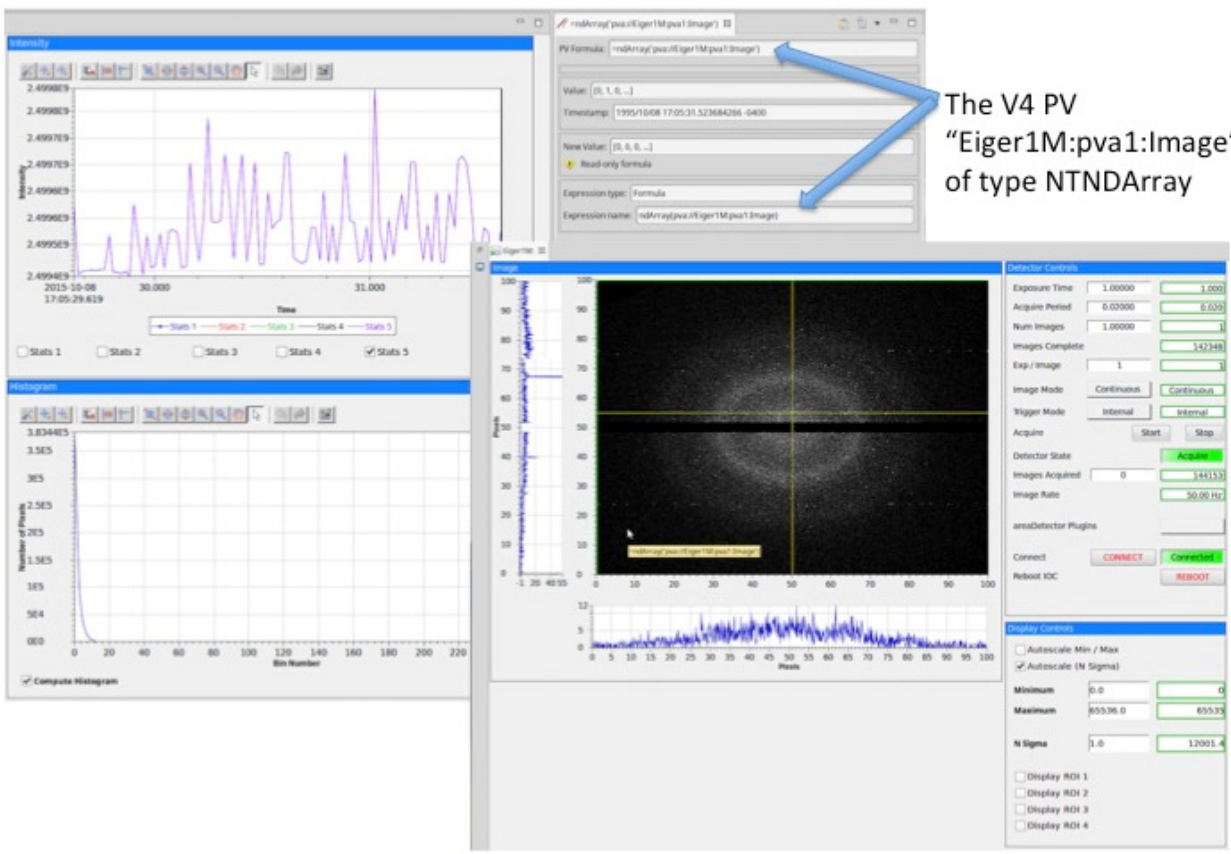


Figure 4: CS-Studio screenshots from NSLS-II, showing an EPICS V4 PV of the Normative Type for areaDetector images (NTNDArray) displayed using a CS-Studio “formula.” *Thanks to Bruno Martins, BNL.*

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TOWARDS BUILDING REUSABILITY IN CONTROL SYSTEMS – A JOURNEY

Puneet Patwari, Amar Banerjee, G. Muralikrishna, N. Swaminathan, Subhrojyoti Roy Chaudhuri
Tata Research Development and Design Centre, Pune 411013, India

Abstract

Development of similar systems leads to a strong motivation for reuse. Our involvement with three large experimental physics facilities led us to appreciate this better in the context of development of their respective monitoring and control (M&C) software. We realized that the approach to allowing reuse follows the onion skin model that is, building reusability in each layer in the solution to the problem. The same motivation led us to create a generic M&C architecture through our first collaborative effort which resulted into a fairly formal M&C domain model. The second collaboration showed us the need to have a common vocabulary that could be used across multiple systems to specify respective domain specific M&C solutions at higher levels of abstraction implemented using the generic underlying M&C engine. This resulted in our definition and creation of a domain specific language for M&C. The third collaboration leads us to imagine capturing domain knowledge using the common vocabulary which will substantially further reuse, this thought is already demonstrated through a preliminary prototype. We discuss our learnings through this journey in this paper.

INTRODUCTION

Monitoring and control systems are central to the working of projects such as SKA[1], ITER [2] and so on. These projects incorporate wide variety of heterogeneous systems and subsystems which require supervisory controllers for coordination. Our involvement with these projects gave us opportunity to learn and understand the kind of challenges involved in building monitoring and control solutions for such systems. One of our key observations is that these projects do reuse a lot of artefacts for the purpose of final implementation of their control systems. However, they still incur a huge amount of cost due to the effort they spend in the initial phases of the development life cycle. We noticed that this effort could also be substantially reduced since it showed large commonality in the type of activities taking place in each phase of their development life cycle.

Motivated by this observation, we started to analyze the prospect of a generic M&C architecture. This led to the creation of a generic M&C design and a prototype to demonstrate it in the context of GMRT. The design was inspired by the data driven paradigm and resulted in identifying a set of engines that could configure themselves based on the supplied input data that described the problem context. This approach enabled capturing the abstract model behind this input data eventually serving as the generic domain or specification model to capture the details of any M&C problem. Our first implementation realized parts of this model based on the format of the underlying execution engine which resulted into fragmentation and duplication of the M&C problem spec. This showed us the need for an integrated environment which could ensure integrity and consistency in the M&C problem specification. We recognized the need for a domain specific language (DSL) [3] to enable specification of any M&C problem so that the solutions created using the DSL could be analyzed independently. Our DSL work showed us the need for an environment which could be made aware of the application domain through its support for extensibility, analyzability, re-targetability and so on. We realized that such an environment would enable reusing a lot of domain knowledge which would enhance consistency in the entire M&C development process.

In this paper we start with a discussion on the current practice and challenges that motivated our research. Next, we highlight the proposed solutions adapted throughout our journey. Subsequently, we provide a view of our current implementation followed by the section which summarizes and concludes the paper with a futuristic view.

STANDARD PRACTICE AND CHALLENGES

Most projects start working on the requirement and design of their M&C systems from the scratch. As a result each project or groups within a project end up creating their own version of the concepts around a general problem domain such as M&C. This leads to some re-invention of concepts that are already created in another project. This point towards the lack of reusable artefacts except implementation packages across a problem domain that could enhance reusability in the entire development life cycle.

System engineering language such as SysML [4][5]provides a convenient way of expressing the designs in most of the projects. However, since much of the M&C concepts are not built into the vocabulary of SysML, it is common for different groups within projects to define the M&C vocabulary independently using SysML. Unfortunately such definitions are mostly not shared across groups. This leads to non-uniformity in the definition and usage of the M&C concepts across groups within projects. Hence it requires manual effort to

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

synthesize the design to create uniformity and consistency in the subsequent development phases.

The good news is, when it comes to implementation there exist a lot of reusable open source SCADA [6]packages such as EPICS [7], TANGO [8] that are popular across the scientific community.

Motivation

So it can be concluded from the discussion above, our primary motivation comes from the lack of a) reuse of domain knowledge to design M&C systems b) domain aware design analysis environment to support the development process.

PROPOSED SOLUTIONS

The course of our work can be broken into three major solution proposition milestones which are discussed below.

Inception of a Generic Architecture for M&C– SACE

Machine control systems are typically hierarchical, consisting of groups of devices at each level whose behaviour is orchestrated and coordinated by central controllers to achieve control objectives. Each device may contain actuators that perform actions on the environment, and sensors that determine the state of the environment. Thus the system can be imagined as a hierarchical network of sensors and actuators with feedback control at each level.

This led us naturally to postulate a generic architecture consisting of functionally identical nodes where the interfaces between hierarchical levels consist of three streams: commands, data and events. Each node sends commands and receives data and events from child nodes. Data and events are processed to develop the worldview. Control logic on the command path is responsible to achieve the control goals through discrete and continuous feedback control. This generic architecture was called Sensor Actuator Control Element (SACE) [9][10] and can be represented as the following figure.

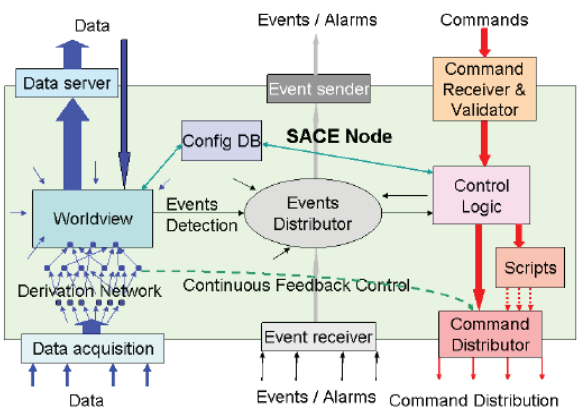


Figure 1: SACE node architecture.

The SACE node architecture incorporates the data-driven paradigm and implements each functional block using off the shelf solution components. Moreover, SACE architectural principles are driven by the requirements for efficiency, scalability and availability which are critical to machine control of scientific instruments.

The SACE prototype was initially created as a proof-of-concept inspired by the ITER architecture. This prototype was assembled by selecting freeware (and evaluation versions) of off-the-shelf technologies, and integrating them using Java. Each component was driven by configuration files, mostly in XML but in some cases had tool-specific formats. The approach of assembling off-the-shelf components made it possible to build the prototype quickly and effectively. The prototype successfully demonstrated the genericity of the architecture in the context of Giant Meterwave Radio Telescope (GMRT).

Although the prototype itself could be viewed as one more implementation of a generic M&C package, it revealed the abstract structure behind the input data which could serve as the generic domain model for M&C systems.

Creating Generic Domain Meta-model to Describe M&C Problem and Solution Design

Based on the results of the prototype discussed above, we decided to follow the Model-driven approach (MDA) [11][12]towards the definition of the M&C domain model and the creation of a domain aware environment to support its usage. This domain model would comprise of a vocabulary that can be mapped to the various concepts in the reference SACE architecture thereby establishing the correctness of the vocabulary.

The goal of this meta-modeling activity was to incorporate much of the standards from the M&C domain into the meta-model. Some elements and relations in the meta-model are custom defined due to the lack of corresponding standards e.g. terminologies such as ‘control node’ and ‘interface description’ and so on. But a good number of concepts implemented in the meta-model are borrowed from existing standards such as usage of state charts to capture behavior and so on. Since the concepts in the meta-model were derived based on the underlying modules of the SACE architecture, the meta-model also ended up becoming modular for example, concepts related to a common concern such as data acquisition generated inter relationships as opposed to the concepts belonging to a different concern such as control behavior. This also creates flexibility in making any change to a specific part of meta-model without affecting the overall structure. Figure 2 provides a glimpse of a part of the meta-model.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

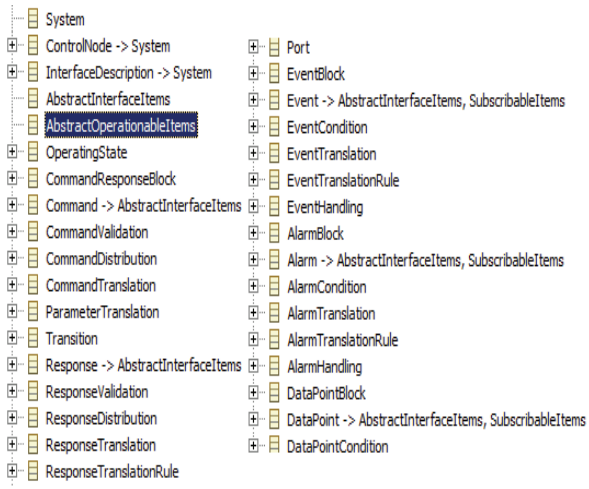


Figure 2: Meta-model: Serves as the M&C domain model.

Developing a DSL and Domain Intelligent Environment

The meta-model served as the starting point towards conceptualizing an environment that facilitates domain driven engineering to build M&C solutions. Figure 3 describes the architecture of the M&C specification environment or framework that captures an M&C problem and solution specification through the instantiation of the M&C domain model.

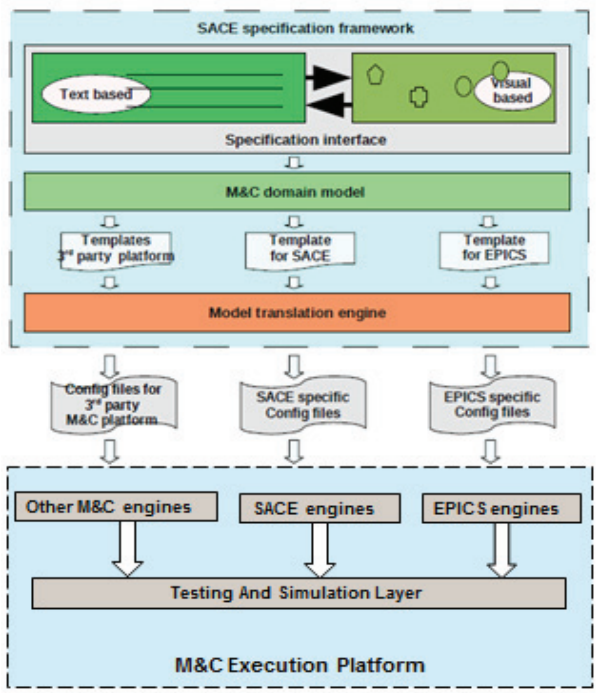


Figure 3: M&C Specification architecture.

As can be seen from the figure above, the top layer addresses the concern of a user interface to capture the M&C problem and solution specification. We implemented this layer using a DSL to capture the M&C solution details using the concepts defined by the specification model and named it M&CML [13]. The DSL helps to capture controller details such as its associated commands, responses, data streams, events, alarms, behavioral aspects like state machines, interaction with other subsystems, coordination logic and so on. The environment aims to provide support for the entire solution creation process: that is requirements, architecture, design decisions, validations and verification, creation of tests and so on. We used Eclipse based technologies such as EMF [14] [15] [16]and XText [14] [17] [18]that provide support for rapid development of DSLs following principles of MDA which suited our purpose. With XText, the various elements of the DSL along with their relationships need to be specified as grammar for the target language. Based on this, XText not only automatically generates a compiler for the target DSL but also generates along with it a complete environment to support the usage of this DSL. The environment provides built in support for user assistance, syntax highlighting, and input validation and also supports translation of the user written DSL to M&C target platform specific input formats. Since the resultant DSL environment is also a plugin in Eclipse, it allows access to other Eclipse based tools that could be leveraged for visualization or editing parts of the DSL in the future. The diagramming environment can be made complementary to the textual interface so that user could switch between textual and visual world whenever it is necessary and feasible.

The domain model is the heart of the specification framework. The domain model is implemented as a meta-model using Ecore of EMF technology. All the information captured using the DSL get populated in a meta-model instance. Hence, such specification files created independently across distributed teams can now be compared since they follow the same underlying structure. It is an effective way to bridge the gap of non-uniformity prevalent in the current approaches.

The model translation engine translates the DSL into target M&C platform specific code using model to model (M2M) transformation. This component is implemented using a combination of XTend which comes bundled with the XText tool-set and Atlas transformation language (ATL) [19] which is available as an independent plugin for Eclipse. With XTend, code generation templates can be written for each target M&C execution platform. Such code generation templates allow XTend to navigate an Ecore based M&C model instance and invoke translation logic to perform the required translation in a non-intrusive and pluggable fashion. This makes it possible for the translators to add incrementally to the framework allowing it to provide support for a wide variety range of M&C target technologies over a long period of time.

Last but not the least; the environment incorporates support to verify the solution created using this environment. Since much of the development of the controllers happen across teams in an isolated manner, the

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

verification of the solution becomes difficult due to the absence of the dependent systems. The testing and simulation layer of our environment provides a way to capture information about the absent systems so that their behavior could be simulated to verify the controllers which have dependency on such systems. Much of the information related to the creation of test cases could also be derived from the information present in the solution spec created using the DSL. The environment also incorporates separate meta-models to capture specific input related to testing and simulation. Based on this the environment is able to generate executable test cases and simulator implementation to verify the specified controller.

CURRENT PROTOTYPE ACTIVITY

Figure 4 shows the usage of our DSL to capture part of the M&C solution spec for GMRT.

```
Model GMRT
InterfaceDescription ID_IF{
    dataPoints{
        float IF_healthStatus = 10.0
    }
    commands{
        DOSET[
            parameter int bw1 = 1
            parameter int bw2 = 16
            parameter int alc1 = 0
            parameter int alc2 = 0
        ]
    }
    responses{
        RES_DOSET[
            parameter int DeviceNo = 10
            parameter int CmdStat = 0
            parameter string SETStr = ""
        ]
    }
    events{
        DOSET_Performed[]
    }
    states{
        Start[]
        Subscribe[]
    }
    commandResponseMap{
        command DOSET => expectedResponse RES_DOSET
    }
    commandEventMap{
        command DOSET => event DOSET_Performed
    }
    alarms{
        A1[ level : 5]
    }
}

ControlNode IF_Controller{
    Associated Interface Description : ID_IF
    statemachine IF_SM{
        state GMRT.ID_IF.Start[
            transitions
            event GMRT.ID_IF.DOSET_Performed => state GMRT.ID_IF.Subscribe
        ]
    }
    commandValidation{
        command GMRT.ID_IF.DOSET [
            parameter bw1 (Possible Values = {1, 16, 32}
            Max Value = 32 Min Value = 0)
            parameter bw2 (Possible Values = {1, 16, 32}
            Max Value = 32 Min Value = 0)
        ]
    }
    alarmConditions{
        AC2[
            dataPoint IF_healthStatus ( Min Value =5 )
            fireAlarm => GMRT.ID_IF.A1
        ]
    }
}
```

Figure 4: Sample DSL specification file.

GMRT is going through a system upgrade where they are moving from control system implemented using legacy software to a Tango based implementation. Since our environment already incorporates translators for Tango, it is envisaged that the upgradation process will get significantly augmented, since it will involve capturing the solution at a higher level and then automatically generate the implementation code specific to Tango requirements. Though, we have limited data to prove this.

Figure 5 shows a glimpse of the generated code adhering to TANGO standards and executable on the TANGO controls framework.

```
@Device
public class Controller {
    private static final Logger logger = LoggerFactory.getLogger(Controller.class);
    private static final XLogger xlogger = XLoggerFactory.getXLogger(Controller.class);
    private String className = this.getClass().getName();
    private static String deviceName = "nodes/Controller/test";
    private String parentDeviceName = "nodes/LMC/test";
    private String[] childDeviceName = {};
    /*Declaire Variables Here */

    /* Variables Declairation Ends */

     /*----------------Initialization Code--------------------*/

     @Init(lazyLoading = false)
     public final void initDevice() throws DevFailed{
        xlogger.entry();
        logger.debug("init");
@Command(name = "DOSET")
public synchronized  String DOSET(String parameters) throws DevFailed {
    System.out.println("Executing command DOSET ([mncModel.impl.SimpleTypeImpl@73a460 (name: bw1
    String inCommand = new CommandHandler().getCurrentMethod(this);
    if (new NodeCommandResponseValidation().validateCommand(inCommand,parameters)) {
String responseReceived = new ControllerSimulator().simulateResponse(inCommand, parameters);
    // Code To Be Written
    if (new NodeCommandResponseValidation().validateResponse(inCommand,responseReceived)) {
        // Calling EventDOSET_Triggered();
            return responseReceived;
    }else{
        return new String("BAD RESPONSE");
      }
    } else {
        return new String("BAD COMMAND");
    }
}
```

Figure 5: Generated TANGO specific code.

Since the environment also supports testing of the created solution, we believe this will add value to the testing and verification of the GMRT controllers significantly as well.

Figure 6 shows code for an automatically generated simulator produced by our environment and figure 7 shows a snapshot of a test run.

```
public class ControllerSimulator {
    String parseResponse = "{\"DOMON\":[{\"RES_DOMON\":[{\"allowedValues\":\"1,3,5,7,9,\
        String parseDataPoint = "{\"IF_Monitoring_HealthStatus\":{\"allowedValues\":\"10
        String deviceName = "nodes/Controller/test";
        JSONObject dataPointJsonObject = (JSONObject) JSONValue.parse(parseDataPoint);
        JSONObject jsonkk = (JSONObject) JSONValue.parse(parseResponse);
    Object object[] = dataPointJsonObject.keySet().toArray();
    public String simulateResponse(String commandName, String commandParameters) {
        JSONArray jsonArr = (JSONArray) jsonkk.get(commandName);
        if(jsonArr==null)
        {
            return "RESPONSE RECEIVED FOR "+commandName.toUpperCase();
        }
        JSONObject mlk = (JSONObject) jsonArr.get(0);

        JSONObject jso =  (JSONObject)JSONValue.parse(commandParameters);
        if (jso != null) {
        JSONObject fixedResp = (JSONObject) jso.get("fixedResponse");
        if(fixedResp!=null)
        {
        Set<String> map =  fixedResp.keySet();
        Iterator<String> iterat = map.iterator();
        String hh = fixedResp.get("Response")+":-";
        while(iterat.hasNext())
        {
            String mm = (String) iterat.next();
            if(!mm.equals("Response"))
            {
            hh= hh+ mm+":"+fixedResp.get(mm)+"||";
            }
        }
        System.out.println(hh);
        return hh;
```

Figure 6: Generated Simulator Code.

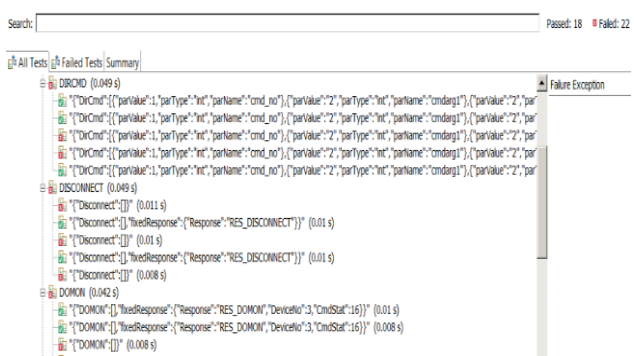


Figure 7: Generated test report.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

LEARNINGS, CONCLUSION AND FUTURE WORK

Our earlier work related to the generalization of the M&C architecture now has found multiple applications for GMRT and SKA and this gives us confidence that it is possible to build reusability in the early development life cycle of M&C systems.

Although the MDE approach has been around for a while now, its application towards solving problems such as ours needs to be carefully thought through. There is always a possibility for the meta-model to become very complex very quickly. An important guidance towards the definition of the model is based on explicating concepts of the underlying architecture and seeing how much of it requires user interaction and specification. The technological support made available by framework such as XText makes is possible to build rich DSL's environments with overall support for user assistance, verification and validation and retarget-ability.

Taking this approach forward, we see the possibility of incorporating support for other aspects such as testing and verification of the developed design during the design phase itself. We foresee the possibility of explicating the application domain knowledge in an executable form so that they could be plugged into this environment incrementally providing more support towards guiding the development of M&C systems for application areas such as Radio Astronomy and so on. This could only be achieved through making this environment highly extensible and flexible which remains our endeavor.

ACKNOWLEDGMENT

We acknowledge all members of the GMRT team, members from the TM consortium and members of the ITER CODAC team for their collaboration that made it possible for us to pursue this work.

REFERENCES

[1] SKA project: https://www.skatelescope.org
[2] International Thermonuclear Experimental Reactor: https://www.iter.org
[3] Johan Den Haan, "DSL Development: 7 recommendations for Domain Specific Language design based on Domain-Driven design", http://www.theenterprisearchitect.eu/blog/2009/05/06/dsl-development-7-recommendations-for-domain-specific-language-design-based-on-domain-driven-design/
[4] Alan Moore, Sanford Friedenthal, Rick Steiner, "A Practical Guide to SysML"
[5] OMG SysML : http://www.omgsysml.org/
[6] Office of the Manager National Communication System, "Supervisory Control and Data Acquisition (SCADA) Systems". National Communications System, (October 2004).
[7] EPICS: http://www.aps.anl.gov/epics/
[8] TANGO: http://www.tango-controls.org/
[9] S. Roy Chaudhuri et al, "Integrated Monitoring and Control Specification Environment", ICALEPCS (2013)
[10] S.R.Chaudhuri, Swami N, Amrit Ahuja., "Model-driven development of control system software", in The Low Frequency Radio Universe, conference at NCRA-TIFR, Pune, Page 384(2008).
[11] Sebastien Gerard, Jean-Philippe Babau, Joel Champeau, "Model-driven engineering for distributed real-time systems", Wiley-ISTE; 1 edition, (April 5, 2010).
[12] OMG Model Driven Architecture: http://www.omg.org/mda/ (2008).
[13] P. Patwari et al, "M&C ML: A modelling language for Monitoring and Control Systems", IAEATM (2015) (Under Review)
[14] Eclipse Modeling Framework: http://www.eclipse.org/modeling/emf/
[15] F. Budinsky, D. Steinberg, R. Raymond Ellersick, E. Ed Merks, S.A. Brodsky, T.J. Grose, "Eclipse Modeling Framework", Addison Wesley (2003)
[16] Sebastian Zarnekow, Sven Efftinge: "Model Driven Software Development with Eclipse", Java User Group Hamburg, (November 2009).
[17] Lorenzo Bettini, "Implementing Domain-Specific Language with XText and XTend", (August 2013)
[18] XText: https://eclipse.org/Xtext/
[19] Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I, "ATL: A model transformation tool." Science of Computer Programming 72(1/2), 31–39 (2008)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

OPEN SOURCE CONTRIBUTIONS AND USING OSGI BUNDLES AT DIAMOND LIGHT SOURCE

M. Gerring, A. Ashton, R. Walton, Diamond Light Source, Oxfordshire, UK

Abstract

This paper presents the involvement of Diamond Light Source (DLS) with the open source community, the Eclipse Science Working Group and how DLS is changing to share software development effort better between groups. The paper explains moving from product-based to bundle-based software development process which lowers reinvention, increases reuse and reduces software development and support costs. This paper details specific ways in which DLS are engaging with the open source community and changing the way that research institutions deliver open source code.

INTRODUCTION

Diamond Light Source [1] is a third-generation 3 GeV synchrotron light source based on a 24-cell double-bend achromatic lattice of 561m circumference. The photon output is optimised for high brightness from undulators and high flux from multi-pole wigglers. The accelerators and first phase of seven photon beamline were constructed from 2002 to 2007; a second phase of fifteen photon beamlines from 2006 to 2012; and a third phase of ten photon beamlines was approved in 2011 with construction due to finish in 2017-8.

As well as the construction of the synchrotron, the early phases of the project saw choices about the software which would be deployed on site. For hardware control such as motors, the EPICS framework was chosen which included a data driven user interface called EDM [2] for configuring devices. The acquisition and online data analysis system was developed from a product in operation at the SRS [3] called GDA [4] previously presented at ICALEPCS.

Diamond Light Source (DLS) have switched GDA (client) and a standalone analysis product called DAWN [5] to load using a system called OSGi (Open Service Gateway Initiative) [6]. In addition, the EPICS/EDM screens are planned to be phased out of active support [7] in line with the move to RHEL7. The next generation of software for controls and acquisition is based on Eclipse Rich Client Platform (RCP). This allows software products to be built from OSGi bundles and features to be developed which are interoperable between controls, acquisition and data analysis software. This paper details how DLS interacts with open source technology to deliver feature rich, interoperable and reusable software systems across groups and in the wider community.

OPEN SOURCE 'ECOSYSTEM'

Eclipse RCP [8] is a software technology which has been available for more than a decade. It is used to build user interface applications and OSGi servers. DLS are utilizing it as a platform to deliver native user interface clients. One of the first RCP clients in production for acquisition was on the B18 beamline, presented at ICALEPCS 2011 [9].

The Eclipse Foundation is also an open source publisher which verifies open source code for intellectual property (IP) and software license. Its rigorous IP process [10] renders code safe for commercial companies and institutions to reuse at reduced risk of litigation. It provides a rich open source feature set, similar to that on offer by the Apache Foundation. This 'ecosystem' of IP checked bundles has provided many useful features which DLS have been able to reuse within software products.

In 2014 DLS proposed an Eclipse project [11] to make aspects of its DAWN product open source and IP checked by the Eclipse Foundation. This project was granted and IP checking is active, nearing completion at the time of this conference.

How Open Source Works

The procedure in scientific institutions active in open source software release has often been to provide source using GNU Public License (GPL). Various mechanisms have been used to do this for example: zip file on an ftp site, by implementing a web site using a technology such as Redmine or by using an open repository site like github. These approaches are now considered unsafe however because the GPL can force institutions to release previously unready or non-public code. More importantly, the source code has often not been IP checked before it is released. Foundations such as Apache and Eclipse provide an IP checking service. The service gives confidence to the copyright holder that they have provided something for which they have a lower risk of litigation. The source code of the software is also more likely safe to be re-used. This promotes wider use and contributions from outside collaborators.

ECLIPSE SCIENCE WORKING GROUP

DLS, Oak Ridge National Laboratory [12] and IBM have formed an Eclipse Science Working Group (SWG) in conjunction with ten other members ranging from small contractors to large commercial companies. The SWG members have started projects such as Triquetrum, Chemclipse and Integrated Computational Environment (ICE), links to which can be found on the web site [13].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The SWG also propose projects such as Eclipse Rich Beans and Eclipse Advanced Visualization. The SWG are working together on interoperable features and to deliver novel software solutions. A key area identified to allow interoperability has been the description and mathematics of n-dimensional data. In order to address this, DLS have produced an API of two OSGi bundles. These bundles are similar in scope and functionality to numpy [14] for python. The SWG have identified data description important to making future software features interoperable/reusable and propose projects planned to reuse this work in other fields.

SOFTWARE IN BUNDLES

Traditionally software has been created using designs which are non-modular at runtime – all of the program has to be in memory for it to run. In the Java world, this resulted in long start up times because a large "classpath" of jar files (compiled code) had to be resolved while software started.

OSGi bundles make code modular and low dependency. They have a feature known as declarative services which export functionality to other bundles without making hard dependencies. This means that software does not have to fully be in memory when it is run and so has fast start up times. Without hard dependencies, the architecture can load features incrementally as they are required by the user. It also means that software is made formally modular making it easy to reuse within different products because its requirements for compilation are low and well defined. The declarative services approach results in lower software development and support costs.

DLS have divided features into OSGi bundles for many features, for example: data including nD mathematics, file loading including HDF5, plotting and visualization, user interface widgets and auto generation, communication and devices, interacting with remote data, hardware configuration and mathematical processing pipelines, see also Table 1. This approach is leading to a swap in orientation from developing products to developing bundles. Management and support engineers are then free to define products based on the needs of the synchrotron users, software developers can more easily work together on multiple products and features can be reused in multiple areas.

CASE STUDIES

Data Format

A first common problem encountered by institutions running large experimental physics control, is the format in which data is written. Then once data is read into memory from a given format, a second problem is that data needs to be described in a standard way. Various frameworks have existed which deal with these problems at different institutions and in different programming languages.

DLS software developers took the approach that the details of the file format and the way it is loaded should be hidden from the part of software where the data is used. An OSGi service was created for loading data which supports a wide range of data formats such as HDF5 incl. NeXus, CBF, ASCII formats and most standard image formats. However it is also extendible in the compiled product by users. This is available because an extension point in the RCP framework has been created whereby the application may be added to after compilation and deployment. This approach means that any data format can be supported, providing a reader for it may be written.

```
ILoaderService service = …  // OSGi
File file = new File(…);
IDataset d = service.getDataset(file, …)
```

Figure 1: Loading data by service.

These lines of code read one dataset from a file and return an object similar to a numpy array, called IDataset. This object is the key to the second problem. (A comparison of MATLAB, numpy and IDataset is available in the DAWNSci Project examples.) It means that any application, written in Java, can load and share the data of multiple formats and create tools which use the data. The mathematics of the tools can be reused because of the common data format. The user interface can also be reused if the chosen platform is SWT. In addition a non-RCP application is not precluded from using the data layer, including non-OSGi based frameworks. The feature is a standard Java 'jar' file.

Plotting

The ability to plot data and interact with visual tools has been designed as an OSGi service. For instance all the plot operations, generic tools like peak fitting or integration of different regions and science specific tools like diffraction experiment ring fitting. This plotting service allows a plotting system to be created and data to be plotted with the proviso it is described as an IDataset.

```
IPlottingService ser = …  // OSGi
IPlottingSystem ps = ser.createPlottingSystem()
ps.createPlotPart(…)
ITrace trace = ps.createImageTrace(…)
trace.setData(d)

// Other config like name, colour map etc. then
ps.addTrace(trace)
```

Figure 2: Plotting an image.

Visual Tools are available to the user once the plotting system has been created, automatically. The plotting system comes with a wide range of tools which are extendible, again via extension points. The data rank plotted is passed to an underlying tool system which determines the available tools for that rank, see Figure 3 and 4. Tools may be chained together.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

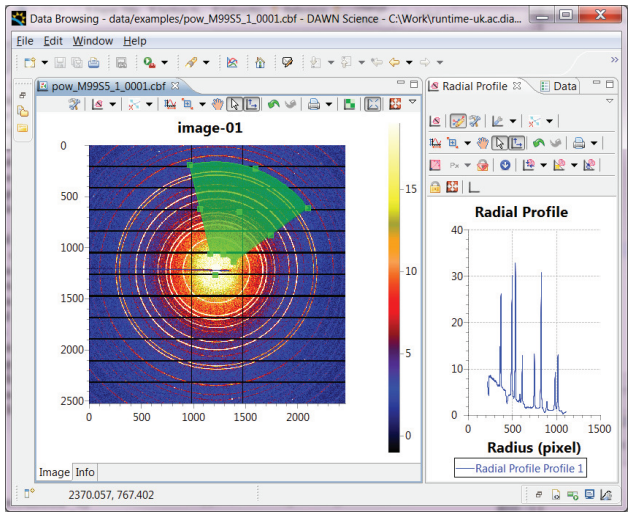


Figure 3: Using a single tool for radial profile

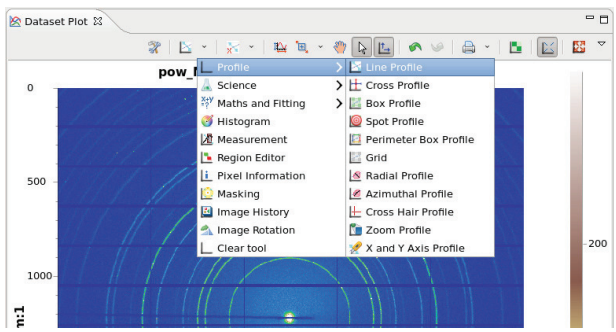


Figure 4 Some of the available tools for 2D data.

Regions of Interest are supported by the plotting system enabling a variety of shapes useful for data analysis to be drawn. The region system is integrated with the plotting system and uses a standard factory pattern (rather than extension points) to define available regions.

Multiple Viewers plug into the plotting system, contributable by extension point. For instance the system abstracts the plotting of lines from that of isosurfaces and uses different underlying user interface toolkits, Draw2D and JavaFX respectively, to render them.

Python Extensibility is supported by the plotting system. It is extensible for those that can write Java because of the hooks into it by extension point. However many users of the applications produced at DLS are not familiar with Java. Therefore the plotting system API is available to Python programmers via a plotting to python link. It is possible to either use simple factory methods for plotting numpy ndarrys or to get a reference to the actual plotting system Java object and make calls on to it, using a technology called py4j [14].

Streaming Data is supported by the plotting system. It can be connected to streams and data from remote file systems using a feature called Remote Dataset developed at DLS and deployed over several bundles. It is based on a servlets running in a Jetty server. In this case the data object on the client must implement an interface to mark it as dynamic and optionally remote. This dataset can be an MJPEG stream or a NeXus file on another file system, for instance. The plotting system will then update live feeds and the tool system is designed to work with live data. This works by using the Eclipse Job system to complete the tool mathematical operations in a separate thread and using a queue.

Reuse of the plotting system bundles are in the GDA client and DAWN products at DLS. It is also available in a product running at ISIS. It is being investigated for use with plotting of data from Control System Studio at DLS because of features like the tool system, python connectivity, streaming and ability to deal with many plot viewers.

Malcolm

DLS have a prototype project designed to abstract configuration and running of devices. The focus of the project, called 'Malcolm', is to deliver a way to configure and run Zebra devices [16]. It exposes a configurable device via a port to any client application via objects encoded into JSON strings. DLS have created a set of OSGi bundles for interacting with Malcolm devices which expose them as an OSGi service, see Fig 5.

```
IMalcolmService ms = …  // OSGi
IMalcolmConnection c = ms.createConnection(…)
IMalcolmDevice device= c.getDevice(…)
device.configure(…)
device.run()
```

Figure 5: A service to configure devices.

A **State Machine** has been designed to control Zebra but it also allows any hardware to be integrated. The well-defined states ensure that a programmer using the device has a clear idea of how to drive it, regardless of the details of the underlying hardware, see Fig 6.

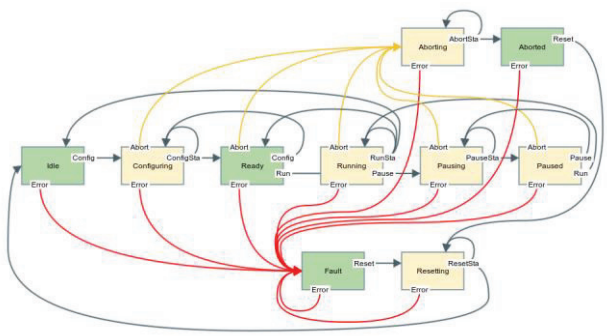


Figure 6: Design of Malcolm state machine.

The service is implemented using a connection to a python program which manages sending commands to the Zebra device and status to the user of the Malcolm state machine. This approach has allowed a software layer to sit close to the device and expose a well-defined way of using that device so reducing redevelopment of user interface and other features for similar devices.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Other Services

DLS would like to reuse more services between applications and have created more, which it hopes to reuse in the future between applications. Some of the more important are in Table 1.

Table 1: Some Useful Services Available from the Framework.

Service	Description
Conversion	Convert files from any format in which can be loaded to any which may be written.
Operation	Allows the programmer to create a processing pipeline from a large library of mathematical operations and run the pipeline over large image stacks on clusters.
Persistence	Save data, meta-data, masks and regions to a persisted NeXus file which can be loaded later and reimported.
Macro	A service which maps user interface actions with their python equivalent and allows the user interface to print macro commands into a running terminal while using the user interface.
Expression	A service to evaluate expressions. For instance the programmer may enter string expressions of datasets in expression language similar to python and evaluate them.

CONCLUSION

Diamond Light Source software developers have and continue to engage with the open source community. Best practice for IP checking code and releasing code is being adopted to ensure that it can be safely contributed for reuse and safely extended by external developers. New open source projects have been created which allow reusers of software produced at the synchrotron to take advantage of features developed. A new Eclipse Science Working Group has been formed where new ideas for projects and APIs can be discussed and the more interesting ones tried out. Many of the internal features developed at DLS are being released as bundles which contribute services. This is decoupling the software at DLS - making it more modular, easier to reuse features between applications and cheaper to support.

ACKNOWLEDGEMENT

The members of the Eclipse Science Working Group, large and small whom have helped create a positive environment for discussing and sharing code. Thanks to the many contributors to RCP products at DLS, especially DAWN and its external contributors.

REFERENCES

[1] R. P. Walker, “Commissioning and Status of The Diamond Storage Ring”, APAC 2007, Indore, India.

[2] John Sinclair, “EDM: Extensible Display Manager for EPICS”, USPAS 2003

[3] V.P. Suller, “Performance of the Daresbury SRS with an Increased Brilliance Optic”, EPAC 1988

[4] Generic Data Acquisition, www.opengda.org

[5] M. Basham, “Data Analysis WorkbeNch (DAWN)”, J. Synchrotron Rad. (2015). 22, 853-858

[6] O. Alliance, “Osgi service platform, release 3” IOS Press, Inc. 2003

[7] M. Furseman, “Adopting and Adapting Control System Studio at Diamond Light Source”, ICALEPCS 2015, Melbourne, Australia

[8] Eclipse - www.eclipse.org

[9] R. J. Woolliscroft, “Quick EXAFS Experiments Using a New GDA Eclipse RCP GUI with EPICS Hardware Control”, ICALEPCS 2011, Grenoble, France

[10] www.eclipse.org/projects/dev_process/ip-process-in-cartoons.php

[11] DAWNSci projects.eclipse.org/proposals/dawnsci

[12] ORNL www.ornl.gov/

[13] Eclipse Science Working Group - science.eclipse.org

[14] NUMPY www.numpy.org

[15] PY4J www.py4j.org

[16] T. Cobb, “Zebra: A Flexible Solution for Controlling Scanning Experiments”, ICALEPCS 2013, San Francisco, USA

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

QUASAR - A GENERIC FRAMEWORK FOR RAPID DEVELOPMENT OF OPC UA SERVERS

P. P. Nikiel, B. Farnham, S. Schlenker, C.-V. Soare, CERN, Geneva, Switzerland
V. Filimonov, PNPI, Gatchina, Russia
D. Abalo Miron, University of Oviedo, Spain

Abstract

This paper describes a new approach for generic design and efficient development of OPC UA servers. Development starts with creation of a design file, in XML format, describing an object-oriented information model of the target system or device. Using this model, the framework generates an executable OPC UA server application, which exposes the per-design OPC UA address space, without the developer writing a single line of code. Furthermore, the framework generates skeleton code into which the developer adds the required target device/system integration logic. This approach allows both developers unfamiliar with the OPC UA standard, and advanced OPC UA developers, to create servers for the systems they are experts in while greatly reducing design and development effort as compared to developments based purely on COTS OPC UA toolkits. Higher level software may further benefit from the explicit device model by using the XML design description as the basis for generating client connectivity configuration and server data representation. Moreover, having the XML design description at hand facilitates automatic generation of validation tools. In this contribution, the concept and implementation of this framework named `quasar` (acronym for **q**uick OPC **UA** **s**erver gener**a**tion f**r**amework) is detailed along with examples of actual production-level usage in the detector control system of the ATLAS experiment at CERN and beyond.

INTRODUCTION AND MOTIVATION

Distributed control systems require middleware – software which transfers data between system components. The ATLAS Detector Control System (DCS) [1] is an example of such a distributed control system, organized as a hierarchical mesh of heterogeneous components. The middleware must be capable of handling numerous data models, while being portable and performant at the same time. For the ATLAS DCS, OPC Unified Architecture (further on: UA) [2] has been selected as its new standard middleware [3] for device integration mainly due to its object oriented design and platform independence. A common approach to create UA servers for the various device types allows to reduce development and maintenance costs.

Apart from obvious common functionality in which identical software parts were identified (such as server startup code, logging implementation etc.), it became evident that development efforts could be largely reduced if the data model was considered a parameter of a generalized UA server. Such a data model, augmented with additional information, is subsequently called *design*. If the format of the design is sufficiently rich to describe and model (potentially complex) subsystems, big parts of an UA server implementation may be automatically created (generated). Thereafter, hand-written custom code is only necessary for providing high level 'business logic' between the generated parts and the handling of the specific subsystem type (e.g. a hardware access library or protocol implementation). Such hand-written code may be very complex depending on its functionality requirements. We chose to call this code *device logic*.

In the following sections we explain the approach of generating UA servers from the preparation of a server design up to obtaining a functional application.

QUASAR ARCHITECTURE

Figure 1 gives an overview of the different layers of `quasar` put into context. Controllable devices or systems are accessed using their specific access layer – often provided together with the specific device. The device logic layer functions as interface with the high level layers of `quasar` which comes in several modules covering different functionality aspects. The address space module lies on the UA end of the server, exposing data towards UA clients, and is implemented using a commercial UA SDK [4]. A configuration module facilitates address space and device instantiation and the definition of their relations. XML is used as configuration format backed by XML schema definitions. A XML schema to C++ mapping generator (here: `xsd-cxx`) is used to build actual instances from configuration files. An additional subsystem called 'calculated items', operating entirely in the address space, enables creation of new variables which are derived from existing ones using mathematical functions. `quasar` comes further with optional modules such as component based logging, certificate handling, server metadata and embedded python processing.

MODELLING DEVICES OR SYSTEMS

In the generic approach of `quasar`, an object oriented model was chosen for two reasons: object orientation is well known and widely understood, and UA itself follows the object orientated paradigm. The purpose of modelling is to establish a comprehensive device or protocol characterization using classes, variables, methods and the relations between them. Classes are types of particular objects. Variables belong to classes and are factual vectors of data while class methods process the associated data. The purpose of relations is to model aggregations and type hierarchies.

Once the model is prepared, it has to be codified in a common format – we call this the *design file*. `quasar` uses the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

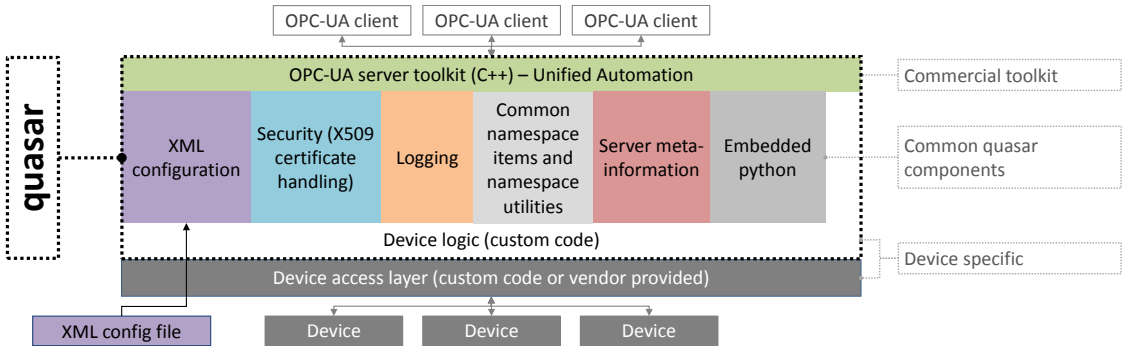


Figure 1: Overview of the quasar components.

XML format for its designs and comes with a ready made XML Schema allowing for easy design creation with commonly available XML editors. A format was required which parametrizes the generic server such that specific instances (design files) match particular subsystems/device types. The most important ingredient of such a design file is the data model handled by the server (discussed in the previous section) augmented with processing-specific attributes.

The first mandatory attribute is capturing the difference between distinct classifications of variables:
Cache Variable: the factual data resides in RAM. Since accessing RAM-based data is a primitive operation for computers, get/set/monitor are just trivial operations handled behind the scenes by the UA toolkit and UA stack.
Source Variable: the factual data resides in undisclosed location (might not even be RAM-based and for our use cases it typically was outside given server computer). An access interface is necessary to read or modify such data, for which glue logic has to be coded in the server. Moreover accessing such data may be a very time-consuming process and often has to be executed in separate thread of execution. Note that these classifications are not seen from the perspective of an UA client, all variables are simply queried (using write/read UA transaction types) or monitored (by creating a monitored item for this variable). However at the UA server implementation side it is beneficial to differentiate between classifications. Another important design attribute is the handling of concurrence inside the UA server. quasar provides capabilites to model domains of mutual exclusion to prevent race conditions in case two objects were to be accessed at the same time.

Finally, the configuration of objects needs to be specified. By configuration we understand a set of values that do not change between creation of an instance and its deletion. The primary configuration parameter is an unique object name, allowing e.g. to traverse the hierarchy of objects using dot as a separator. Additionally, objects often require additional configuration by assigning values to specific attributes, either constants or e.g. for initialization purposes. This is also handled in the design stage through items called “config entries”. Config entries belong to classes.

In summary – the design represents the description of types while the configuration is the description of instances.

DESIGN TRANSFORMATION

quasar generates a number of distinct elements based on the server design:
- source code (mostly C++),
- dependent XML schemas,
- design-dependent parts of the build system,
- visualizations of the object structure,
- UA address configuration for quick integration into higher level control system layers (e.g. SCADA system),
- additional utilities (e.g. for testing the address space).

Almost all of these tasks are achieved by XSLT transforms, either to text output (e.g. C++ code) or to XML (e.g. XML schema). Figure 2 illustrates these transformations.

The full procedure of creating an UA Server using the framework is as follows:

1. Create or modify the design.
2. Request creation of device logic stubs for classes defined in the design (initially empty stub implementations are generated, thereafter existing implementations are merged with respect to the new design).
3. Extend device logic stubs by providing factual implementation.
4. Build the server (e.g. by using the provided CMake based build system).
5. Develop by re-iterating the steps 1 → 4.

In the following, the individual transformation steps performed by quasar are detailed:

Generation of Address Space C++ classes: For each class in the server design, one C++ class is generated, conforming to the interface provided by the UA Toolkit (therefore instances of these classes can be directly “injected” into the server address space). For every variable of the class, appropriate setters, getters and/or write/read handlers are generated; this ensures that code outside of the Address Space module (typically, hand-written code in the Device Logic module) can interface with the address space class using straightforward C++ function calls.

Information model: UA has rich modelling capabilities from which smart UA clients may profit. quasar exposes

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

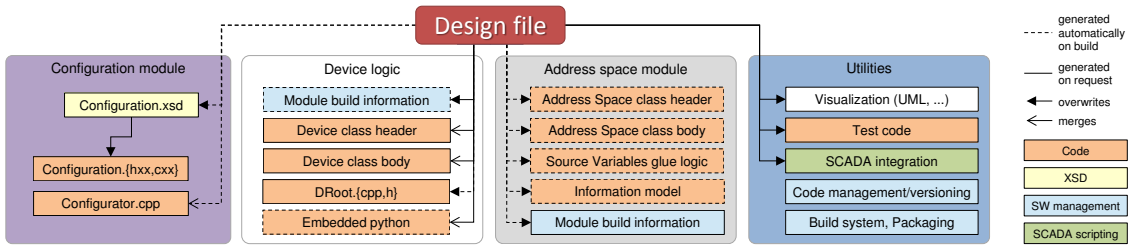


Figure 2: Transformation diagram illustrating the software generation or modification process for a given design file.

the information model derived from the design both in the generated code and within the UA address space at server runtime.

Generation of the configuration XSD schema: Each class from the design is transformed into a complexType in an XSD schema while aggregation relations between classes are respected (e.g. complexType may have a sequence of elements of a different class). Moreover, config entries specified in the design which provide configuration data to specific instances become attributes within the given relevant complexType.

Generation of the configuration loader: C++ code is generated to handle parsing a given XML configuration file to create instances of objects at the startup of the server. This code builds on top of code generated using xsd-cxx with Configuration schema as a parameter. Furthermore, an additional validator is generated to check for constraints that are not easily explained through Configuration schema but easy to explain through server's design.

Device Logic transformation: The Device Logic is the only `quasar`-provided module in which the developer is expected to write C++ source code – the device specific implementation – starting from the generated stubs. The stubs contain skeleton classes and methods according to the design along with an interface to the corresponding address space items. As development progresses, the design may be fluid; classes may be added or removed; variables may change type or perhaps are suppressed completely. After a change of the design file, the developer re-runs the Device Logic generation. If user-modified class sources exist already, the hand-written code will not be overwritten – a merge tool opens to facilitate the adaption of code to the new design.

SCADA integration code transformation: Additionally, `quasar` may generate tools which let the server be easily integrated into some SCADA systems. The typical use case at CERN is as follows: scripts are generated which allow creating the corresponding data structures and their UA addresses within the SCADA system (here: Siemens WinCC OA). This step may not be necessary if a SCADA is used which provides information model aware UA clients.

ADDITIONAL TOOLS FOR DEVELOPERS

Significant effort has been invested to avoid duplication of work for `quasar` users. Thus a number of tools are provided, helping to carry out the following tasks:

- Visualizing object structures: an UML-like diagram creator is provided which helps to visualize the design.
- Validating and upgrading design files.
- Managing consistency of source files: a tool is provided which ensures that source files are properly versioned and that certain files (e.g. XSLT transformations) are not accidentally modified.
- Building binaries: a build system based on CMake is provided along with pre-configured toolchains for several platforms such as `x86_64` or ARM-based Linux and Microsoft Windows.
- Creating installers: a preconfigured spec file for creating RPM packages.
- Testing the address space: a tool is added which continuously pushes random data into the address space. This can be used e.g. to test client mappings and server/client performance under specific conditions.

EXAMPLES

At the time of writing, `quasar` had already been used to create 11 different UA server implementations which cover numerous use cases, applications and various subsystems which are interfaced. These servers are currently in production use at CERN (in numerous instances). We briefly discuss two of these servers – the VME crates server and SNMP server – as their architecture and use case are very different demonstrating the flexibility of the `quasar` approach.

VME crates server: The VME crates UA server implements full monitoring and control of specific VME crates using a proprietary, polling-based communication protocol [5] via CAN bus interfaced to a COTS rack server of the control system. The object model of the design is defined by CAN buses formed by a chain of VME crates which in turn contain channels, fans and temperature probes, c.f. the `quasar`-generated class hierarchy diagram (Fig. 3). Each of these elements (buses, crates, etc.) is to be hierarchically declared in the configuration file allowing a variety of configurations, from very simple single-crate systems up to multi-bus, multi-crate systems with thousands of channels.

From the hardware point of view, each crate has a communication module which has to be polled for the status of the crate itself and its channels and sensors. From the device logic point of view, a software entity at the server side, called communication controller, manages communication between factual crates and the device logic of the UA server.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

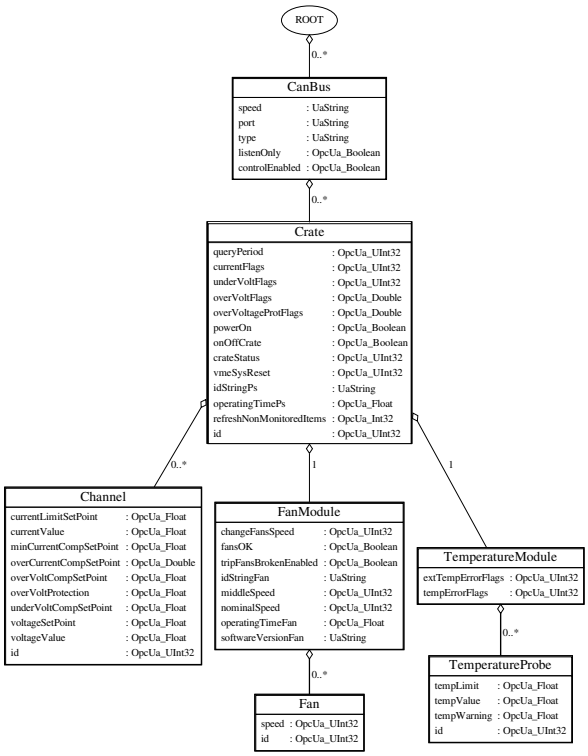


Figure 3: Generated design diagram of the VME server.

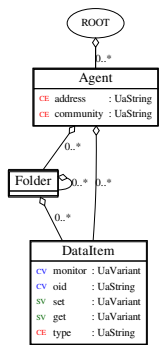


Figure 4: Generated design diagram of the SNMP server.

Even though the crate communication protocol is polling based, the software of the device logic is event based and as soon as new data arrives from a crate, it is pushed into variables of the UA address space and then further to any subscribed UA client – in UA terms "monitored data". The server uses a “CAN interface” module – an optional `quasar` module – a hardware access library supporting CAN interfaces from numerous vendors. A second, server-specific module implements the specific VME crate communication protocol facilitating encapsulation of data in the server device logic. The VME crates server proved to work stably in production and is used in a wide span of configurations – the biggest being a system of 62 VME crates of the ATLAS trigger and data acquisition system with ~5000 channels for which the server CPU load and memory consumption is negligible on a modern server computer.

SNMP UA server: The SNMP (Simple Network Management Protocol) is an Internet-standard protocol for managing devices on IP networks. The server exposes a tree-shaped address space in which data items are basic building blocks (leaves of the tree). Figure 4 shows the design of the server: each data item is bound to one SNMP object identifier in the configuration. Each read/write request coming from an UA client is transformed into a get/set SNMP operation.

Compared to the previous example where cached data coming from the hardware is pushed into memory, this server makes extensive use of Source Variables, which assumes that only the data provider (here: SNMP agent) has the most up-to-date contents of variables. Since a SNMP transaction is a blocking operation (which, compared to in-memory access, may fail), appropriate processing has to be estabilished in order to avoid blocking the whole server while transactions are ongoing. `quasar` automatically generates the corresponding code: each SNMP transaction is spawning a job belonging to a thread pool and synchronization might be automatically applied depending on the synchronization attributes of each hierarchy level in the design file.

CONCLUSIONS

`quasar` has been already successfully used for the creation of a number of UA servers by non-expert developers which demonstrates its advantages: versatility and efficiency of development. The former has been proven by a vast span of applications, from custom devices to generic designs for well-known protocols. The efficiency of the development process becomes evident by the reduction of development efforts since up to 90% of source code can be generated which results in a lower chance of bugs being introduced and at the same time achieving better source code managability. `quasar` doesn’t add any overhead compared to classic server development using an UA toolkit only and thus no performance penalty is expected nor observed. Further exploitation of `quasar` at CERN and beyond is envisaged while its functionality being further expanded.

REFERENCES

[1] Barriuso Poy A, Boterenbrood H, Burckhart H J, Cook J, Filimonov V, Franz S, Gutzwiller O, Hallgren B, Khomutnikov V, Schlenker S and Varela F “The detector control system of the ATLAS experiment”, Journal of Instrumentation, Vol. 3, May 2008, doi:10.1088/1748-0221/3/05/P05006.

[2] The OPC Foundation, “OPC Unified Architecture”, http://opcfoundation.org/opc-ua/

[3] Nikiel P P, Farnham B, Franz S, Schlenker S, Boterenbrood H and Filimonov V “OPC Unified Architecture within the Control System of the ATLAS Experiment”, Proceedings of ICALEPCS2013, San Francisco, CA, USA, p 113-6.

[4] Unified Automation GmbH, “C++ based UA Server SDK”.

[5] W-IE-NE-R Plein & Baus GmbH, “CAN-BUS Interface for W-Ie-Ne-R Crate Remote Control”.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

DISRUPTOR - USING HIGH PERFORMANCE, LOW LATENCY TECHNOLOGY IN THE CERN CONTROL SYSTEM

M. Gabriel, R. Gorbonosov, CERN, Geneva, Switzerland

Abstract

Accelerator control systems process thousands of concurrent events per second, which adds complexity to their implementation. The Disruptor library provides an innovative single-threaded approach, which combines high performance event processing with a simplified software design, implementation and maintenance. This open-source library was originally developed by a financial company to build a low latency trading exchange. In 2014 the high-level control system for CERN experimental areas (CESAR) was renovated. CESAR calculates the states of thousands of devices by processing more than 2500 asynchronous event streams. The Disruptor was used as an event-processing engine. This allowed the code to be greatly simplified by removing the concurrency concerns. This paper discusses the benefits of the programming model encouraged by the Disruptor (simplification of the code base, performance, determinism), the design challenges faced while integrating the Disruptor into CESAR as well as the limitations it implies on the architecture.

INTRODUCTION

CESAR is the high level software used to control CERN experimental areas. The experimental areas are composed of eleven beam lines used by experimental physicists for fixed target research and detectors tests. Four beam lines are located in the East Area, using a beam extracted from the PS ring, and seven are in the North Area, using a beam extracted from the SPS ring. The core of CESAR is responsible for the data acquisition of all the devices controlling these beam lines. While refactoring this data acquisition part of Cesar, we decided to use the Disruptor library in order to simplify the design of the code handling the 2500 asynchronous event streams coming from these devices.

In the last decade, the actors of the world of finance and high frequency trading have been involved in an arms race to build exchanges and trading robots that can operate at the nanosecond scale. From time to time, some technologies created by the massive investments in this field are shared with the community [1].The Disruptor library was created by LMAX [2] -a London-based financial company- in order to develop a low-latency forex [3] trading venue [4]. In the early design phase, they tried different approaches: functional programming, Actors, SEDA [5], and noticed that they could not achieve the required latency because the cost of queuing was higher than the time spent executing the business logic. They finally settled on an innovative design and decided to open source it.

THEORETICAL BACKGROUND

The base idea around the Disruptor is to make the most of the available CPU resources, following a concept that its creators call ‘mechanical sympathy’. This term coming from the car racing world is used to describe software working in harmony with the hardware design, similar to a driver understanding how a car works in order to achieve the best performance. Since the appearance of multicore CPUs, we have heard expressions like “the free lunch is over” [6] and there is a general belief that CPUs are not getting any faster. Although their clock speed is not getting higher, modern CPUs have brought significant performance improvements. Regrettably, the progresses made in hardware are often lost by software designs that do not consider how modern processors work.

Feeding the Core

The most important aspect to consider in order to use a processor efficiently is to feed it correctly. Processors are very fast, but this speed is of little use if they spend most of their time waiting for the data they need to process. Looking at a simplified view of the memory architecture of a modern CPU such as Intel’s Sandy Bridge (Fig. 1), we see that the cores read data from several cache layers (L1, L2, and L3).

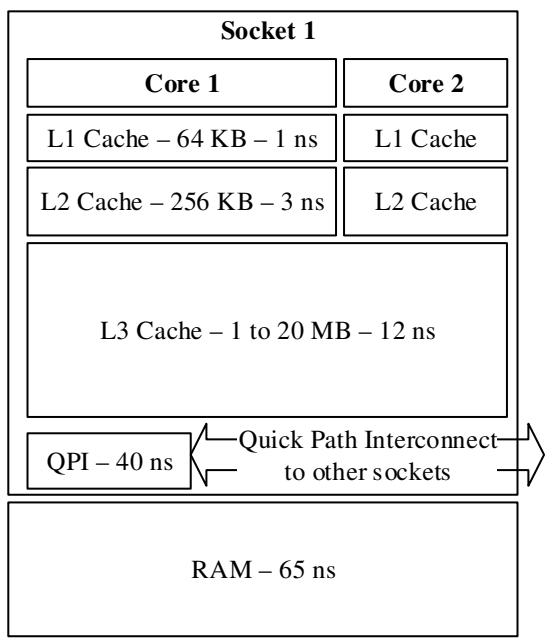


Figure 1: Memory Hierarchy

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

When a core needs to process data, it looks down the chain in the L1, L2 and L3 caches. If the data is not found in any cache, it is fetched from the main memory. As we physically move away from the core, each memory layer has an increased capacity, but is also several orders of magnitude slower than the previous one. In order to use the core at its maximum throughput, a program should work on data that is available within the caches.

Developers do not have to do this manually, because the processor is able to automatically prefetch the data in its caches. However, the prefetcher has a limitation: it only works if the memory is accessed with a predictable pattern: this means that the program has to walk through memory in a predictable stride (either forward or backward) [7]. This works very well while iterating on data structures that use memory allocated contiguously, such as arrays, but cannot be used with linked lists or trees because the prefetcher is unable to recognize an access pattern on these complex structures. This mechanism is not limited to one thread; on modern Intel processors, up to 32 streams of data can be prefetched concurrently.

Padding Cache Lines

Caches are composed of cache lines: these are memory blocks of fixed size (64 bytes on modern x86). Cache lines are the atomic units of memory used in caches: the prefetcher always loads full cache lines, and when a variable is written in a cache line, the full cache line is considered as being modified. The phenomenon known as 'false sharing' happens when two unrelated variables share the same cache line, and are written concurrently by two threads running on two different cores (Fig. 2). These threads constantly fight for the ownership of the cache line. Since each write of a variable results in the invalidation of the same cache line on the other core, the data needs to be reloaded. If the two cores are on the same socket, it can be reloaded from the L3 cache. If they are on different sockets, this battle is fought through the QPI bus (Fig. 1), adding even more latency.

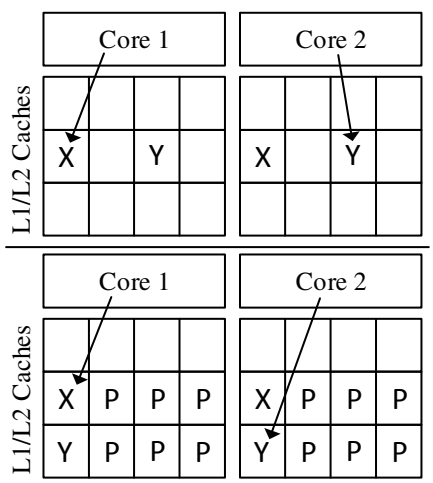


Figure 2: Top: variables X and Y sharing the same cache line. Bottom: cache lines padded (represented as P) to avoid false sharing.

Although false sharing is often overlooked, it can significantly and silently degrade the performance of concurrent code. The simplest example is a thread incrementing a long variable in a loop. Running two of these threads in parallel on a multi-core system should in theory yield the same performance figures as running a single thread, since each one should run on its own core and increment its own independent variable. In reality, when both variables share the same cache line, the threads will need more than twice the time to complete, and this ratio will increase as we add more threads [8]. In order to solve this issue, the variables which are the most susceptible to write contention can be padded in order to fill the full cache line (Fig. 2). The padding forces the variables to be placed in different cache lines.

DISRUPTOR ARCHITECTURE

At the heart of the Disruptor is the ring buffer (Fig. 3). This data structure is used to pass messages between producers and consumers. It has a bounded size, in order to apply backpressure if the consumers are not able to keep up with the producers.

It is backed by an array, which is initialised up front in order to be fully allocated in contiguous blocks of memory. The array structure and the contiguous memory make it cache friendly because the CPU will be able to detect that a consumer is walking through the memory in a predictable pattern and will prefetch the memory into its caches.

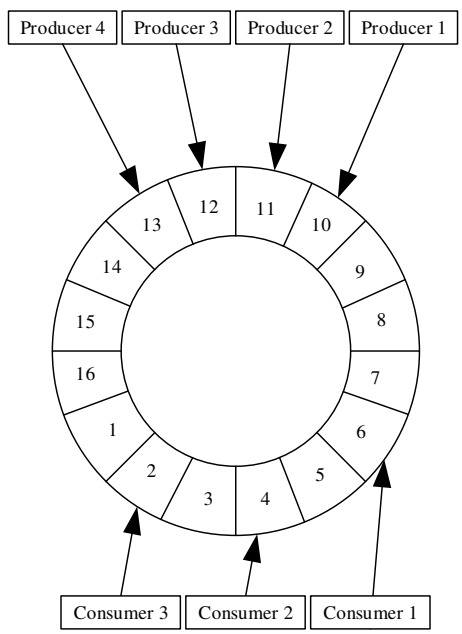


Figure 3: Ring Buffer

The elements in the ring buffer are mutable. When the producers reach the end of the buffer and wrap to the start, they reuse existing entries and overwrite them. This means that this data structure does not generate any work for the garbage collector.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The progress of producers and consumers on the ring buffer is tracked by sequence numbers. A sequence number is a 64 bit long number, which is padded to fill a full cache line. Since these variables are frequently read and updated from concurrent threads, they could be a source of contention. The padding removes the risk of false sharing.

Memory Visibility

The memory visibility of the data exchanged between producers and consumers relies on the sequence numbers. Updating a sequence number is similar to writing a Java *AtomicLong*: it is translated into a *compareAndSwap* CPU instruction. The CPU memory model guarantees that the memory modified before a *compareAndSwap* will be visible by other threads (this is true for the most common CPU architectures, in the other case the compiler will add an additional synchronization) [9]. This allows the Disruptor framework to be lock-free, thus eliminating the main contention point that comes with traditional implementations of queues.

Batching

In most producer/consumer architectures, the producers regularly outperform the consumers. The Disruptor framework offers a smart way to catch up for consumers: if several items are waiting on the queue, consumers can process the full batch of available items at once instead of processing one item at a time. Batching is usually more efficient, especially when it involves I/O.

Consumer Dependencies

For a simple usage, the Disruptor can be used as a queue between producers and consumers. In addition, the API allows to declare a dependency graph between consumers (Fig. 4).

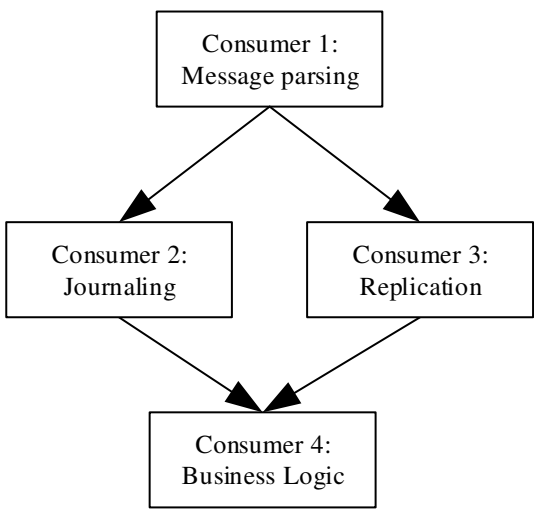


Figure 4: Example of consumer dependency

Real world applications often have several processing layers connected by queues. With this feature, the Disruptor can replace multiple layers of queues. The data will be exchanged between the dependant consumers over the ring buffer, eliminating the contention and delays brought by the traditional queuing approach.

All these concurrent programming concepts are complex and very error prone. They are a common source of errors in real world applications and often difficult to troubleshoot.

The Disruptor architecture actually encourages code simplification by writing the business logic in a single thread. Since the framework takes care of the synchronization, the business logic can be uncluttered of the concurrency concerns. As a result, it is easier to reason about it, as well as to test and maintain it.

DISRUPTOR USAGE IN CESAR

The 1300 devices controlling the beam lines generate multiple streams of data. The CESAR server is in charge of acquiring this data in order to compute an overall state for each device. The first implementation was using manual locking on the data structures in order to synchronize the concurrent data streams. The main reason to use the Disruptor is to simplify the overall device state calculation.

In the new design, the messages coming from the devices are stored on the ring buffer. Then a single Disruptor thread updates simple buffers dedicated to keep the last value for each data stream, and based on these last values computes the overall device state that is published to clients (Fig. 5).

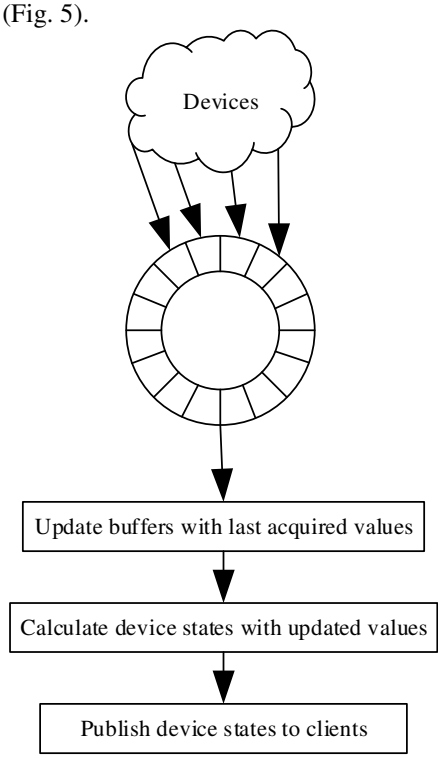


Figure 5: State computation in CESAR

Benefits

This architecture has many benefits. The threads delivering messages from devices never encounter a lock and are released very quickly. If multiple messages are

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

available on the ring buffer, they are processed as a batch. This is a good fit for the experimental areas, where most detectors publish their state simultaneously after being triggered by a timing event. The batching mechanism allows CESAR to process such bursts of messages more efficiently.

The state computations are completely deterministic: since a single thread processes messages in a known FIFO order, we can easily understand why a given state was calculated by looking at the message log. When a race condition causes a bug in concurrent code, developers are often left wondering why something that seemed impossible actually happened. This usually occurs at the worst possible times, when the system is heavily loaded and when the applications logs offer little help to understand in which order things really happened. The single threaded design eliminates this class of problems while keeping an excellent throughput.

Since we can rely on the fact that the business logic code runs on a single thread, we can write lock-free code that is more efficient and simpler. This also reduces the overall cost of maintenance, because the business logic code has a high chance of being modified during the software's lifetime.

The main concern when moving to a single thread is to know if it will be fast enough. Considering that the Disruptor was designed for high performance, we speculated that it would easily cover our needs. We indeed measured that our code, without any optimization effort, was able to process around 1 million messages per second. This is more than enough for our current needs and gives us some room to grow, as other Disruptor users reach more than 10 million messages per second with optimized code.

Limitations

During our refactoring we found that integrating the Disruptor in an existing architecture is reasonably simple, and less invasive than other approaches like actor frameworks. There are nonetheless some aspects to consider carefully before using this model.

The ring buffer is designed to apply backpressure. This is usually a good design choice to fail gracefully under load, but one should evaluate if the other parts of the application are compatible with that approach. For instance it might be a business choice to define if messages can be lost when the buffer is full.

As the computations run on a single thread, developers must make sure that this thread is never blocked. Any kind of blocking I/O such as an interaction with a traditional database should be avoided. The most common logging frameworks also make use of locks, which could add contention to this thread and reduce the performance dramatically [10]. After executing the business logic, a common use case is to publish the result of the computation. If this publication is using potentially blocking I/O, such as a remote message broker, the design should handle a network or broker failure without blocking the processing. In our case, we publish the calculated device states over a JMS broker, and decided to add an additional layer of buffering that keeps only the latest device states if the publication cannot keep up. This is acceptable because our GUI only needs the latest updates.

The Disruptor architecture is inherently asynchronous. This is a natural design choice for control applications that handle streams of data. At the same time, an extra effort is necessary if it is required to support synchronous operations as well. In our case, CESAR is required to support a synchronous refresh for the overall device states, based on the current hardware information. In order to create a synchronous functionality based on asynchronous services, we had to carefully analyse the different scenarios that could happen in the asynchronous world (timeouts, concurrent requests, etc.).

CONCLUSION AND OUTLOOK

The new CESAR architecture based on the Disruptor has been used operationally for over a year and proved to be very stable. The simplification brought by the separation of concerns between the concurrency aspects and the business logic allows for easy maintenance and extension of the code base. We believe that a similar architecture can be used for other control systems and can be particularly beneficial for the ones that need to handle large amounts of events with low latency.

A possible area of improvement for CESAR would be to follow the design adopted by the creator of the library [11]: journal and replicate all messages and base the server state on these messages only. This would bring hot-swappable servers and an easy way to reproduce operational scenarios on a developer's machine.

REFERENCES

[1] For example Goldman Sachs collections: https://github.com/goldmansachs/gs-collections
[2] http://www.lmax.com/exchange
[3] https://en.wikipedia.org/wiki/Foreign_exchange_Market
[4] https://en.wikipedia.org/wiki/Multilateral_trading_facility
[5] https://en.wikipedia.org/wiki/Staged_event-driven_architecture
[6] http://www.gotw.ca/publications/concurrency-ddj.htm
[7] http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html
Section 2.2.5.4 Data Prefetching
[8] http://mechanical-sympathy.blogspot.ch/2011/07/false-sharing.html
[9] http://www.azulsystems.com/blog/cliff/2010-07-24-unsafe-compareandswap
[10] http://www.grobmeier.de/log4j-2-performance-close-to-insane-20072013.html
[11] http://martinfowler.com/articles/lmax.html

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

ACCELERATOR MODELLING AND MESSAGE LOGGING WITH ZeroMQ

J. Chrin, M. Aiba, A. Rawat, Z. Wang, Paul Scherrer Institut, 5232 Villigen PSI, Switzerland

Abstract

ZeroMQ is an emerging message-oriented architecture that is being increasingly adopted in the software engineering of distributed control and data acquisition systems within the accelerator community. The rich array of built-in core messaging patterns may, however, be equally applied to within the domain of high-level applications where a seamless integration of accelerator models and message logging capabilities, respectively, serve to extend the effectiveness of beam dynamics applications and allow for their monitoring. Various advanced patterns that include intermediaries and proxies further provide for reliable service-oriented brokers, as may be required in real-world operations. A report on an investigation into ZeroMQ's suitability for integrating key distributed components into high-level applications, and the experience gained, are presented.

MOTIVATION

ZeroMQ [1] is an emerging message-oriented architecture that has already made a profound impact within the wider accelerator and experimental physics community. A favourable evaluation from among several contemporaries [2] has since seen it target existing CORBA (Common Object Request Broker Architecture) [3] systems at several facilities [4–6]. It is also being adopted in message-based data acquisition systems [7–10] and has provided the means by which beam synchronous data is retrieved in high-frequency pulsed accelerators [9, 11]. Its apparent identification as today's middleware of choice has triggered an interest for its applicability within the domain of high-level applications at SwissFEL, Switzerland's X-ray Free-Electron Laser Facility [12, 13]; where CORBA may once have been used at a previous facility for incorporating distributed components into beam dynamics applications [14, 15], ZeroMQ presents itself as a viable, state-of-the-art, alternative that deserves consideration. Of particular interest is the integration of accelerator models and message logging capabilities which respectively serve to extend the effectiveness of beam dynamics applications and allow for their monitoring.

DISTRIBUTED COMPUTING WITH ZeroMQ

ZeroMQ is a lightweight, socket-like, asynchronous messaging library that provides for the transport of raw message buffers in a flexible and scalable distributed computing environment. The idiosyncratic name lends itself to the project's ambition to reach maximal performance by minimizing latency, copying, and the necessity for brokers (i.e. their numbers approach the limit of *Zero*). A first investigation into the ZeroMQ library already reveals a number of compelling features:

- A rich array of messaging patterns, including the familiar request-reply, publish-subscribe and push-pull (pipeline) patterns, each of which defines a distinct network topology.
- The availability of both unicast (inproc, ipc, tcp) and multicast (pgm, epgm) transport layers.
- The ability to use these patterns and transports as building blocks to establish connections between processes, with or without intermediate brokers/proxies.
- Support for multipart messages, which allow multiple frames to be concatenated into a single message to be sent over the network.

That these features are all available in a *single* library is positively favourable. (By comparison, CORBA requires separate libraries for their event driven and other services.) Furthermore, an active ZeroMQ community provides support for numerous platforms and an increasing multitude of programming languages.

Despite these benefits, some important components, that are outside ZeroMQ's stated interest, still need to be catered for to attain a fully fledged distributed infrastructure:

- A Name Service that translates logical addresses into bind/connect endpoints.
- An Implementation Repository for the activation and re-activation of servers.
- Support for object serialization.

With these shortcomings identified, what then are the remedies? Although a name resolution service may be developed from among ZeroMQ's architectural patterns, for the present time, the use of configuration files for publicising tcp/ip addresses is manageable. Most CORBA developers will have become accustomed to an Implementation Repository that interacted with the server's Object Adapter Mediator to handle the administrative aspects of server (re-)activations [14]. A similar setup for ZeroMQ would ensure that applications are never starved of the services that they require. The lack of an interface to serialize structured data may, at first, appear as a glaring omission given that most use cases would require it. Fortunately, this situation is redeemed through third-party solutions which vary in form, complexity and performance, endowing developers with the prerogative to choose that which best suits their needs.

Having now gained an insight into ZeroMQ's capabilities, its applicability to within the beam dynamics environment is readily recognized. The request-reply messaging pattern,

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

coupled with a suitable protocol buffer for the serialization of complex data types, provide the ingredients for the implementation of a platform independent and language neutral, client-server framework, enabling the exchange of data between online accelerator models and distributed applications. The reactive publish-subscribe pattern, in preference, delivers an appropriate event-driven paradigm for constructing a software framework for the propagation of diagnostic messages to a central logging and monitoring facility. Here, ZeroMQ's multipart message protocol allows a single message type to be composed from several frames. The various advanced patterns that include intermediaries and proxies further provide for reliable service-oriented brokers, as may be required in real-world operations.

ACCELERATOR MODELLING

The use of accelerator models is an essential feature in both accelerator design and emulation, allowing developers to manipulate the variables that determine the dynamics of the particle beam in a simulated framework. For the most part, however, these models were originally intended for use in isolation. Typically, an ASCII input file, in the model-specific format, containing lattice information and a set of directives to compute the desired quantities is provided, and the resulting output is directed to files for post-processing analysis. For certain models, however, where the compilation of the code into a shared object is a workable prospect, then their accessibility from a high-level language may be anticipated. Procedures that have been verified *offline* can then be effectively engaged for the optimization of the accelerator *online*. Methods may be executed to retrieve beam dynamics (e.g. linear optics) parameters for a given setting of the accelerator, for example, and model based corrections subsequently applied [16, 17].

The feasibility to expose such accelerator models to beam dynamics applications in a language-neutral manner in itself provides strong motivation for their incorporation into the ZeroMQ messaging architecture. The resulting data interfaces are invariably structured and any meaningful data exchange requires serialization, however.

Serialization with Google Protocol Buffers

While a number of options exist for data serialization, the Google Protocol Buffers [18] provide a binary encoding format that has a number of recognized advantages. An interface definition language allows structured data schemas to be specified from numbered fields affixed with well-defined keywords and data types. These ensure backward compatibility, validation and extensibility. The Protocol Buffers are implemented in several languages easing interoperability between applications from different domains. In the course of this work, Google introduced Protocol Buffers language version 3, `proto3`. The new version has a simplified interface definition language structure, making it more accessible to a broader range of programming languages. Several new features have also been added to support nomenclatures such as the `Any` type, the associative `map` and `oneof` keyword. A future release of `proto3` is to provide a well-defined encoding in JSON (JavaScript Notation Object) [19] as an alternative to binary proto encoding for data consumption by e.g. web browsers. The migration from `proto2` to `proto3` for our limited set of `proto` files was a straightforward endeavour; `proto3` is not, however, backward compatible with `proto2` although the latter syntax may still be incorporated into the former if so declared.

PyLiTrack

The Python computation of LiTrack [20] provides fast, two-dimensional, longitudinal single-bunch tracking. The relatively uncomplicated interface provides a useful test case for integrating an accelerator model into the ZeroMQ architecture through the Google Protocol Buffers. The data schema (`proto`) file, shown in Fig. 1, is, consequently, comparatively light. The `protoc` binary compiler automatically generates stub class files that are used for data encoding and parsing. Input arguments to the line command determine what languages are provided for. In the present setup, PyLiTrack runs as a server and clients connect using ZeroMQ's request-reply pattern. In this way, the Pythonic tracking code is made available to non-Pythonic applications.

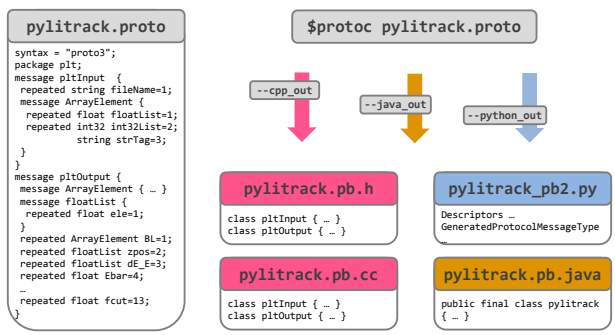


Figure 1: The generation of stub classes, for various languages, from the PyLiTrack proto file.

MAD-X

The MAD-X (Methodical Accelerator Design) simulation code [21] is regarded as a defacto standard for the computation of beam-optics parameters for a given accelerator lattice. In addition to the stand-alone executable, it is available as a C++ library with bindings for Python [22]. An identical procedure for data serialization is applied here. The corresponding proto file exposes the so-called Twiss tables which parameterise the beam ellipse in phase space. One distinct advantage of using the library as opposed to running the stand-alone executable is that the lengthy and computer intensive initialization step (where the long list of sequences that define the model are interpreted) need only be loaded into memory once. Numerous iterations, as required in fitting procedures for instance, can be undertaken without having to continually re-initialize with the same, given model definition. At the same time, in cases where

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

successive single tasks necessitate a newly created address space, a server-client configuration further gives confidence that the accelerator model is always properly initialized on (re-)activation. One variant applied sees a Python client communicating with a Python MAD-X server, using ZeroMQ as the transport layer, but preferring Python's own Pickle module for the serialization. The activation of the server is enacted through the use of Python scripts. While this may suffice in the short term, the call for an Implementation Repository for handling the (re-)activation of ZeroMQ servers becomes apparent!

Next Steps

The accelerator models so far considered represent the most use-cases for online modelling at SwissFEL. While a proof-of-principle has been demonstrated, a client-side Application Programming Interface (API) for PyLiTrack and MAD-X that hides the ZeroMQ and serialization implementation details is to be finalized. In the absence of an Implementation Repository, advanced patterns that provide for reliable service-oriented brokers may also be preferred to the present synchronous interactions.

MESSAGE LOGGING AND MONITORING

A messaging logging and monitoring facility, henceforth referred to as the message logger, has been developed to allow applications to report (publish) their diagnostic and information data to interested subscribers. These include a database (Oracle [23]) writer process for the storage of messages and a Graphical User Interface (GUI) for the display of messages both in real-time and offline through database retrieval operations. ZeroMQ's multipart frames and the extended publish-subscribe pattern, respectively form the message envelope and communication layer.

Multipart Messages

ZeroMQ allows messages to be assembled from individual frames arising from different sources. The resulting "multipart message" effectively adds a coarsely formed structure to the single message that is delivered to the network. The need for marshalling/unmarshalling the data is alleviated and ZeroMQ's low-latency performance is not compromised. The leading frame of the multipart message also serves as a "topic" in ZeroMQ's publish/subscribe architecture to set filters that allow only events of interest to propagate to the subscriber.

Extended Publish and Subscribe

The extended publish-subscribe model, shown in Fig. 2, is the pattern adopted for broadcasting messages to interested clients. The proxy (or broker), which lies between the publishers (PUB) and subscribers (SUB), provides the solution to the so-called "dynamic discovery problem" by binding XSUB and XPUB sockets (which expose subscriptions as special messages) to the advertised IP addresses and ports [24]. Publishers and subscribers need only connect to the XSUB and XPUB sockets of the proxy, rather than to one another. The proxy then takes charge of forwarding messages to subscribers. In this way, publishers, i.e. high-level applications, may be easily hooked in to the system and have their messages written to the database, viewed from within a GUI or received by any other subscriber, e.g. console, that can be readily added to the network.

ZeroMQ's response to continuous messages from multiple publishers is for the proxy to collect the messages evenly from among the publishers ("fair-queued"). In the event of message overload, where publishers outpace consumers, messages are queued on the publisher's side with the size of the cached buffer determined by a configurable "high-water mark" limit. The framework also profits from ZeroMQ's "zero-copy" capability in that buffers created by the publisher can be sent directly by the message. The data does still need to be written into the application buffers at the receiving end, however.

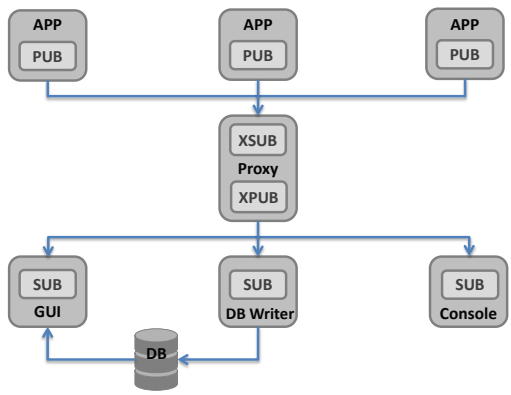


Figure 2: The extended publish-subscribe pattern as applied to the message logger.

Message Content

The specific details that comprise the message content require careful consideration. More information within a message may prove to be synonymous with a better reporting ability, but should nevertheless be balanced against network traffic and storage capacity interests. Naturally, for each message (or event) a consistent set of data should be evident. The established `syslog` protocol [25] acts as a basis for deciding on the mandatory fields. These are supplemented by a number of optional fields that are filled at the discretion of the user. The message content was finalized in consultation with machine operation leaders [26]. It is, however, the provision of the user to supply meaningful and helpful messages, that also propose solutions (for which an action field is also provided) that ultimately hastens a return to normal operation.

Each message field is housed within a multipart message frame, simplifying the data unpacking process. The database writer, for instance, maps frames directly onto a corresponding database schema, as indicated in Fig. 3.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

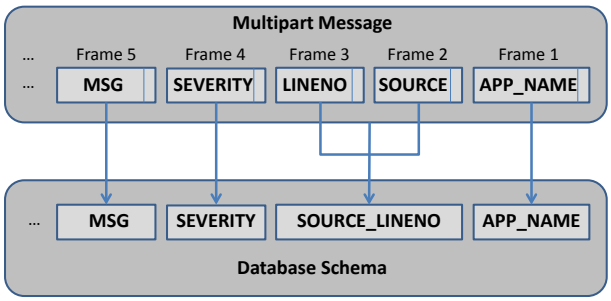


Figure 3: The database writer maps multipart message frames to database columns on a near one-to-one basis.

A Library for Publishers

To facilitate application developers with the reporting of messages, a dedicated library, `zmsglog`, was developed that hides the ZeroMQ implementation details and provides an easy-to-use interface. The composed messages follow a predefined format with required entries being filled automatically by the provided API, with the obvious exception of the user supplied message itself. An important feature of the `zmsglog` is its ability to efficiently handle bursts of repeated error messages. Such message bursts are cached on the publisher's side and only a summary of their occurrence need be sent over the network, thereby minimizing the network traffic. The message logging facility has been successfully put through high-volume data stress tests (without caching of repeated methods). Listings 1 and 2 show a minimal implementation from Python, using an inherited class, and MATLAB, using the base class, respectively.

Listing 1: Python API for `zmsglog`

```
warnMsg = MsgLog.CyWarnMsg("OrbitDisplay")
warnMsg.setMsg("RMS outside operating limits")
warnMsg.send(__file__, __LINE__())
```

Listing 2: MATLAB API for `zmsglog`

```
msglog('setAppName','3DScan')
msglog('setMsg','setVal outside hardware limit')
msglog('send','error',dbstack())
```

A GUI for Subscribers

A GUI to the message logger has been developed in Python/Qt (Fig. 4). It is able to display live messages in real-time, with a dedicated window pane configured to monitor priority, i.e. machine-critical, applications and fatal messages. A third window provides the user with an interface for database retrieval operations, with ample sorting and filtering possibilities. A database polling mechanism may also be activated if preferred. The GUI may also be activated for the purpose of a single, specific publisher by supplying the application name as a command argument. The application name acts as the topic in the multipart message by which the GUI (subscriber) sets a filter in order to select only the given application's messages.

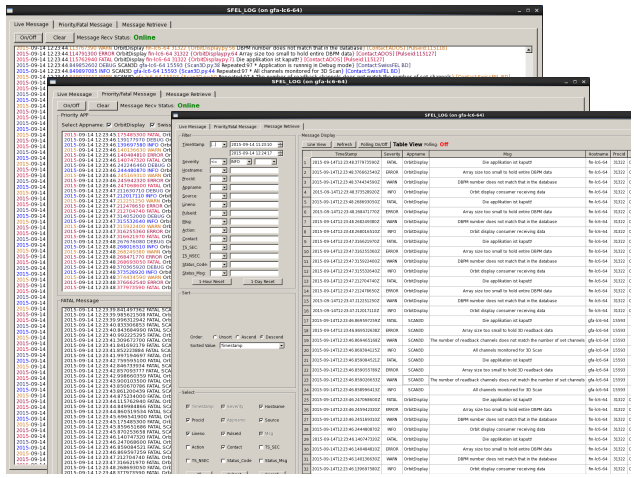


Figure 4: A GUI for viewing messages: in real-time (back, middle), from priority applications (middle top), of highest severity (middle bottom), and from the database (front).

Configuration File

A number of properties of the message logger are kept separate from the application code and are managed through a JSON configuration file, enhancing flexibility and simplifying code maintenance. The possibilities range from setting the ZeroMQ bind/connect endpoints, the high water mark limit, announcing priority applications, to selecting the display colours for the various message severity levels.

Next Steps

The present framework is well matched for the anticipated demand from high-level applications during the first, two-year, SwissFEL beam commissioning phase [13]. Nevertheless, since interest has surfaced to provide `zmsglog` to low-level (e.g. feedback) systems, and with it the potential for high-volume data flows, the scalability of the logging and data mining components for problem tracing [27] requires further consideration. To this end, a number of open-source solutions, based on Apache Lucene [28], have been identified [29]. Among these is the ELK stack [30] – Elasticsearch, Logstash, and Kirbana – which provide a complete framework for data redirection, storage, analysis and visualization. The Logstash API also provides a data pipeline for receiving ZeroMQ messages that would allow it to plug in effortlessly to the architecture presented in Fig. 2, as a subscriber to the proxy.

SUMMARY

Various facets of the ZeroMQ asynchronous messaging library have been explored and their usefulness to within the domain of high-level applications has been recognized. In particular, a framework based on the request-reply pattern, coupled with the Google Protocol Buffers for the serialization of data, has been implemented for accessing accelerator models from different programming languages. The publish-subscribe pattern together with ZeroMQ's multi-

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

part messaging framework have formed the ingredients for a message logging and monitoring facility that displays live data in real-time. A notable feature throughout has been the relative ease with which it has proved to employ the various ZeroMQ messaging patterns, thereby releasing time and effort to focus on the specific goals at hand.

REFERENCES

[1] ZeroMQ, http://zeromq.org/.

[2] A. Dworak, P. Charrue, F. Ehm, W. Sliwinski, and M. Sobczak, "Middleware Trends and Market Leaders 2011", in *Proc. 13th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'11)*, Grenoble, France, Oct. 2011, paper FRBHMULT05, pp. 1334–1337.

[3] CORBA (Common Object Request Broker Architecrure), http://www.omg.org/.

[4] W. Sliwinski, I. Yastrebov, and A. Dworak, "Middleware Proxy: A Request-driven Messaging Broker for High Volume Data Distribution", in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper TUCOCB02, pp. 948–951.

[5] A. Götz, E. Taurel, P. Verdier, and G. Abeille, "TANGO - Can ZMQ Replace CORBA?", in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper TUCOCB07, pp. 964–968.

[6] Y. Le Goc *et al.*, "Prototype of a Simple ZeroMQ-based RPC in Replacement of CORBA in NOMAD", in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper TUPPC042, pp. 654–657.

[7] T. Matsumoto, Y. Furukawa, and M. Ishii, "Next-generation MADOCA for the SPring-8 Control Framework", in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper TUCOCB01, pp. 944–947.

[8] A. Yamashita and M. Kago, "A New Message-based Data Acquisition System for Accelerator Control", in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper MOPPC130, pp. 413–416.

[9] K. Rehlich, "Recent Hardware and Software Achievements for the European XFEL", presented at the 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13), San Francisco, CA, USA, Oct. 2013, paper THCOBB02.

[10] S.G. Ebner, H.R. Billich, H. Brands, E.H. Panepucci and L. Sala, "Data Streaming - Efficient Handling of Large and Small (Detector) Data at the Paul Scherrer Institute", presented at the 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15), Melbourne, Australia, Oct. 2015, paper WED3O06, these proceedings.

[11] S.G. Ebner, H. Brands, B. Kalantari, F. Märki, and L. Sala, "SwissFEL Beam Synchronous Data Acquisition - A Sneak Peek under the Hood", presented at the 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15), Melbourne, Australia, Oct. 2015, paper MOPGF059, these proceedings.

[12] "SwissFEL Conceptual Design Report", R. Ganter, Ed. PSI, Villigen, Switzerland, Rep. 10-04, Version Apr. 2012.

[13] T. Schietinger, "Beam Commissioning Plan for the SwissFEL Hard X-ray Facility", presented at the 37th Int. Free-Electron Laser Conf. (FEL'15), Daejeon, Korea, Aug. 2015, paper MOP017.

[14] M. Böge and J. Chrin, "On the Use of CORBA in High Level Software Applications at the SLS", in *Proc. 8th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'01)*, San Jose, CA, USA, Oct. 2001, paper THAT002, pp. 430–432.

[15] M. Böge and J. Chrin, "Developments to the SLS CORBA Framework for High Level Software Applications", in *Proc. 10th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'05)*, Geneva, Switzerland, Oct. 2005, paper WE4A.1-5O.

[16] M. Böge, J. Chrin, M. Muñoz, and A. Streun, "Commissioning of the SLS using CORBA Based Beam Dynamics Applications", in *Proc. 2001 Particle Accelerator Conf. (PAC'01)*, Chicago, IL, USA, Jun. 2001, paper TOPB012, pp. 292–294.

[17] M. Böge and J. Chrin, "Integrating Control Systems to Beam Dynamics Applications with CORBA", in *Proc. 2003 Particle Accelerator Conf. (PAC'03)*, Portland, OR, USA, May 2003, paper TOPB010, pp. 291–293.

[18] Google Protocol Buffers, https://developers.google.com/protocol-buffers/.

[19] JSON (JavaScript Notation Object), http://json.org/.

[20] K.L.F. Bane and P. Emma, "LiTrack: A Fast Longitudinal Phase Space Tracking Code with Grapghical User Interface ", in *Proc. 2005 Particle Accelerator Conf. (PAC'05)*, Knoxville, TN, USA, May 2005, paper FPAT091, pp. 4266–4268.

[21] MAD-X, http://madx.web.cern.ch/madx/.

[22] K. Fuchsberger and Y. Inntjore Levinsen, "PyMad – Integration of MadX in Python", in *2nd Int. Particle Accelerator Conf. (IPAC'11)*, San Sebastián, Spain, Sep. 2011, paper WEPC119, pp. 2289–2291.

[23] Oracle, http://www.oracle.com/.

[24] P. Hintjens, "The Dynamic Discovery Problem", in *ZeroMQ*, Sebastopol, CA, USA: O'Reilly Media, Inc., 2013, pp. 45–47.

[25] R. Gerhards, "The Syslog Protocol", RFC 5424, Mar. 2009, doi: http://dx.doi.org/10.17487/RFC5424

[26] J. Chrin, A. Lüdeke, and D. Voulot, "Message Logging Specifications for SwissFEL Applications", PSI, Villigen, Switzerland, SwissFEL Document ID. FEL-CJ84-001-01, Sep. 2015.

[27] F. Ehm and A. Dworak, "A Remote Tracing Facility for Distributed Systems", in *Proc. 13th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'11)*, Grenoble, France, Oct. 2011, paper WEPKS024, pp. 840–843.

[28] Apache Lucene, http://lucene.apache.org/.

[29] S.G. Ebner, private communication.

[30] Elasticsearch, https://www.elastic.co/.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

WHY SEMANTICS MATTER: A DEMONSTRATION ON KNOWLEDGE-BASED CONTROL SYSTEM DESIGN

W. Pessemier, G. Raskin, and H. Van Winckel, Institute of Astronomy, KU Leuven, Leuven, Belgium
P. Saey and G. Deconinck, ESAT, KU Leuven, Leuven, Belgium

Abstract

Knowledge representation and reasoning are hot topics in academics and industry today, as they are enabling technologies for building more complex and intelligent future systems. At the Mercator Telescope, we've built a software framework based on these technologies to support the design of our control systems. At the heart of the framework is a metamodel: a set of ontologies based on the formal semantics of the Web Ontology Language (OWL), to provide meaningful reusable building blocks. Those building blocks are instantiated in the models of our control systems, via a Domain Specific Language (DSL). The metamodels and models jointly form a knowledge base, i.e. an integrated model that can be viewed from different perspectives, or processed by an inference engine for model verification purposes. In this paper we present a tool called OntoManager, which demonstrates the added value of semantic modeling to the engineering process. By querying the integrated model, our web-based tool is able to generate systems engineering views, verification test reports, graphical software models, PLCopen compliant software code, Python client-side code, and much more, in a user-friendly way.

INTRODUCTION

Semantic models consist of pieces of information, and the relationships between those pieces. The ability to link any piece of information with another, thereby conveying the meaning (semantics) of the information, is what sets them apart from more "rigid" models such as those found in relational databases or object-oriented software. Semantic models are therefore well suited to represent all sorts of knowledge about the real world. At the Belgian Mercator Telescope (La Palma, Spain) we use the expressive power of semantic models to capture engineering knowledge of the telescope control system, which is being ported to a Programmable Logic Controller (PLC). As will be demonstrated in this paper, we have developed systems, electrical and software engineering models of several subsystems of the telescope, and successfully used these models for documentation, verification and implementation purposes.

FRAMEWORK ARCHITECTURE

As shown in Fig. 1, we have built a software framework centered around a Knowledge Base (KB) that combines (integrates) metamodels and models [1]. This KB can be queried by a tool which we developed (OntoManager), and the results of those queries can be fed into a template system to produce documents such as web pages and source code files. The metamodels provide the building blocks to construct models.

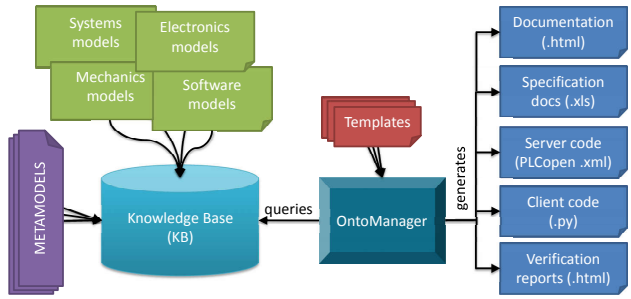


Figure 1: Framework architecture.

They are true *ontologies* as they define a vocabulary in such a way that the meaning of this vocabulary is well defined. Fig. 2 shows small excerpts of our systems metamodel (sys), mechanics metamodel (mech) and electronics metamodel (elec). They define concepts ("classes") such as `sys:Feature` and `mech:Assembly`, and relationships ("properties") such as `sys:hasFeature` and `elec:isConnectedTo`. Unlike a simple vocabulary, the meaning of these terms is further constrained whenever possible. For instance, the definition of a `mech:Assembly` says that "something" is a mechanical assembly if and only if it has at least two mechanical parts. Models based on this vocabulary will have to adhere to these constraints in order to be valid. For the underlying formal semantics (SubClassOf, EquivalentTo, Domain, ...) we depend on the Semantic Web standards RDFS (Resource Description Framework Schema) and OWL (Web Ontology Language) [2].

Systems metamodel (*sys*)

```
ObjectProperty: sys:hasFeature
  Domain: sys:System
  Range: sys:Feature
Class: sys:Feature
Class: sys:System
  EquivalentTo: sys:hasFeature some sys:Feature
```

imports

Mechanics metamodel (*mech*)

```
Class: mech:Part
  SubClassOf: sys:Feature
Class: mech:Assembly
  EquivalentTo: sys:hasFeature min 2 mech:Part
```

imports

Electronics metamodel (*elec*)

```
Class: elec:Conductor
  SubClassOf: mech:Part
  EquivalentTo: elec:carries exactly 1 quantity:ElectricCharge
  spin:constraint:
    [ a       sp:Construct .
      sp:text "CONSTRUCT { _:b0 a          spin:ConstraintViolation ;
                           _:b0 rdfs:label 'Signal mismatch!' .      }
               WHERE     { ?this  elec:carries       ?thisSignal .
                           ?this  elec:isConnectedTo ?other .
                           ?other elec:carries       ?otherSignal .
                           FILTER(?thisSignal != ?otherSignal)       }" ]
Class: elec:Contact
  EquivalentTo: sys:Interface and elec:Conductor
ObjectProperty: elec:isConnectedTo
  Characteristics: Symmetric
  Domain: elec:Contact
  Range: elec:Contact
```

Figure 2: Small excerpts of some metamodels.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Models that use the metamodel vocabulary consist of **explicit** statements such as `"x rdf:type elec:Contact"` (x is an electric contact) and `"x elec:isConnectedTo y"` (x is electrically connected to y). However, according to the semantics of the metamodel (see Fig. 2), many more **implicit** statements may be inferred. For example, the formerly mentioned statement implicitely leads to `"x rdf:type elec:Conductor"` (because `elec:Contact` is a subclass of `elec:Conductor`) while the latter statement implicitely leads to `"y elec:isConnectedTo x"` (because the `elec:isConnectedTo` relationship is symmetric according to the metamodel). A so-called *reasoner* or *inference engine* (SPIN API[1] in our case) can automatically produce these inferred statements and add them to the KB. Similarly, supposing two contacts are connected to each other, one of them carrying a 24V signal and the other one a 4...20mA signal, then the constraint pattern of the `elec:Conductor` will match, and a constraint violation will be generated by the reasoner and added to the KB. The metamodels, models and inferred knowledge can thus all be represented as lists of statements, which can be stored in text files. In our framework we parse those files using the open source RDFLib[2] Python package. While systems can be modeled "directly" as a list of statements, real-world systems will unavoidably result in many thousands of explicit statements (and a multitude of implicit ones). We therefore need a more efficient way to populate the models, as explained in the next section.

POPULATING THE KB

To develop models based on the metamodel vocabulary in a convenient way, we need a modeling language. Graphical languages such as UML and SysML are an option (as they can be extended with stereotypes), but in our experience, complex models of real-world systems require more syntax than these languages typically offer. We have therefore developed a Domain Specific Language (DSL) called Ontoscript. Ontoscript is an "internal" DSL as it is valid coffeescript[3], only it is used in a particular way. We have adopted this idea from the Giant Magellan Telescope project [3]. An example of an Ontoscript model is displayed in Fig. 3. It shows how an instance of an I/O module of type "EL1088" is added to a project, and its terminals are connected to the pins of a connector. When this script is executed, then the `IO_MODULE_INSTANCE` function is called, producing hundreds of statements. For instance, for each channel and each terminal of the "EL1088" I/O module **type**, a corresponding channel and terminal will be added automatically to the I/O module **instance**. These newly created terminal instances can then be further described and connected (via the `elec:isConnectedTo` relationship) to the pins of some previously modeled connector. This idea is very similar to object-oriented software, because when a class is instantiated, then also all variables (and sub-variables) of the class definition must be added to the instance. In our framework this functionality is programmed by Ontoscript functions such as `IO_MODULE_INSTANCE`, `CLASS_INSTANCE`, etc. When the Ontoscript models are executed, they produce text files containing thousands of statements. When parsing these text files using RDFLib, we have a KB which we can start to query.

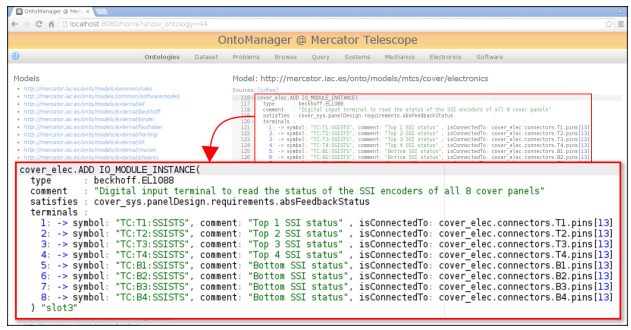


Figure 3: I/O module instance modeled in Ontoscript.

QUERYING THE KB

Using the "Query" tab of our OntoManager tool, we can submit arbitrary queries to the KB (which is essentially an RDFLib instance). The de-facto query language of the Semantic Web (called SPARQL[4]) uses pattern matching and filters to select the requested information. Such a SPARQL query executed by the OntoManager tool is shown in Fig. 4. The query selects all I/O module types in the KB, their manufacturer and their description, and counts their instances. By looking at the query, one can see that this kind of information can be retrieved with only knowing the metamodel and some simple SPARQL syntax. SPARQL also supports more advanced queries (e.g. including numerical calculations or comparison), which are often used for verification purposes.

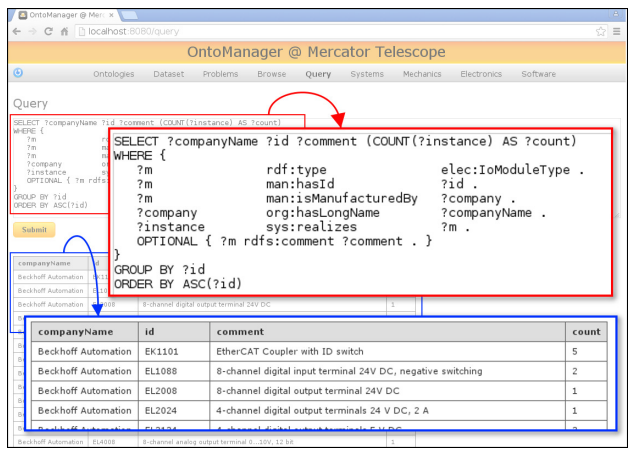


Figure 4: Arbitrary query executed in OntoManager (red) and its results (blue).

[1] `http://topbraid.org/spin/api`
[2] `http://github.com/RDFLib`
[3] `http://coffeescript.org`
[4] `http://www.w3.org/TR/sparql11-overview`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

VIEWING THE KB

While the Query tab of OntoManager can be used to extract some specific information from the KB, we can also predefine some queries and a suitable template as a so-called "view". Several of these views have been integrated in OntoManager so far, as can be seen in Fig. 5. The main part of the figure shows a web-based view of the I/O module instance "slot3" of the "Cover" electrical configuration. The latter is part of the dust cover subsystem of the Mercator Telescope, which seals the telescope tube during daytime via eight pivoting aluminum panels (similar to the petals of a flower). Dozens of queries have been executed in the background to produce this "electronics" view, e.g. to retrieve the system properties, the channel and terminal properties, the electric connections to the connectors of the cabinet, and so on. The results of these queries are fed into a web-based template (written in mako[5] syntax) hosted by a web server (based on the Pyramid[6] framework). We have developed very similar views for other frequently used electrical components such as connectors and motor drives. All views are linked to each other via the semantic model: for instance, we can see that terminal 1 of the shown I/O module instance is connected to pin 13 of the connector instance `T1`. By clicking the `T1` hyperlink, the connector view will be shown, offering the connector's pin layout and other details. Thanks to a predefined template for viewing I/O modules, the Ontoscript model `slot3` of Fig. 3 can thus be displayed as a nicely formatted and "clickable" web page without any additional effort. The resulting web pages document the electrical system and may be used as specification documents (for drawing the electrical schemas) or for troubleshooting purposes. Due to the many visible hyperlinks, it is very easy to browse the documentation, thereby navigating trough the electrical system from one component to another.

In a similar way, also information of other engineering disciplines may be linked to the electrical information via the underlying semantic model. For instance, in Fig. 5 the shown I/O module instance satisfies a requirement, which can be inspected by clicking the corresponding hyperlink (arrow 1). The resulting web page shows the properties of the `absFeedbackStatus` requirement, from which the `panelDesign` systems engineering view can be opened (arrow 2). The I/O module is also linked to software models via its interface: see arrow 3. The `SM_CoverPanel` PLC function block has been fully modeled using the Ontoscript DSL and can therefore be illustrated in the web browser. It should be noted that a large fraction of the `SM_CoverPanel` model is the result of a model transformation implemented in the Ontoscript DSL. For instance, in this model we only specified some essential information about the `startOpening` process. When running the DSL script, the model is expanded automatically with additional software features (methods, variables, struct definitions) and even an implementation. As can be seen on the figure, every variable of this implementation can be clicked. In other words: the model is aware of each individual variable and each individual operation of this implementation. Running the ontoscript models thus generated a **model** of the implementation, not just some **text**. One of the advantages is that we can use the same model to generate source code instead of web pages. Using the library view (arrow 4) we can convert the model into source code that can be loaded in a commercial PLC programming environment (arrow 5) or into a Python project (arrow 6). The generated Python source code contains an OPC UA (OPC Unified Architecture) information model based on our in-house developed Unified Architecture Framework (UAF[7]). One can see that with just three Python instructions, we are able to read any variable exposed by the main PLC of the Mercator Telescope.

[5] `http://www.makotemplates.org`
[6] `http://www.pylonsproject.org`

CONCLUSIONS

Several subsystems of the Mercator Telescope have been developed using the presented tools, and are in operation. The models (and the systems they represent) currently consist of 55 I/O module instances, 159 PLC function block definitions, and 626 PLC function block instances. Based on these experiences, we attempt to answer the initial question: why semantics matter?

1. *Because any piece of information is just one query "away".* Semantic models expose the relationships between all kinds of (electrical, software, systems, ...) information and thereby facilitate organizing, integrating, browsing and finding (querying) information.

2. *Because well defined semantics allow verification.* Semantic models allow "implicit" information to be inferred from the explicit information captured by the models, thereby allowing constraints verification.

3. *Because the same semantics may be used for building future "smart" systems.* Semantic modeling languages are a key enabling technology for future intelligent systems. One can imagine a newly developed astronomical instrument being able to expose its capabilities and to "learn" about the capabilities and services of its environment (e.g. observatory or lab), in order to automate integration or even collaboration with "fellow" instruments.

REFERENCES

[1] W. Pessemier et al., "A practical approach to ontology-enabled control systems for astronomical instrumentation", Proc. ICALEPCS 2013, San Francisco, October 2013, TUCOCB03 (2013).

[2] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*, 2nd Ed., (Waltham, MA: Morgan Kaufmann Publishers, 2011).

[3] J. M. Filgueira, "GMT software and controls overview", Proc. SPIE 8451, Amsterdam, July 2012, 845111 (2012).

[7] `http://github.com/uaf/uaf`

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

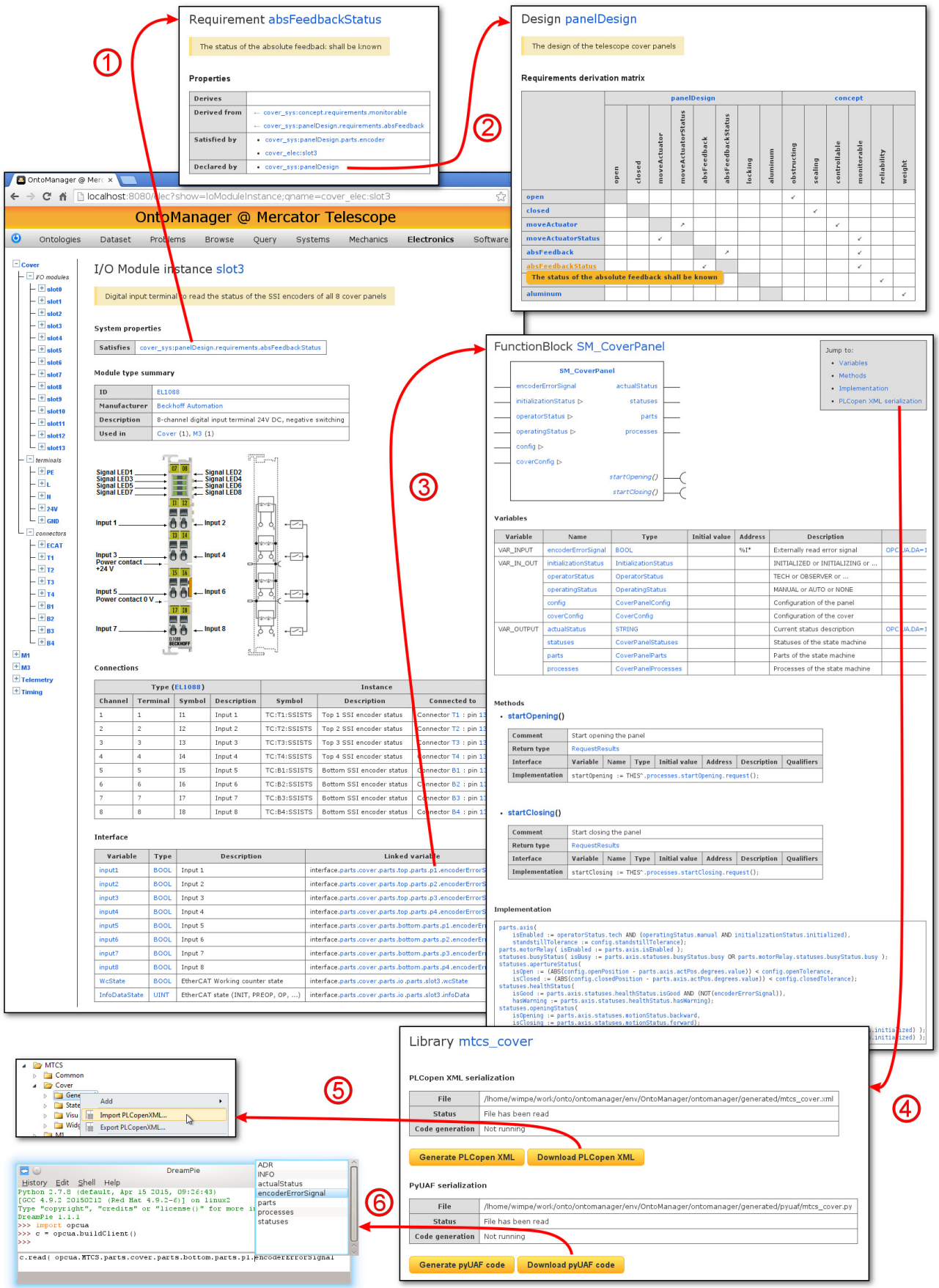


Figure 5: Example of some views, all linked to the I/O module instance “slot3”.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TRIGGER AND RF DISTRIBUTION USING WHITE RABBIT

T. Włostowski, J. Serrano, G. Daniluk, M. Lipiński, CERN, Geneva, Switzerland
F. Vaga, University of Pavia

Abstract

White Rabbit is an extension of Ethernet which allows remote synchronization of nodes with jitters of around 10ps. The technology can be used for a variety of purposes. This paper presents a fixed-latency trigger distribution system for the study of instabilities in the LHC. Fixed latency is achieved by precisely time-stamping incoming triggers, notifying other nodes via an Ethernet broadcast containing these time stamps and having these nodes produce pulses at well-defined time offsets. The same system is used to distribute the 89us LHC revolution tick. This paper also describes current efforts for distributing multiple RF signals over a WR network, using a Distributed DDS (Direct Digital Synthesis) paradigm.

SYNCHRONIZATION IN WHITE RABBIT

Both described systems require a precise time and frequency reference, provided by a WR network and locked to a GPS receiver or an atomic clock. WR achieves its sub-nanosecond accuracy and low jitter by employing multiple techniques:

- Distribution of frequency reference encoded in the physical data stream (Synchronous Ethernet).
- Coarse clock offset measurement by timestamping Ethernet frames using the Precision Time Protocol (PTP, IEEE1588).
- Fine offset compensation by tracking the phase shift between the outgoing and incoming clock/data stream.

Further details on the synchronization algorithms used in WR can be found in [1].

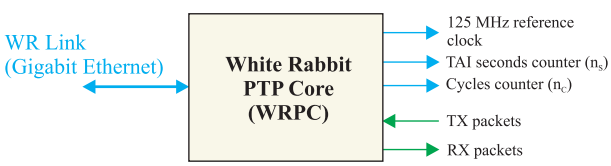


Figure 1: The White Rabbit PTP Core (WRPC).

The RF and Trigger distribution systems are interfaced with the WR network by the WR PTP Core (WRPC) [2], a standardized HDL block implementing the full WR synchronization stack. The WRPC is integrated in the FPGA firmware and delivers a 125 MHz reference frequency and time-of-day (counters of seconds n_S and 125 MHz clock cycles n_C). Furthermore, the WRPC can be used as general-purpose Ethernet Media Access Controller (MAC), sending and receiving Ethernet frames coming from the user's FPGA design (see Figure 1).

RF DISTRIBUTION

Introduction

The RF distribution system is part of a larger project which aims to merge the two historically separate timing systems:

- General purpose timing, usually referenced to the Universal Coordinated Time (UTC) timescale, which usually drives beam injection/extraction, synchronizes currents in the magnets and provides UTC-traceable timestamps for any sort of events that may occur in the machine.
- Beam-synchronous timing, using the bunch or RF frequency as the reference. Its natural applications are driving the RF cavities, synchronizing data acquisition from beam instrumentation or providing precise collision timestamps for event reconstruction (e.g. in the LHC Experiments).

The classic way of distributing a frequency reference in a timing system is to encode it in the data stream (using Manchester or 8B10B encoding) at the master node and recover it in the slave nodes using a Clock-Data Recovery (CDR) PLL. Examples of such timing systems can be found in [3] and [4]. This approach, however, can provide only one reference frequency per each physical link, necessitating the use of two separate timing networks. In many cases the machine's equipment needs simultaneous access to both timing systems, resulting in duplication of cabling and electronics.

Furthermore, many traditional systems experience troubles tracking large frequency changes (e.g. ramping of the RF in low energy or ion machines), due to bandwidth limitations of the CDR PLLs in the deserializer chips.

The method we developed does not rely on physical frequency encoding, but uses the common notion of time and frequency provided by WR to drive RF synthesis in each node of the system, as depicted in Figure 2. The master node keeps is local DDS phase-locked to the RF reference input and broadcasts the DDS tuning values calculated by its PLL over the WR network. The slaves simply feed the received data to their local DDS synthesizers. Since the reference clocks are identical, the DDS in the slave node produces an exact copy of the RF clock coming to the master node.

RF Encoding and Broadcasting

The RF encoding done by the master node is illustrated in Figure 3. The central element of the system is the DDS synthesizer, which produces a sinusoidal signal of arbitrarily controlled frequency and phase. The synthesizer we used is a custom-designed FPGA core attached to a high speed DAC (Digital to Analog Converter). It employs the classical frequency synthesis approach, using a phase accumulator

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

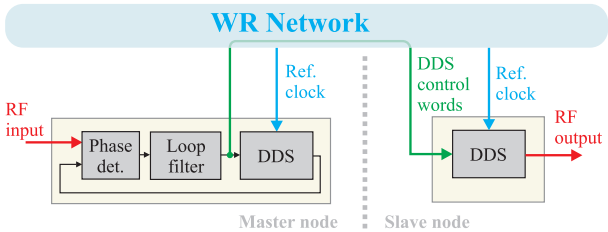


Figure 2: Idea of the RF distribution over White Rabbit.

and a sine lookup table, as described in [5]. We decided to implement a custom synthesizer since in-phase RF reconstruction requires taking snapshots of the DDS phase accumulator at fixed moments in WR time, a feature not available in low-cost integrated DDS+DAC chips.

The core runs synchronously to the 125 MHz WR reference clock F_{ref} and at each clock cycle produces four samples (phase shifted by one-fourth of the F_{ref} period), forming a 500 MSPS data stream for the DAC. In order to improve the dynamic range and spectral purity of the output signal, we used linear interpolation of the sine lookup table values and dithering of the sine signal with noise coming from a pseudo-random number generator.

The output of the DAC passes through an anti-aliasing low-pass filter, which limits the bandwidth of the system to approximately 70 MHz, followed by a zero-crossing detector which produces a square-wave clock signal of frequency f_{DDS}.

The DDS is programmed to a user-specified center frequency F_C, which is continuously adjusted to keep a fixed phase with respect to the varying input RF reference F_{in}. The input can be divided by a user-programmable integer value, extending the frequency range above the bandwidth limit of the DDS.

The adjustment is performed using a typical PLL scheme: the clocks are compared by a Phase-Frequency Detector (PFD) and the resulting phase error is fed to a PI (Proportional-Integral) regulator. The PI outputs the frequency adjustment (tune) of the DDS f_{tune}, closing the feedback loop. In our implementation the PLL consists of a discrete analog PFD followed by an Analog-to-Digital converter (ADC) which drives a digital PI controller. The analog PFD has been chosen to minimize the jitter.

As the RF frequency does not change very rapidly, the controller can sample the phase error and produce a new tune value with a relatively slow period T_S (up to several kHz). The PLL works synchronously to the beginning of the current second (e.g. the first tune update is done when $n_C = 0$, the second one when $n_C = 1 \cdot T_S/8$ ns and so on).

The values produced by the feedback loop are encapsulated into Etherbone [6] packets, comprising:

- The current tune value and the timestamp it was taken at. To simplify calculations, we send the current second and the index i_S of the tune sample with respect to the beginning of the current second.
- The value of the DDS accumulator ϕ_M at the beginning of the current second (when $n_C = 0$), allowing the slave node to compensate for the phase drift of the RF clock caused by the latency of the Ethernet link.
- Configuration parameters: T_S, f_C and the identifier of the master node to auto-configure the slave and allow for multiplexing different RF signals within the same network.

These messages provide all the information necessary for the slave nodes to reproduce the RF signal.

Reception and Reconstruction

RF reconstruction is done at the slave side using the following algorithm:

1. Set a fixed reconstruction latency N_{REC}, which defines by how many samples of f_{tune} the slave's RF output is delayed with respect to the master RF signal. The latency must be obviously larger than the signal processing and packet transmission/reception delays. In the test system, we used $N_{REC} = 3$ tune samples, corresponding to a 300 μs delay.

2. Filter the incoming packet stream to select only the messages containing information about the RF signal the slave is subscribed to.

3. Calculate the initial DDS accumulator value to align the phase of the slave's RF output with the master. Since phase is the integral of frequency, the slave's approximate output phase after N_{REC} tune samples can be derived from the last N_{REC} tune values (assuming that the distributed frequency remains stable over the period of N_{REC} samples):

$$\phi_S = \phi_M + \sum_{i=1}^{N_{REC}} \frac{T_S}{f_{tune}[i]} \qquad (1)$$

4. Apply the phase correction by overwriting the DDS accumulator with the value of ϕ_s.

5. Continuously feed the slave's DDS with the received tune values after delaying them by N_{REC} samples. For example, the first sample produced by the master (having $i_S = 0$) is written to the slave's DDS when $n_C = \frac{N_{REC} \cdot T_S}{8ns}$.

As the result, the slave's DDS produces an in-phase copy of the master RF clock. Note that the current algorithm compensates the phase shift only once, so the phase will drift over time due to changes of the RF frequency. A continuous phase compensation mechanism is currently being developed.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

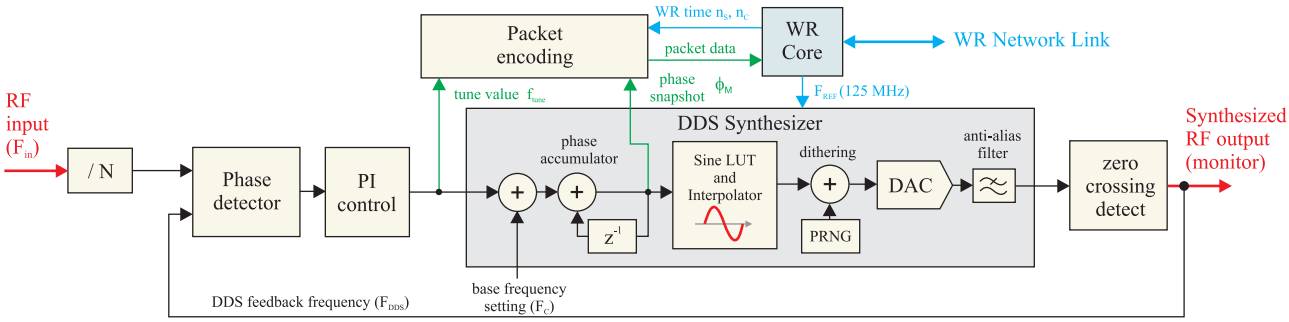


Figure 3: Architecture of the DDS master node.

TRIGGER DISTRIBUTION

The WR Trigger Distribution (WRTD) system was designed to provide transport of trigger pulses between the devices involved in the LHC beam instability diagnostics. The WRTD receives a trigger from a "cloud" of devices (e.g. the Base Band Tune Monitor or the Transverse Damper) upon the onset of an instability and distributes it to all relevant devices (e.g. the Beam Pickup oscilloscopes) to freeze their acquisition buffers, with low and fixed latency. The assignment of trigger outputs to trigger inputs as well as the latency is configurable by the user.

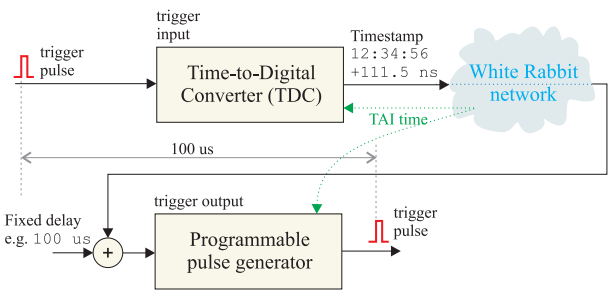


Figure 4: Trigger distribution over White Rabbit.

The concept of WRTD is shown in Figure 4:

- A pulse comes from a source device and is time tagged using a Time-to- Digital Converter (TDC).
- The timestamp produced by the TDC is broadcast over the WR network, with a user- assigned identifier, allowing to uniquely identify the device the trigger came from.
- A node interested in that trigger takes its origin timestamp, adds a certain (fixed) latency and produces a pulse at the calculated moment.

Since all the nodes in the network are synchronized by WR, the measured delay between input and output pulses is equal to the value added to the origin timestamp.

Functionality

The current set of features comprises:

- Nodes in VME64x format, each with 5 inputs and 4 outputs.
- An output can be simultaneously assigned to 128 trigger inputs, each with independently programmable delay.
- Single-shot and continuous triggering.
- Programmable input and output dead time.
- All parameters of triggering can be changed during run-time.
- Extensive diagnostic features: detailed status of each channel, injection of software trigger messages and logging of all trigger events (including triggers missed due to excessive latency).

IMPLEMENTATION

Hardware

Both presented systems are based on the CERN BE-CO-HT's Standard Hardware Kit [7] - a collection of carriers and FPGA Mezzanine (FMC) boards. We chose the Simple VME64x FMC Carrier board (SVEC, [8]), hosting the FPGA that runs the DDS/WRTD firmware and the three mezzanines, interfacing the FPGA with the physical signals:

- DDS FMC [9], with a 500 MSPS DAC, a phase detector and clock distribution and conditioning circuits. This card provides the hardware part of the RF Distribution system.
- TDC FMC [10], a 5-channel TTL input Time-To-Digital converter with 81ps resolution, used by the WRTD trigger inputs.
- Fine Delay FMC [11], used as a programmable 4-channel TTL pulse generator with 10 ps resolution, outputting triggers in the WRTD.

FPGA AND SOFTWARE

The FPGA architecture is based on a mix of dedicated VHDL IP cores and embedded soft CPUs, provided by the *Mock Turtle* framework [12] and interconnected through the Wishbone bus [13].

The trigger distribution FPGA firmware was built using unmodified Fine Delay and TDC Cores [11] [10]. Processing of the network messages, configuration and delay adjustment

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

is handled by the soft CPUs. One processor is responsible for the trigger inputs and the other for the trigger outputs, as shown in Figure 5. No additional VHDL development was necessary except for connecting the blocks together.

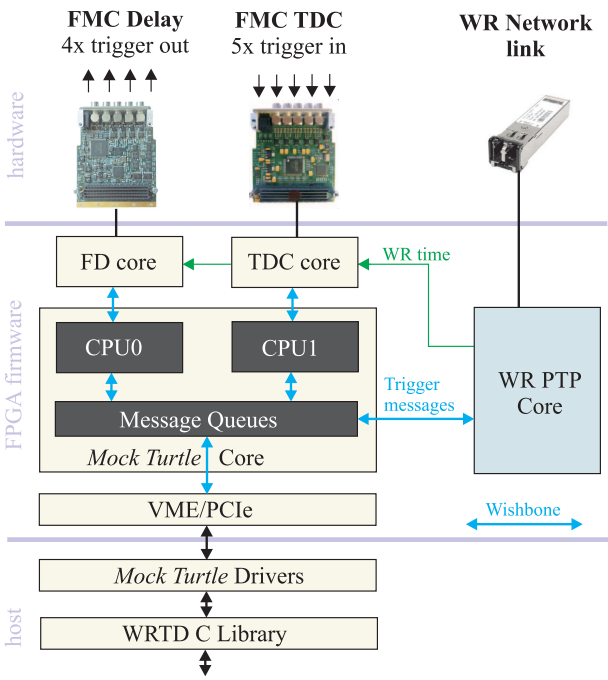


Figure 5: Architecture of the WRTD Hardware, FPGA and software.

In the case of the RF distribution system, the VHDL core provides only the DDS synthesizer and interfaces to the clock distribution circuit and the phase detector. The complex part: the entire feedback loop, packet encoding and phase compensation is incorporated in the embedded software running on the CPU, as its processing power and determinism are sufficient for the required latencies (300 μs) and deadlines (PLL update every 100 μs).

The carrier-mezzanine abstraction model we employed made the designs independent from the host platform. Initially developed for VME64x, both systems can be easily ported to other platforms (PCI Express or PXIe) by simply changing a carrier-dependent VHDL template module.

The software stack consists of bare-metal (i.e. without an operating system) real time applications that run on the embedded CPUs, and user-space libraries providing the APIs for trigger management (WRTD) and DDS configuration (RF distribution). Both of them are built on top of the *Mock Turtle* software, which provides a real-time development environment, as well as a generic Linux device driver/library to communicate between the host system and the real-time applications.

PERFORMANCE

RF Distribution

Preliminary measurements were performed for f_{in} = 352 MHz, f_{DDS} = 44 MHz (input divided by 8) and an update period T_S = 100 μs (10 kHz sampling rate). The system provided phase alignment of the slave node better than 1 nanosecond and rms jitter (100 Hz - 10 MHz) of 20 picoseconds. The largest contribution to the jitter comes from sub-optimal design of the WR reference clock generation circuit on the DDS mezzanine, which will be improved in the next version of the hardware, with an expected jitter of 3 ps rms.

Trigger Distribution

The number of trigger sources in the system and the trigger repetition rate is limited by the network bandwidth and maximum acceptable latency requirements. The system installed in the LHC is specified to transport 35000 trigger pulses per second, which occupies approximately 5% of the bandwidth of a Gigabit Ethernet link at the worst-case latency of 300 μs (out of which, 200 μs is allocated for the delays introduced by the fibers). This allows for distribution of the 89 μs revolution tick using the WRTD network, for the purpose of realigning asynchronously generated triggers with the machine's bunch crossing frequency.

PROJECT STATUS AND OUTLOOK

The RF distribution is part of a larger effort to design a unified timing system, incorporating the general purpose and beam synchronous timing in a single standardized network. We have already proven the feasibility of transferring an RF clock over an Ethernet link. In order to provide a complete timing system, we are currently developing the event distribution and reception infrastructure.

WRTD is a mature system, working in operational installations in the LHC and being integrated in the CERN Controls software stack. Furthermore, the OASIS project at CERN [14], which provides acquisition, correlation and visualization of analog signals (a "distributed oscilloscope") is developing a new WRTD-based infrastructure. WRTD will gradually replace the current trigger distribution system based on a central trigger multiplexer and discrete cabling.

The RF and trigger distribution are only two examples of applications of WR technology to solve real problems in distributed real-time controls and data acquisition. In our experience, the availability of a set of modular, reusable open source hardware building blocks greatly reduces development time and allows us to better fulfill the needs our users.

REFERENCES

[1] J. Serrano, M. Cattin, E. Gousiou, E. van der Bij, T. Włostowski, G. Daniluk, M. Lipiński, "The White Rabbit Project", IBIC2013, Oxford, UK (2013).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

[2] G. Daniluk, "White Rabbit PTP Core: the sub-nanosecond time synchronization over Ethernet", *M.Sc. thesis*, Warsaw University of Technology (2012), http://www.ohwr.org/documents/174

[3] Micro-Research Finland Oy website: http://www.mrf.fi/index.php/timing-system/70-timing-system-structure

[4] D. Domínguez, J.J. Gras, J. Lewis, J.J. Savioz, J. Serrano, F.J. Ballester, "An FPGA-based Multiprocessing CPU for Beam-Synchronous Timing in CERN's SPS and LHC", ICALEPCS2003, Gyeongju, Korea (2003).

[5] Analog Devices, "A Technical Tutorial on Digital Signal Synthesis" (2009), http://www.ieee.li/pdf/essay/dds.pdf

[6] M. Kreider, R. Baer, D. Beck, W. Terpstra, J. Davies, V. Grout, J. Lewis, J. Serrano, T. Włostowski, "Open borders for system-on-a-chip buses: A wire format for connecting large physics controls", Phys. Rev. ST Accel. Beams, vol. 15 (2012).

[7] E. Van der Bij, M. Cattin, E. Gousiou, J. Serrano, T. Włostowski, "CERN's FMC Kit", ICALEPCS2013, San Francisco, USA (2013).

[8] Simple VME64x FMC Carrier project homepage: http://www.ohwr.org/projects/svec/

[9] FMC DDS project homepage: http://www.ohwr.org/projects/fmc-dac-600m-12b-1cha-dds/wiki

[10] FMC TDC project homepage: http://www.ohwr.org/projects/fmc-tdc-1ns-5cha-hw/wiki

[11] FMC Fine Delay project homepage: http://www.ohwr.org/projects/fmc-delay-1ns-8cha/wiki

[12] T. Włostowski, J. Serrano, F. Vaga, "Developing Distributed Hard-Real Time Software Systems Using FPGAs and Soft Cores", THHA2I01, these proceedings, ICALEPCS'2015, Melbourne, Australia (2015).

[13] Wishbone bus specification, version B.4: cdn.opencores.org/downloads/wbspec_b4.pdf

[14] S. Deghaye, L. Bojtar, C. Charrondiere, Y. Georgievskiy, F. Peters, I. Zharinov, "OASIS Evolution", ICALEPCS2007, Knoxville, USA (2007).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE PHASE-LOCKED LOOP ALGORITHM OF THE FUNCTION GENERATION/CONTROLLER

M. Magrans de Abril, Q. King, R. Murillo Garcia, CERN, Geneva, Switzerland

Abstract

This paper describes the phase-locked loop algorithms that are used by the real-time power converter controllers at CERN. The algorithms allow the recovery of the machine time and events received by an embedded controller through WorldFIP or Ethernet-based fieldbuses. During normal operation, the algorithm provides less than 10 µs of time precision and 0.5 µs of clock jitter for the WorldFIP case, and less than 2.5 µs of time precision and 40 ns of clock jitter for the Ethernet case.

INTRODUCTION

It is widely known that the existence of measurement and computation delays and jitter affect the stability and performance of the control algorithms [1–3]. It is also well known that as the control system becomes larger (i.e. spatial extension or number of jointly controlled elements), the complexity of the control algorithm increases and its performance and stability degrades [4]. This is precisely the situation of the control of power converters at CERN. For example, the LHC project required the joint control of over 1700 converters across the 27 km of underground tunnels with a tracking error of less than five part per million (ppm). The large spatial extension requires a correspondingly large communication network to transport the information, which increases the delay, jitter, and packet loss. If there are different subsystems to be jointly controlled (e.g. main dipole and quadrupole magnets, simultaneous setting of the real-time orbit corrections, synchronisation of injection and ejection converters, etc.), then the control algorithm should also consider the relative delays between them.

In order to minimise the above effects, the devices that require real-time synchronisation at CERN used to employ a special purpose communication network that transports timing information with small jitter and known delay [5, 6]. The information necessary to control a device is therefore sent using two different networks. An Ethernet network (known as the technical network) transports the soft real-time data. A second network (known as timing network) transports the hard real-time data containing such things as the machine time, accelerator state and events.

Since the year 2000, the power converter controllers known as Function Generation Controllers (FGC) have deviated from this general architecture [7]. As shown in Fig. 1 the FGC uses the fieldbus as the mechanism to receive both soft and hard real-time data from the general purpose Linux computer (known as the front-end). That is, the control and the timing networks converge on a front-end, instead of being connected directly to the embedded controller. This change reduces the installation cost and the number of connection related incidents. In the FGC case, further reduction of development, installation, and maintenance costs have been achieved by using commercial off-the-shelf components and protocols for the fieldbuses. That is, WorldFIP on the FGC version 2 (FGC2), and FGC_Ether on the FGC version 3 (FGC3) [8].

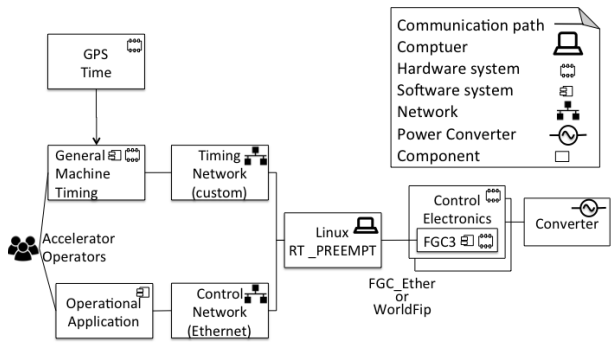


Figure 1: Transport of timing and control information from the GPS signal and the particle accelerator operator to the power converter using one fieldbus to transport timing, control, and monitoring information.

Removing the direct connection between the timing network and the FGC implies that the UTC time and its phase have to be recovered. The time with 20 ms precision is recovered by a periodic broadcast signal from the general purpose computer to the FGC, while the 20 millisecond phase is recovered using a Phase-Locked Loop (PLL) running on the FGC. This paper describes the requirements, design, and performance of the PLL algorithms used for both the FGC2 and FGC3.

SYNCHRONISATION OF THE LHC CONVERTERS

Requirements and Design

Between the year 2000 and 2003 the FGC2 was developed for the LHC. On one hand, the correct operation of the LHC required a radiation tolerant fieldbus and the synchronised application of the converter current across the accelerator with an accuracy of less than 5 ppm. This requires a time precision of 1 ms to achieve the synchronised operation of the converters [9]. It also required a jitter of less than 10 µs to have less than 1 ppm jitter induced noise from the sigma-delta ADC. On the other hand, it was not of primary concern the control and monitoring bandwidth between the operator and the controller because the circuits and the ramps in current are very slow. These requirements were fulfilled by a 2.5 Mbit/s WorldFIP fieldbus cycling at 50 Hz. Given the low jitter of this real-time fieldbus (i.e. $\sigma_j \approx 0.5\,\mu s$),

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

it has been possible to reconstruct the 20 ms phase with respect to the timing network using a periodic broadcast packet containing the UTC time and the accelerator events.

Figure 2 shows the block diagram of the software PLL algorithm. The implementation relies on a free running 16-bit 2 MHz counter provided by the Motorola M68HC16 CPU. First, the fieldbus synchronisation signal $f(t)$ latches the free running counter $c[m]$, and generates the sync reference $r[n]$. Every 20 ms the reference is compared against the expected sync time $y[n]$ to compute the phase error $e[n]$. The error is then used to adjust the expected millisecond period in terms of the CPU 2 MHz tics $q[n]$ using a Proportional Integral (PI) controller with gain scheduling (the PI is the simplest controller that can follow the phase ramp generated by the fieldbus time signal). The millisecond Interrupt Service Routine (ISR) uses the current value of the free running counter $c[m]$ and the expected millisecond duration to generate the next millisecond interrupt $m(t)$. As the CPU counter comparator only handles integers, the fractional part $q*[n] - floor(q*[n])$ is carried over to the next iteration to minimise the quantisation error.

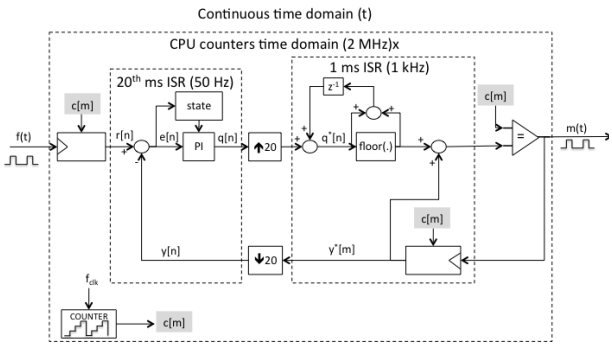


Figure 2: FGC2 PLL Algorithm.

Model

Figure 2 can be analysed using a linear module using the following assumptions: the output counter generating the millisecond interrupt can handle real numbers, the quantisation noise is much smaller than the fieldbus jitter, and finally, the FGC2 clock drift rate is small compared to the fieldbus jitter. In this case, the z-transform model of the PLL is:

$$\begin{aligned} E(z) &= \frac{1}{1+L(z)}R(z) \\ Y(z) &= \frac{L(z)}{1+L(z)}R(z) \\ L(z) &= \frac{20(k_i+k_p)}{(z-1)^2}(z-\frac{k_p}{k_i+k_p}) \end{aligned} \quad (1)$$

Where $L(z)$ is the open loop transfer function, k_i and k_p are the integral and proportional gains of the PI controller, $E(z)$ is the z-transform of the phase error, $R(z)$ is the reference input (i.e. a ramp plus white noise representing the jitter), and finally, $Y(z)$ is the z-transform of the expected arrival time.

The relation between the input jitter and the phase error is straightforward [10]:

$$\begin{aligned} S_{ee}(f) &= |H_{je}(e^{j2\pi f})|^2 S_{jj}(f) \\ &= |\frac{1}{1+L(e^{j2\pi f})}|^2 \sigma_j^2 \\ \sigma_e^2 &= \sigma_j^2 \int_0^1 |\frac{1}{1+L(e^{j2\pi f})}|^2 df \end{aligned} \quad (2)$$

Where S_{ee} and S_{jj} are the power spectrum of the jitter and phase error in the discrete frequency domain, $H_{je}(z)$ is equal to the z-transform transfer function between the reference and the phase error, and finally, σ_e and σ_j are the respective phase error and jitter standard deviations measured in 2 MHz tics.

Note that the diagram in Fig. 2 obviates the propagation of the fieldbus boradcast packet. The PLL design assumes a fixed propagation delay of 450 µs. Therefore, given that the maximum difference in cable length across the LHC is less than 2 km, the time accuracy should be better than 10 µs.

Stability and Performance

There are three basic criteria that should be met in the design of a PLL: stability, lock time, and jitter rejection. A PLL is stable if the magnitude of the closed loop poles is less than 1. As shown in Fig. 3 the proportional and integral gains should be well below 1/10. The lock time is determined by the decay of the transient response, which in turn is determined by the magnitude of the open loop poles. That is, smaller pole magnitude implies faster lock time and also a more stable PLL. Finally, jitter rejection can be calculated using eq. 2. Figure 4 shows the logarithm of the jitter rejection as a function of the proportional and integral gains. A minimum jitter requires the smallest possible integral gain, and a proportional gain about 0.005. So, stability and jitter rejection are at odds with each other. In order to bring together both requirements and have a lock time below 10 seconds, a gain scheduling algorithm was implemented. The PLL state machine is composed of three states:

- *Fast Slew* ($|e[n]| > 1$ ms). Instead of a PI algorithm the millisecond duration is fixed to $q[n] = 2100$ ticks (1.05 ms). This state allows the phase error to reach a close initial condition in less than half a second without the risk of overrunning the millisecond ISR.
- *Capture* ($50\mu s > |e[n]| \geq 1$ ms). A PI algorithm is used with $k_i = 2^{-12}$ and $k_p = 2^{-8}$.
- *Lock* ($|e[n]| < 50$ µs for more than 1.5 seconds). A PI algorithm was used with $k_i = 2^{-16}$ and $k_p = 2^{-9}$.

Simulation and Measurements

Figure 5 shows the simulated and measured phase errors in µs after a power cycle of the FGC2. The picture shows three differentiated phases corresponding to the three different states of the PLL. Note that the simulation shows a longer period in the Fast Slew state due to a difference in the initial values of the free running counters. Note that the PLL locks

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

in less than 3 seconds with a phase error jitter with σ_e = 0.5 µs.

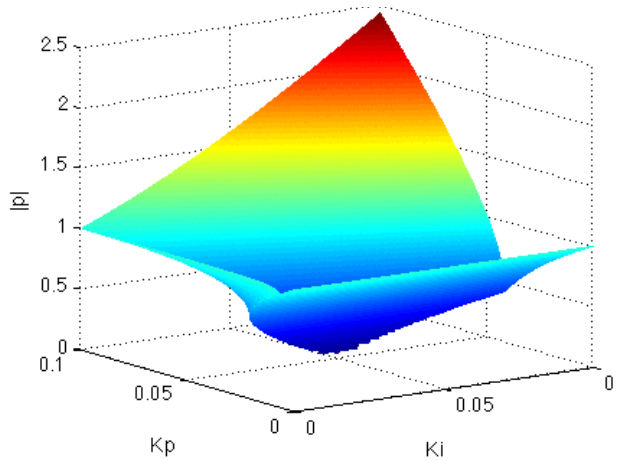


Figure 3: Magnitude of the closed loop poles of the FGC2 PLL algorithm.

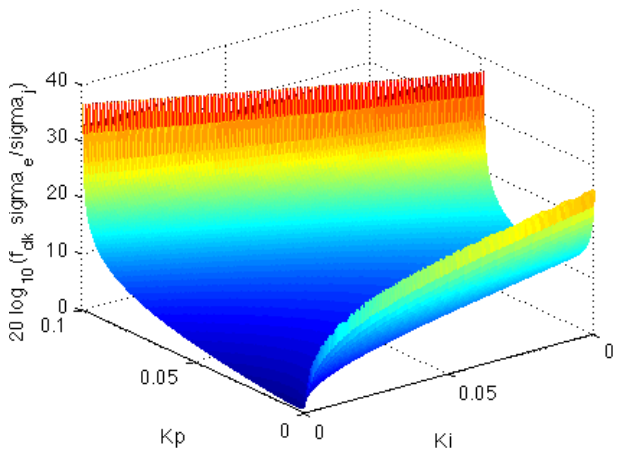


Figure 4: Jitter rejection logarithm of the FGC2 PLL algorithm.

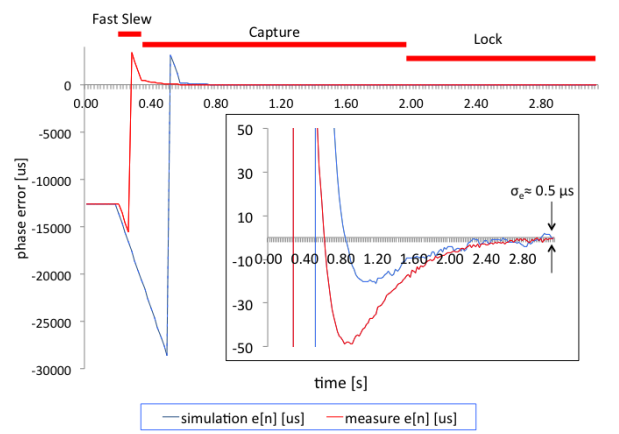


Figure 5: FGC2 PLL phase error after a power cycle.

SYNCHRONISATION OF THE INJECTOR CONVERTERS

Requirements and Design

The FGC3 controller was developed between the years 2007 and 2011 for the Proton Synchroton Booster (PSB) and Linac 4 accelerators. The FGC3 will be also used for the new projects at ISOLDE, AD, ELENA, and the renovation of the power converter controls across the LHC injection chain. During the early phases of the project it was clearly appreciated that the bandwidth and timing requirements of the injector chain were more demanding than the ones for the LHC [7]. A time precision of 1 µs and a jitter of 100 ns was required.This allowed a higher regulation regulation frequency (10 kHz), the short (1-10 ms) and high precision ramps (100-1000 ppm) on some of the injector magnets, the reliable propagation of timing information from the FGC3 to the rest of the converter control electronics (e.g. voltage control loop), and finally, the synchronised setting of the current between converters.

For this purpose, the FGC_Ether fieldbus was developed, allowing not just a 100 Mbps bandwidth between the general purpose computer and the FGC3, but also the distribution of a low jitter ($\sigma_j \approx 10$ ns) 50 Hz signal from the timing network to the FGC3 using the same Ethernet cable [8]. The 50 Hz phase was then recovered with a hybrid PLL using an IQD 25 MHz VCXO to clock the free running counter on the FPGAmk. An FPGA was used as phase comparator. Finally, a Renesas RX610 CPU runs the 20 ms ISR to perform the filtering and control calculations of the PLL algorithm.

Figure 6 shows the FGC3 PLL algorithm. The reference $r[n]$ is calculated internally every 20 ms by adding 800,000 tics (20 ms) to the last reference value. This reference is compared with the 25 MHz counter latched by the arrival of the FGC_Ether 50 Hz signal $ext(t)$ (or time packet in case the 50Hz sync pulses are temporarily missing). The phase error $e_{ext}[n]$ is smoothed using an averaging filter of length 10, and the reference is fed to the PI algorithm every tenth iteration. The PI output $q[n]$ is limited by the state machine to the maximum VCXO pullability of ±100 ppm, and sent to the FPGA to set the frequency offset. The FPGA upsamples the PI output to 625 KHz to recover its fractional part, and to increase the theoretical resolution of the 14-bit DAC by 8 additional bits. The VCXO frequency is then used as a clock for the free running counter $c[m]$, and also as a clock for the FGC3 CPU and DSP.

The linear analysis of the algorithm follows exactly the one for the FGC2 PLL by assuming that the DAC has an infinite resolution, and taking into account that the down-sampled and filtered noise, $e[n]$, is uncorrelated. The open loop transfer function is:

$$L(z) = \frac{K_{vcxo}(k_i + k_p)}{(z-1)^2}(z - \frac{k_p}{k_i + k_p}) \qquad (3)$$

Where $K_{vcxo} \approx 0.001$, the regulation frequency is 5 Hz, $k_i = 0.003$, and $k_p = 0.05$. The integral and proportional

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

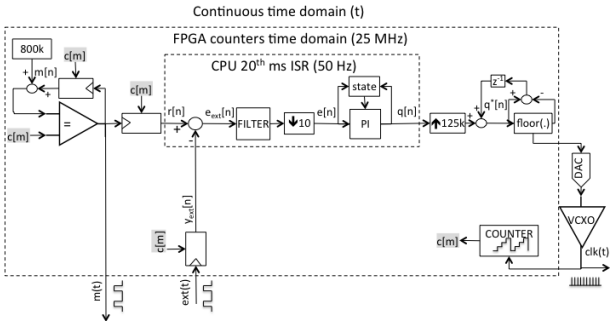


Figure 6: FGC3 PLL Algorithm.

gains have been chosen to minimise the jitter while keeping the lock time under 2-3 seconds. Note that the loop gain is much lower than the FGC2, which corresponds to the slower lock time constant, and the higher accuracy.

It is also worth mentioning that as in the FGC2 case, we have obviated the compensation of the Ethernet and 50 Hz signal propagation delay over the FGC_Ether fieldbus. In the FGC3 case, there is no compensation on the 50 Hz signal, and the propagation delay of the Ethernet packet is compensated using the average offset with respect to the 50 Hz signal. Given that the distance between the computer and the FGC3 in the injectors is smaller than 500 m, the time precision is better than 2.5 µs, and adjusting the cable length it is easy to achieve 100 ns.

Simulation and Measurements

The FGC3 PLL algorithm achieves a phase error jitter with $\sigma_e < 40$ ns when the 50 Hz signal is used, and $\sigma_e \approx 0.5$ µs when the time of arrival of the Ethernet packet is used (degraded mode). In order to achieve a small and stationary jitter on the Ethernet arrival time, careful attention should be put on the real time thread sending the packet on the Linux front-end: the power saving features of the CPU and operating system should be disabled, the load of the computer should be as low as possible, the thread handling the synchronous send of the Ethernet time packet should minimise the computational load and be as deterministic as possible, the real-time thread should also have the highest priority possible, and finally, the CPU affinity should be fixed. Otherwise, the jitter won't only be higher but it could be non-stationary.

The better FGC3 phase error jitter is achieved by the lower jitter of the 50 Hz signal ($\sigma_j \approx 10$ ns), and the higher resolution of the 25 MHz VCXO with respect to the 2 MHz counter used on the FGC2. On the other hand, the pullability of the VCXO (±100 ppm) is much smaller than the equivalent pullability of the FGC2. This is the price paid for the additional resolution of the actuator. In order to minimise its impact and improve the lock time, two measures were taken. First, the 20 millisecond reference counter, $m[n]$, is initialised to the value of the free running counter at the arrival of the first Ethernet time packet. And second, the VCXO frequency, $q[n]$, is initialised to the previously held value before a reset.

As in the case of the FGC2, the algorithm performance was both simulated and measured using the same source code. Given that the free running counter is initialised by the Ethernet time packet, the time to lock after a power cycle is always the same (< 2 s). Figure 7 shows the simulated and measured phase errors for the worst case scenario where the FGC_Ether fieldbus has been reconnected after being disconnected for several hours. In this case, the phase error has drifted with a speed of below 1 µs/s due to the inaccuracy of $q[n]$ and the relative drifts of the FGC3 and timing network clocks. For this worst case scenario the lock time is around 24 seconds. Although the FGC3 PLL didn't need a PI with gain scheduling, the PLL states were kept for informative purposes. On the plot the Fast Slew phase corresponds to the VCXO being clipped to its maximum pullability, while Lock corresponds to a phase error below 16 µs for more than 2 s.

Note that plot shows two differentiated phases, one where the VCXO input is saturated, and then once the phase difference is small enough, the linear regulator takes over.

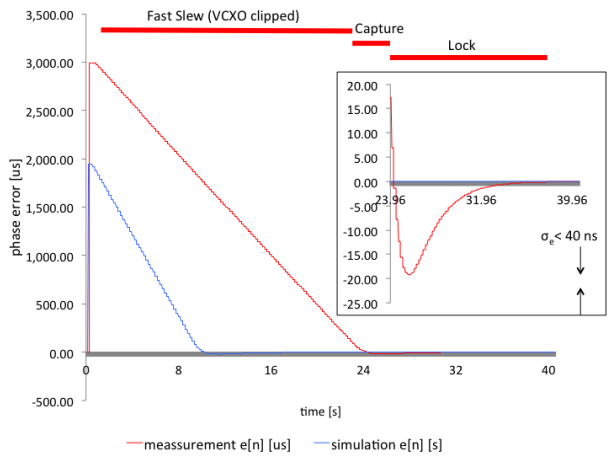


Figure 7: FGC3 PLL phase error after loosing the timing signal for two hours.

CONCLUSION

The PLL algorithms for the FGC2 and FGC3 have been described and their performance exceed the operational requirements of the LHC and its injectors. Although both algorithms use the same control algorithm, the FGC3 PLL departed from the pure software solution chosen for the FGC2. The FGC3 uses an VCXO and a higher frequency counter clock to achieve a more precise lock of the phase. This additional precision can be exploited thanks to the lower jitter of the 50 Hz signal coming from the FGC_Ether interface ($\sigma_j \approx 10$ ns). Instead on the FGC2 the input jitter depends on the time of arrival of the WorldFIP periodic packet ($\sigma_j \approx 0.5$ µs). This additional precision on the FGC3 does not affect the lock time thanks to the proper initialisation of the FGC3 free running counters when the first FGC_Ether packet is received.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

REFERENCES

[1] C. Azeredo-Leme,"Clock Jitter Effects on Sampling: A Tutorial", IEEE Circuits and Systems Magazine, Aug 2011.

[2] Kao, Chung-Yao, et al., "Simple stability criteria for systems with time-varying delays", Automatica 40.8, 2004.

[3] K. J. Astrom, R. M. Murray, "Feedback Systems", Princeton University Press, 2008.

[4] V.D. Blondel and J.N Tsitsiklis, "A survey of computational complexity results in systems and control", Automatica 36, 2000.

[5] J. C. Bau et al., "Managing the real-time behaviour of a particle beam factory: the CERN Proton Synchrotron complex and its timing system principles", IEEE Transactions on Nuclear Science 45, 1998.

[6] J. Lewis et al., "The CERN LHC Central timing, a vertical slice", ICALEPCS, 2007.

[7] D. Calcoen et al., "Evolution of the CERN power converter function generator/controller for operation in fast cycling accelerators", ICALEPCS, 2011.

[8] S. Page et al., "Migration from WorldFIP to a Low-Cost Ethernet Fieldbus for Power Converter Control at CERN", ICALEPCS, 2013.

[9] T. Wijnands, "Functional specification for LHC Power Converter control", LHC Project Note 183, 1999.

[10] J. G. Proakis et al., "Digital Signal Processing", Prentice-Hall, 1992.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

NEW EVENT TIMING SYSTEM FOR DAMPING RING AT SuperKEKB

H. Kaji, K. Furukawa, M. Iwasaki, T. Kobayashi, F. Miyahara, T.T. Nakamura,
M. Satoh, M. Suetake, M. Tobiyama, KEK, Japan
Y. Iitsuka, East Japan Institute of Technology, Japan
T. Kudou, S. Kusano, Mitsubishi Electric System & Service Co., Ltd, Japan
M. Liu, C. Yin, SINAP, China

Abstract

We utilize the Event Timing System to perform the operation of injector for the SuperKEKB accelerators. We construct the Sub-timing Station to manage two kinds of triggers for the Damping Ring devices. We developed new Event modules, VME-EVO and VME-EVE, to deliver triggers to the beam monitors at Damping Ring. The triggers for 84 beam position monitors can be provided with only 5 new Event modules. The timing accuracy of outputs from the new Event modules is ~15 ps. It well satisfies the requirements from the beam monitors at Damping Ring.

INTRODUCTION

SuperKEKB [1, 2] is the electron-positron collider which targets the world's largest record of luminosity. This is the upgrade machine of KEKB [3, 4] and one of the most important projects at KEK. SuperKEKB is designed to achieve 40 times larger luminosity than that of KEKB – it is realized from two times larger storage current and 20 times smaller vertical emittance.

The control system of injector linac (LINAC) [5] is extremely important for both KEKB and coming SuperKEKB. One of key technologies for the LINAC control is Pulse-to-Pulse Modulation (PPM). This injector "simultaneously" performs the top-up filling operations into four independent rings – two rings of SuperKEKB and two light sources [6, 7]. Therefore more than 150 of parameters to operate the LINAC beamline should be changed, pulse-by-pulse, typically in 50 Hz.

The Event Timing System of LINAC [8–10] works for not just the delivery of timing-triggers to the beamline devices but the above-mentioned PPM operation. It forms star topology optical network. The Event Generator (EVG) [11] is placed at Main Timing Station[1] – it is the focal point of optical network. The EVG controls the LINAC devices by delivering various kinds of Events. The Event Receivers (EVRs) [11] are placed at the individual edges of the optical network. The EVRs work for the beamline devices in the two kinds of ways when they receive the Events. One is the launching of triggers, like NIM or TTL, for indicating the timing of injection process. The other is the CPU interruption to switch the parameters of devices for the PPM.

We develop Sub-timing Station on one edge of Event network for the newly constructed Damping Ring (DR). The several Event modules including the newly developed modules are installed at DR Sub-timing Station since the delivery of timing-triggers towards the DR devices during the storage of beam-pulses is complicated.

In this report, we introduce the timing system at DR with the newly developed Event modules.

[1] We call it as Main Trigger Station in the KEKB era.

TRIGGER REQUIREMENTS AT DR

In this section, we explain the requirements to the timing system at DR. We need the special treatments to the triggers for the beam monitors since they are not synchronized with the LINAC operation. We explain the triggers for the beam position monitors (BPMs) as an example.

Operation Scheme of DR

DR is constructed for SuperKEKB. It is utilized in the injection of positron-pulses at LINAC. The frequency of RF cavity at DR is 508.89 MHz and it is same as that at MR. The harmonic number is decided to be 230 for making Bucket Selection of Main Ring (MR) [12, 13] efficient. The positron-pulses are stored at least 40 ms for damping their emittance.

In each positron injection, two of RF-buckets are occupied by the beam-pulses. The RF-buckets to be occupied are changed depending on the Bucket Selection in pulse-by-pulse.

Triggers for BPM

There are 84 BPMs at DR. They work during the storage of positron-pulses and measure the orbit. The timing of trigger for BPM is set by the delay value from the DR revolution as a reference and is independent from the LINAC operation.

The delay value is determined from two components. The individual BPMs have their own delay which are related with the installed positions along the DR beamline. In addition to these individual delays, there is the common delay which is related with the RF-buckets occupied with beam-pulses. The common delay must be set in the 508.89 MHz step since it is the frequency of RF cavities at DR. It should be adjusted, in pulse-by-pulse, because of Bucket Selection.

To validate the certain performance of BPMs, the accuracy (jitter) of the total delay values should be < 2 ns.

Summary of Requirements to the Timing System

Here we summarized the requirements to the timing system for the DR operation.

There are the triggers only for the DR devices like the beam monitors. They are delivered during the storage of

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

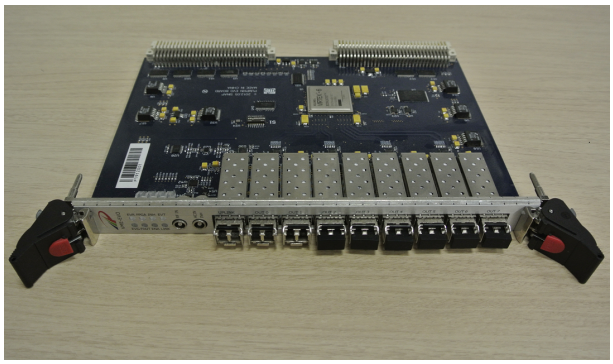


Figure 1: Picture of the VME-EVO module.

positron-pulses. The timing is independent from the LINAC operation and synchronized with the DR revolution.

The timing of triggers for 84 BPMs at DR should be set with the accuracy of < 2 ns. The delays for the individual BPMs consist of their own delays and the common delay which are adjusted in the 508.89 MHz step.

In addition to above "DR specific" triggers, of course, we deliver triggers for indicating timing on the injection (extraction) of beam-pulse to (from) DR. They are utilized for the septum magnets and so on and can be realized by simply connecting EVRs with the EVG at Main Timing Station.

EVENT TIMING SYSTEM AT DR

In this section, we explain the configurtion of Event Timing System at DR. We introduce the new Event modules which are utilized for delivering the triggers towards BPMs. They are the VME type modules originally developed for the proton therapy machine at SINAP [14] and customized for the SuperKEKB project. We confirmed also the new modules are compatible with the KEK's current standard modules, VME-EVG-230 and VME-EVR-230RF [11].

New Event Generator

Figure 1 is a picture of the VME-EVO module. Its specification is summarized in Table 1. The sequence for delivering Events can be triggered by the TTL input or the up-link Event. The operation clock (Event clock) can be set independently from the upstream module when its up-link channel connects with the EVG. The VME-EVO module has 8 TX channels (outputs) so that it does not need the dedicated fanout module.

It is also worth noting, VME-EVO can be utilized also for the EVR or funout modules. It depends on settings.

New Event Receiver

Figure 2 is a picture of the VME-EVE and VME-TTB modules. Its specification is summarized in Table 1. The VME-EVE module has 8 outputs which launch the TTL level triggers. The output channels can be increased by installing the VME-TTB module on the rear slot of VME sub-rack. The VME-TTB module has additional 16 output channels.

Table 1: Specifications for New Modules

	VME-EVO	**VME-EVE**
Type	EVG/EVR/fanout	EVR
Event Clock	60-135 MHz	60-135 MHz
Clock Sync.	RF input	Up-link
Trigger	TTL, Up-link Event	Up-link Event
Output	Optical×8	TTL×8 (×24)
Precision	Event Clock	Event Clock/20

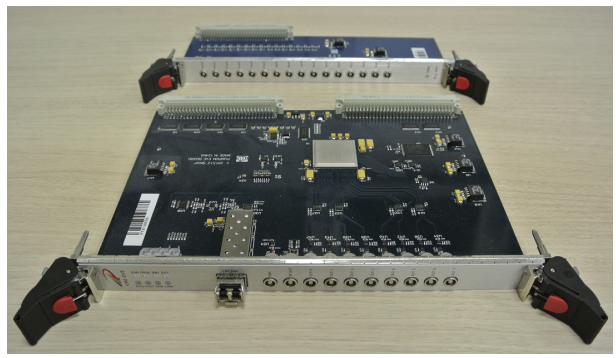


Figure 2: Picture of the VME-EVE and VME-TTB modules.

Therefore one set of VME-EVE and VME-TB can manage totally 24 kinds of TTL level triggers[2].

The trigger delivery is implemented when VME-EVE receives an Event from up-link. The delay can be set on the individual outputs, independently. The individual channels have two kinds of delay functions. One is the Event clock delay. The other is the 20 times more precise step than the Event clock and named "RF delay". The RF delay is provided with the GTX technology.

Overview of Event Timing System at DR

Figure 3 is the schematic view of module configuration for the DR timing system. We install two VME-EVR-230RF modules and one VME-EVO at DR Sub-timing Station.

Two EVRs are connected with two lower-level EVGs of Main Timing Station, respectively. Therefore, one EVR is synchronized on the timing of the positron-pulses injection while the other EVR is synchronized on that of extraction. Their Event clocks are 114.24 MHz and phase-locked by the up-link EVGs.

The VME-EVO module is operated with the Event clock of 127.2225 MHz. The 508.89 MHz clock for the RF cavities at DR is put into the RF-IN channel. It is 4 times divided on the inside circuit and utilized as the reference clock. We put the DR revolution into the AC-IN channel of VME-EVO.

There are 4 local IOCs for the BPMs. In each IOC, the VME-EVE modules are installed with VME-TTB. The individual IOCs manage the triggers for the 21 BPMs. The up-link channels of VME-EVEs are connected with VME-EVO at Sub-timing Station.

[2] Note, now we are developing also the module with the NIM level outputs.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

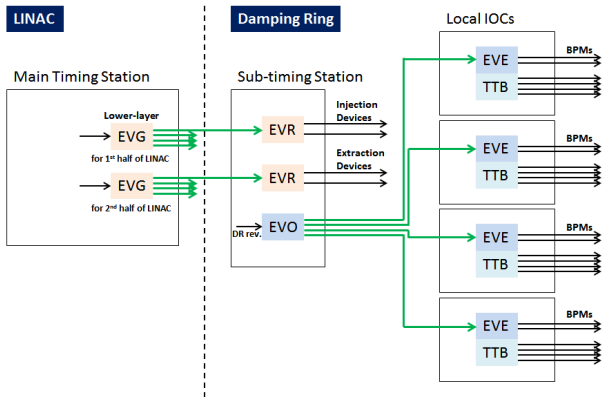


Figure 3: Schematic view of configuration of Event Timing System: the right part from dashed line is the configuration for Damping Ring. We configure DR Sub-timing Station with the three Event modules.

The entire "VME-EVO+VME-EVE" system can provide the triggers with the timing reference of the DR revolution and the arbitrary delays. They are utilized as the triggers for the beam monitors. The two components of delays for the BPM triggers are realized in the following way.

The delay adjustments for individual BPMs are needed to compensate the difference in the their installed positions. They are implemented by the Event clock delay of individual VME-EVE channels.

The common delay is needed to fit the timing of triggers to the storage bunches and adjusted by the Event scheduling on the sequence RAM of VME-EVO. However the Event clock of VME-EVO is 4 times coarser than the required precision, 508.89 MHz, for setting the common delay. Therefore we additionally utilize the RF delay of VME-EVE (and VME-TB) channels. The RF delay can manage 4 times more precise delay since its intrinsic precision is 20 times more than the Event clock.

The module configuration of Event Timing System for the beam monitor at DR is simple even though the complicated timing adjustments are required to the triggers. We can manage the triggers for the 84 BPMs with only 5 Event modules. The triggers for other beam monitors, like the synchrotron radiation monitor, also can be delivered from the similar system.

PERFORMANCE OF NEW MODULES

Here we introduce the timing accuracy (jitter) of TTL outputs from the new Event modules as an example of their basic performance.

The test system is configured by connecting the VME-EVO and VME-EVE modules. The VME-EVE is equipped with the VME-TTB. The RF clock of 508.89 MHz are put into the RF-IN channel of VME-EVO and the test system is operated with the Event clock of 127.2225 MHz. They are same condition as the real operation.

We measure the timing accuracy of TTL outputs from VME-EVE and VME-TTB with the oscilloscope. The same

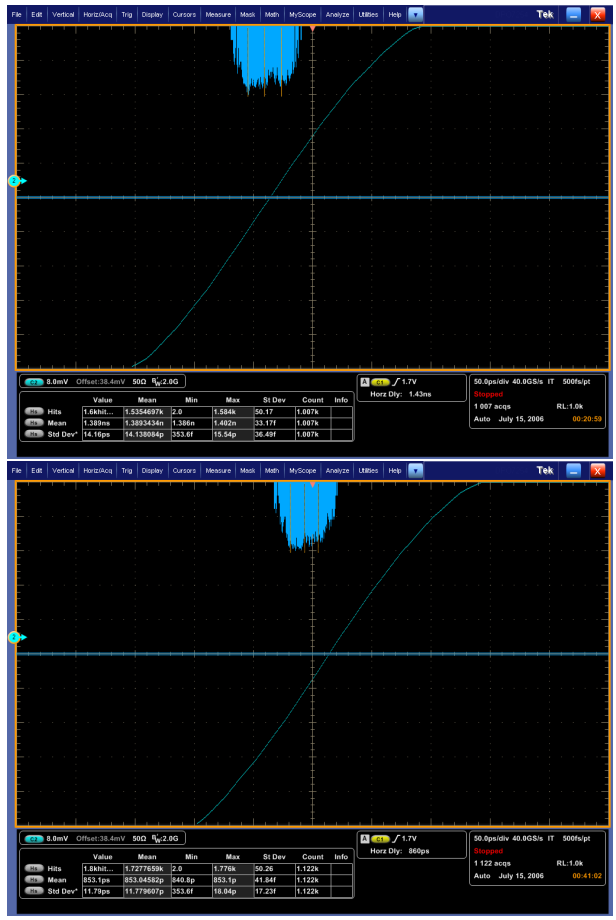


Figure 4: Results of timing accuracy measurements: results for CH1 of VME-EVE and CH0 of VME-TTB are shown in the upper and lower figures, respectively.

508.89 MHz clock is put into this oscilloscope as the reference of measurements. The oscilloscope is triggered by the TTL outputs and operated in the sampling mode. The trigger resolution and sampling step of oscilloscope are 100 fs and 500 fs, respectively. Therefore the capability in this timing measurement is enough to discuss following results.

The examples of results are shown in Figure 4. The upper picture is the result of CH1 of VME-EVE. The accuracy of output is determined from the standard deviation of histogram to be ~15 ps. The resultant accuracies of all other channels of VME-EVE are similar with this result. The result of CH0 of VME-TTB is shown in the lower picture. The accuracy of output is ~12 ps. This example picture also well represents the accuracy of all other channels of VME-TTB.

Note our results are a little bit worse than the same measurements performed at SINAP. One of the reasons is the quality of the reference signal. The RF clock which we use is a little bit noisy so that it affects the measurement.

CONCLUSION

We develop Sub-timing Station to manage two kinds of triggers for the DR devices. One is the triggers for indicating timing on the injection and extraction of beam-pulses.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The other is those for the beam monitors. The three Event modules are configured at Sub-timing Station.

The newly developed Event modules are adopted for the triggers to the beam monitors. They are implemented by inputing the DR revolution. Their Event clocks are 4 times division of the 508.89 MHz frequency which is for the RF cavities at DR.

The triggers for 84 BPMs at DR can be managed by the only 5 modules and it is one of the advantages of newly developed modules.

The timing accuracy of outputs from the VME-EVE and VME-TTB modules are determined to be ~15 ps and ~12 ps, respectively. They are quite better than the requirements for the BPM operation.

REFERENCES

[1] Y. Ohnishi, et al., "Accelerator design at SuperKEKB", Prog. Theor. Exp. Phys., 2013, 03A011.

[2] K. Abe, et al., "Letter of Intent for KEK Super B Factory", KEK Report 20014-4.

[3] K. Oide, et al., "KEKB B-Factory", Prog. Theor. Phys. 122, 2009, pp.69-80.

[4] S. Kurokawa, et al., "Overview of the KEKB Accelerators", Nucl. Instrum. Meth. A 499, 2003, pp.1-7.

[5] M. Akemoto, et al., "The KEKB Injector Linac", Prog. Theor. Exp. Phys., 2013, 03A002.

[6] PF website: `http://www.kek.jp/en/Facility/IMSS/PF/PFRing/`

[7] PF-AR website: `http://www.kek.jp/en/Facility/IMSS/PF/PFAR/`

[8] H. Kaji et al., "Upgrade of Event Timing System at SuperKEKB", Proceedings of ICALEPCS'13, San Francisco, CA, USA (2013).

[9] H. Kaji et al., "Construction and Commissioning of Event Timing System at SuperKEKB", Proceedings of IPAC'14, Dresden, Germany (2014).

[10] H. Kaji et al., "Installation and Commissioning of New Event Timing System for SuperKEKB", Proceedings of 12th Annual Meeting of PASJ, Fukui, Japan.

[11] MRF website: `http://www.mrf.fi/index.php/vme-products`

[12] E. Kikutani et al., "KEKB Bucket Selection System, Present and Future", Proceedings of 7th Annual Meeting of PASJ, Himeji, Japan.

[13] H. Kaji et al., "Bucket Selection System for SuperKEKB", Proceedings of 12th Annual Meeting of PASJ, Fukui, Japan.

[14] M. Liu et al., "Development Status of SINAP Timing System", Proceedings of IPAC'13, Shanghai, China (2013).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TIMING SYSTEM FOR THE HALPS/L3 ELI PROJECT

P. Camino, D. Monnier-Bourdin, Greenfield Technology, Bordeaux, France
M-A. Drouin, J.A. Naylon, B. Rus, FZU, Czech Republic
S. Telford, G. Johnson, C. Haefner, LLNL, Livermore, CA, USA

Abstract

The High Repetition-Rate Advanced Petawatt Laser System (HAPLS) forms part of the European Union's Extreme Light Infrastructure Beamlines project (ELI-Beamlines) which will be the first international laser research infrastructure of its kind. HAPLS will generate peak powers greater than one petawatt at a repetition rate of 10 Hz with 30fs wide pulses. ELI will enable unprecedented techniques for many diverse areas of research. HAPLS requires a high-precision timing system that operates either independently or synchronized with ELI's system. Greenfield Technology, a producer of mature picosecond timing systems for several years, has been hired by LLNL to provide just such a timing system. It consists of a central Master Timing Generator (MTG) that generates and transmits serial data streams over an optical network that synchronizes local multi-channel delay generators which generate trigger pulses to a resolution of 1ps. The MTG is phase-locked to an external 80 MHz reference that ensures a jitter of less than 10ps rms. The various qualities and functions of this timing system are presented including the LabVIEW interface and precision phase locking to the 80MHz reference.

INTRODUCTION

HALPS is planned to become PW workhouse of the ELI-beamlines facility composed of 3 other laser systems. ELI-beamlines located in Czech Republic is designed to explore fundamental physics under extreme conditions. It will contain the world's most powerful lasers for use by the international scientific community. ELI-beamlines is one of three laser facilities currently under construction as part of the ELI project (European Union's Extreme Light Infrastructure): ELI Attosecond located in Hungary to investigate natural phenomena on ultra-short timescales and ELI Nuclear Physics located in Romania dedicated to the new field of photonuclear physics are the two remaining facilities.

HALPS's short-pulse laser will use titanium-doped sapphire as its amplification medium. This laser is designed to convert the energy from the pump laser to 30-joule, 30-femtosecond pulses for a peak power exceeding 1 petawatt. HALPS is key to firing at 10Hz for hours at a time. The aim of such a laser is to generate secondary sources of electromagnetic radiation and accelerate charged particles. It will contribute to the development of laser-driven fusion power plants as well [1].

The HALPS block diagram is depicted on Figure 1. The main parts are: a pump laser + frequency converter and a short pulse laser + compressor.

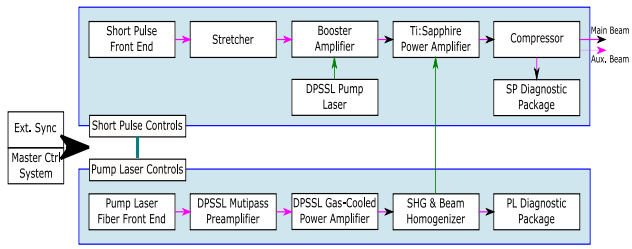


Figure 1 : HALPS block diagram.

We discuss the timing system required for such a laser.

PURPOSE

Synchronization of the pump laser beam and the short pulse laser beam requires generation of numerous precisely timed triggers. In addition the output of the laser needs to be synchronized in time with the balance of the ELI-Beamlines facility.

The purpose of the facility timing system is to generate and distribute over optical fibres a serially encoded, time critical data pattern to multiple zones. These zones contain multichannel delay generators that comprise the Local Timing System. The delay generators decode the data pattern contents and generate the electrical and optical triggers required by HALPS.

The low-jitter high precision timing system from Greenfield Technology has been selected by LLNL.

LOW JITTER – 1PS ELECTRICAL TIMING SYSTEM

Greenfield Technology has developed high precision timing systems for 10 years [2]. Its timing systems are already in use in several big physic instrument systems such as SOLEIL [3], LMJ, APOLLON [4], LULI (France) or Jupiter, NIF (LLNL, USA).

The one presented here has been customized to fulfill LLNL/HALPS requirements (output standard, clock frequency and content of the serial data stream). One of the major system specifications is the MTG phase locked specification to the 80MHz external clock with an rms jitter from clock to optical data stream less than 10ps. The different parts of the designed timing system are described in the following sections.

Master Timing Generator

The primary function of the MTG is the generation and distribution of timing data. It is composed of the Master Timing Transmitter and a Control Interface.

The Control Interface function serves the dual purpose of providing internal computation and control within the MTG and as interface to an external computer: Ethernet protocol communication using TCP/IP.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

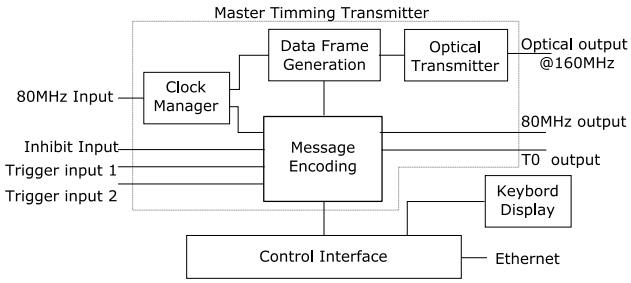


Figure 2 : MTG block diagram.

The Master Timing Transmitter is composed of 4 blocks (see Figure 2):

- Clock Manager, ensuring the phase lock to the external 80MHz reference or operation in the absence of this reference,
- Message Encoding, in charge of the assembly of a data frame containing synchronization data based on both internally generated and externally provided data,
- Data Frame Generation, in charge of serialization and 1B/2B encoding of the data frame contents @160MS/s,
- Optical Transmitter, converting the serial electrical data signal into a serial optical data, 6dBm – 1550nm.

The 20 kHz rate data frame is composed of a synch pattern, dynamic data and a 2 byte CRC. The dynamic part carries periodic events – epochs, single shot events and inhibit control bit – keys.

6 epochs are encoded in the data stream (see Table 1). Epochs occur for a single frame at both fixed rates and user specified periodic rates, each epoch phased to the shortest one and coincident to the 80MHz reference clock.

Table 1 : Epochs Rates

Name	Allowable Frequencies (Hz)
FE1	1000
FE2	100
FE3	10
F1	1000, 500, 200, 100, 50, 20, 10, 5, 3.333, 2
F2	200, 100, 50, 20, 10, 5, 3.333, 2, 1, 0.5
F3	50, 20, 10, 5, 3.333, 2, 1, 0.5, 0.2, 0.1

The 4 single shot keys encoded in the data stream can be initiated in one of three ways: front panel, external trigger inputs and software command. These 4 keys are two pairs of independent synchronous keys, each pair consisting of a primary key and a secondary key synchronous with the F3 epoch. A burst function is available enabling the generation of N sequenced primary and secondary keys (see Figure 3 – SS1/SS2 generation, N=2).

The inhibit key reflects the state of the inhibit control that can be controlled by either the external input or a software command. When active, it inhibits the data frame generation.

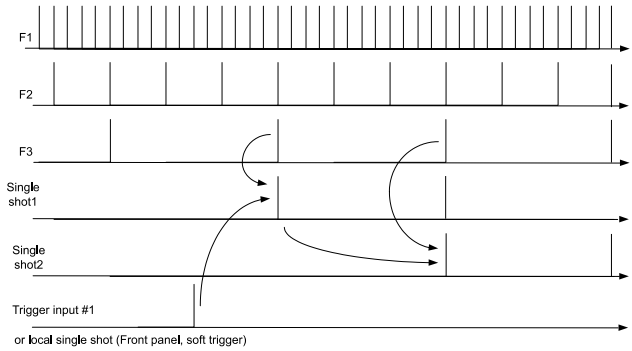


Figure 3 : Key sequence (burst mode N=2).

Multichannel Delay Generators (8 Channels)

The primary function of the Delay Generators is the generation of precision trigger pulses based on user settable delays and data control keys received from the MTG via an optically transmitted serial data stream.

The control keys include epochs and single shot events that control timing of the laser system. The 8 channel Delay Generators are using an Ethernet protocol communication via a TCP/IP interface.

Channel delays are achieved by combining digital delay – coarse delay – and analog delay – fine delay. Each channel has independent delay setting. The resolution of the digital delay part is determined by the recovered clock from the optical data stream, that is to say 6.25ns. The 1ps resolution of the analog delay part is achieved by using electrical verniers. By combining both parts, the system can deliver delays in a range 0 to 10s with a 1ps resolution (rms jitter $<$ 15ps) and accuracy$< 250ps + delay * t_c$, with t_c related to the reference clock (e.g. Rubidium clock : $t_c \sim 10^{-12}$).

Each output delivers square pulses that can be independently set in amplitude, delay, trigger and width.

Time Interval Meter (TIM)

Its aim is to measure the elapsed time between two trigger events with 1ps resolution. This measure is reported back to the timing control computer. The timing control computer will then adjust the delay settings of the system Delay Generators such that the time difference between the two measured triggers is less than 10ps.

The purpose is to enable two independent high precision timing systems to operate as a single time coordinated timing system. The TIM is using an Ethernet protocol communication via a TCP/IP interface.

The Greenfield Technology time interval meter is 1ps resolution device that can be optically linked to the MTG (to ensure clock coherence). It is composed of 4 inputs: one start, one stop, one gate and one optical input. One interesting feature of the device is its arming mode selection: a measure can be initiated by one of the 4 inputs.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

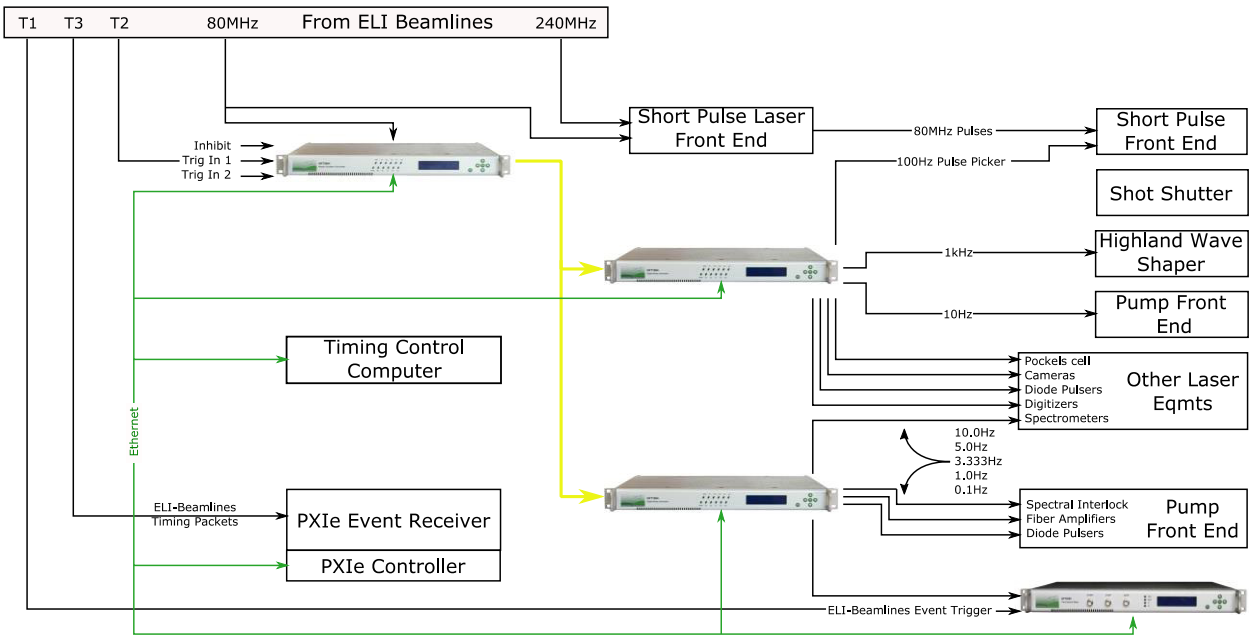


Figure 4 : HALPS timing system.

PRECISION CLOCK – RATE STABILITY

The main goal of the MTG design was to achieve a jitter from reference clock to optical data stream less than 10ps rms with a phase lock relationship. The ELI Beamline system is using a 80MHz Ti:sapphire master oscillator distributed over the 4 laser rooms. This clock ensures a high stability reference to the overall system, including the timing system.

The MTG converts that 80MHz reference into a 160MHz clock used to generate 1B/2B encoded message. A "0-delay" PLL is in charge of the conversion and of the fixed phase relationship. It ensures a deterministic phase, reproducible on every device reset. Its architecture is based on a two-stage cascaded PLL: the first stage provides a low-noise jitter cleaner function while the second stage performs the clock generation locked on the internal high-performance VCO. It enables a sub-200fs rms jitter even coupled with a low cost external crystal.

The reference clock to optical data stream jitter that is achieved by the MTG is presented on Figure 5: 7.8ps rms.

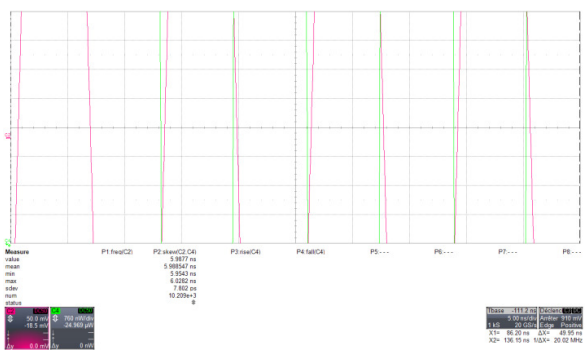


Figure 5 : Reference clock to optical data stream jitter.

L3 CONTROL SYSTEM

The overall L3 timing system architecture is presented on Figure 4.

All the Greenfield Technology devices are controlled via an Ethernet link by using simple syntax commands compliant with LXI standard. An embedded control application allows any web browser to check and modify the device parameters as well. Each device is composed of a front panel allowing users to instantly check/modify device parameters locally.

Greenfield Technology provides with every devices low level Labview VIs to build its own control interface.

People from ELI have developed a complete Labview interface based on these low level VIs to control each device.

CONCLUSION

HALPS timing system is currently under test at the LLNL facility. This system will provide functionalities needed for the L3 part of ELI-beamlines with the benefit of the Greenfield Technology device flexibility.

Such a system can be extended to manage up to 256 trigger events and control more experimentation rooms.

REFERENCES

[1] A.H., Lighting a New Era of Scientific Discovery, HALPS, S&TR Jan/Feb 2014.
[2] D.MB., Picosecond Timing System, Proceeding of ICALEPCS2013, 2013.
[3] JP.R., Synchronization System of Synchrotron Soleil, Proceeding of ICALEPCS07, 2007.
[4] M.P., Cilex-Apollon Synchronization and Security System, Proceedings of ICALEPCS2013, 2013.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

ERL TIME MANAGEMENT SYSTEM*

P. Kankiya†, T. Miller, B. Sheehy, Brookhaven National Laboratory, Upton, NY 11973, USA

Abstract

The Energy Recovery LINAC (ERL) at BNL is an R&D project. A timing system was developed in conjunction with other available timing systems in order to operate and synchronize instruments at the ERL. This paper describes the time management software which is responsible for automating the delay configuration based on beam power and instrument limitations, for maintaining beam operational parameters, and respond to machine protection system.

INTRODUCTION

The Energy Recovery LINAC at BNL is an R&D project and has been constructed in a blockhouse at BNL's large complex [1]. It is advantageous to setup a prototype machine on the same site which has long running accelerators such as RHIC because of access to resources and expertise. The challenging part is to meet project specific needs that are unable to utilise the existing infrastructure. In case of ERL's timing system it was partly possible to use RHIC style delay generator boards but certain features required programming new or buying commercial off-the-shelf (COTS) delay generators. The timing clock and events are distributed to all parts of ERL such that Laser and RF are always phase locked. This is the basis for synchronizing all aspects of ERL behaviour. The timing system is clocked by a 4.69 MHz clock optically derived from the photocathode laser; the laser is itself phase-locked to the 704 MHz SRF gun that serves as the master clock of the system. The laser clock is distributed to the timing system, installed about 200 ft away from Laser room in a VME rack. The laser-clocked VME system triggers some hardware systems directly, and others are triggered by low jitter (10s of psec) COTS delay generators (Stanford Research Systems DG645), which are triggered by the VME system. The DG645's offer more flexibility in certain functions (eg, burst, inhibit, gate shape, pre-scaling) than the native VME system, and this hybrid approach expands the capability and adaptability of the timing system. This assembly of various delay units and monitoring software together constitute the timing system of ERL and is credited for tasks such as coherent operation of measuring instruments, automatic delay calculation, supporting system testing, integrating several key hardware components, beam parameter limitation, inhibiting diagnostic systems and Laser beam production.

SYSTEM LAYOUT

The timing system currently comprises of two V202 delay modules installed in two different VME chassis and four DG645 boxes. A top level flow-diagram of timing signals is illustrated in Fig. 1. The inception of timing signals is at a V202 delay module. These modules are capable of decoding RHIC event link events [2]. A C++ class called Accelerator Device Object (ADO) [3] is created for controlling V202 boards. This ADO lets a user have control of each delay channel by providing fields for delay values, output pulse width, source of trigger etc. Using the ADO interface the board is configured to operate in external clock mode and is connected to 4.69 MHz clock generated by the photocathode laser. All output channels of 202 modules can be triggered externally, manually or at an event. The software representation of a DG645 is also realised by ADO. These units are all configured to be externally triggered and concurrently triggered by V202 channels with a maximum lag of 213ns lag (one clock cycle).

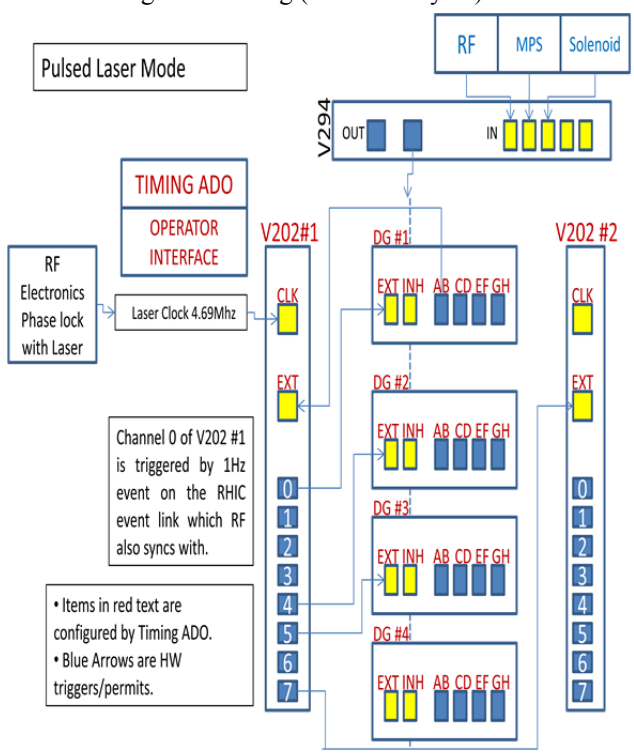


Figure 1: Block Diagram of the timing system with software and hardware components.

All configurable controls and statuses on these devices are displayed on user screens as ADO parameters. An ADO class referred to as master timing ADO is constructed to orchestrate timing signals across all ERL equipment via both styles of delay boxes such that the operator interface stays streamlined and unaffected by variety of hardware at lower level.

*Work supported by Brookhaven Science Associates, LLC under Contract No. DE-AC02-98CH10886 with the U.S. DOE.

† pkankiya@bnl.gov

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Bunch Structure

As per design ERL is capable of operating in continuous wave (CW) and pulsed mode. CW means that both laser and gun run at the full laser rep rate of f_L= 9.38 MHz. In pulsed mode, the delivery of laser pulses to the gun photocathode and triggers to other diagnostics is structured. The laser operates at 9.38 MHz, and this is the maximum frequency at which electron bunches may be produced; a fast optoelectronic switch is used, with a variable width gate, to select a number of bunches into a macrobunch or "car". The number of pulses may be arbitrarily large, and these macrobunches may be produced at rates of up to 10 kHz. Multiple macrobunches combine to form a "train". This relationship is illustrated in Fig. 2.

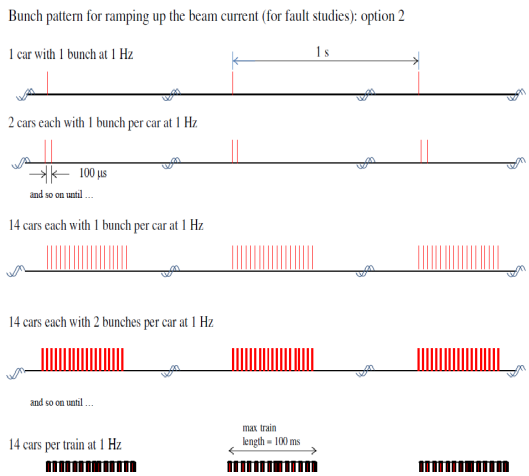


Figure 2: ERL bunch structure at different duty cycles with fixed macrobunch of 9.38 Mhz.

Generation of Macrobunches

In case of pulsed beam mode keeping the afore mentioned structure in mind a combination of in house developed delay generator boards(V202) and COTS Stanford Research System's DG645 are employed. V202 board as shown in Fig. 1 is responsible for generating the start trigger of the system. This is referred to as the start event and is repeated at 1Hz frequency. Thereafter the first of the four DG645s is triggered by this event, leading to the production of a burst of pulses. The burst parameters, such as frequency, delay, and period, are configured by the Timing ADO. Each burst produced by the DG645 device is a macrobunch train of the ERL beam.

Clock and Synchronisation with RF

The frequency of the SRF gun is chosen to be 703.75 MHz, or the 75th harmonic of the 9.383 MHz bunch spacing frequency of RHIC [3]. A PLL matched at the 75th harmonic of the laser pulse train and feeding back to the laser oscillator cavity length phase-locks the laser to the 703.75 MHz master clock. Synchronicity between RF pulsing in the cavities and the timing system is achieved by initiating both from the same event on the RHIC event link.

Jitter and Insertion Delays

Due to the presence of multiple types of delay generators and that the timing system hardware is split up between two chassis there are multiple contributors to the jitter of the overall system. The jitter values of a V202 externally trigger channel is on the order of 400ps. DG645 has a low jitter value of about 12ps and an insertion delay of 80ns. The timing ADO compensates for insertion delay introduced in the sequence by calculating the total insertion delay and adding that to the delay value configured for the fastest device; the Laser.

Start of Laser Train

The downstream DG645s are used to trigger various instruments and laser itself. Each unit has 4 output gates, AB, CD, EF, and GH, determined by the eight edges A-H. Each edge is software-configured relative to any other edge or T0. Output AB of DG645 #2 is chosen as the trigger for the electro-optical switch inside of the laser. Since the macrobunch might contain only one pulse, we want all measuring devices to be triggered in time to measure the first pulse. For each device "dev", the trigger comes before the optical switch opens, and the optical switch opens at T0 +δ_{max} , where δ_{max}= max(δ_{cam}, δ_{ICTLi} δ_{bFC} , …), so:

$$XXdev = T0 + \delta max - \delta dev$$

XXdev is the delay value to set on corresponding delay generator channels. This equation ensures the laser gate channel is one with longest delay value.

Maintaining Beam Power Limitation

Another key aspect of this ADO is to ensure the beam operation bounds are not violated. This is a preventive step to assure that the system never reaches the radiation safety limits of the designed accelerator. Any changes in the operating beam current, QE of the cathode [4] or maximum beam power are entered via the user interface described in a following section. The duty factor of the ERL operation cycle is limited by this equation:

$$f * w <= I0 /(Plmax * QE)$$

Automation Tasks Performed in Timing ADO

To simplify the task of an operator and reduce human error, the ADO automates the calculation of the length of the train and makes use of this value to set the gate width of selected beam diagnostics that need to remain open for entire time of beam cycle. The calculation of delay value is based on the special needs of electronics such as trigger width, and position. These settings are entirely automated and require no human intervention once the burst count is set.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

OPERATIONS

The ERL includes beam diagnostics required to characterize and tune beam parameters, as well as for machine protection [4]. Experimental data acquisition systems require a precision timing reference and triggers to properly coordinate their measurements in coincidence with the beam. It is of critical importance that these devices are triggered at a precise time with respect to the beam pulse for correct measurement of beam properties and losses. After deliberate discussions and testing with expert instrumentation engineers the pre-trigger values for these electronics have been determined.

The timing of most of the trigger signals have been measured using a Lecroy WR610Zi Oscilloscope with mixed signal input module (MSO), as shown in Fig. 3. The MSO provides 18 digital channels for various trigger and gate signals to be plotted on the scope display. The operators and engineers find this setup invaluable to the timing system of an R&D accelerator. Measurement signals are digitised by feeding them to high and slow speed digitiser VME boards namely 3123 and 3122 which themselves are triggered by the timing system to start measurements. The Analog signals on the scope in Fig. 3 show live beam charge measured by the faraday cup.

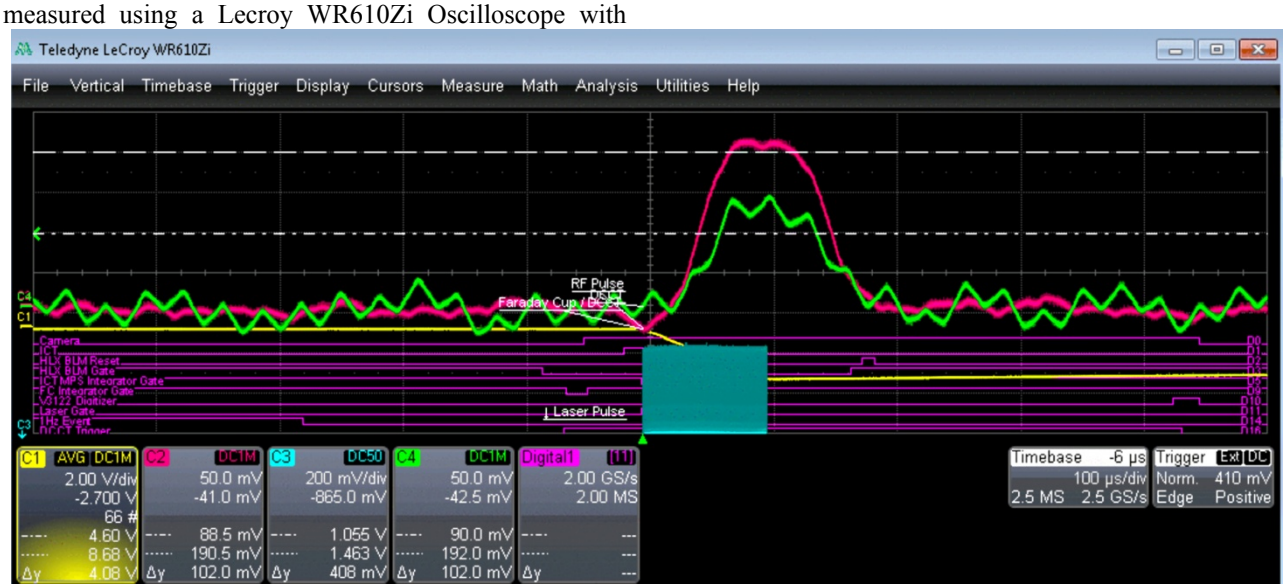


Figure 3: ERL accelerator instrumentation with the laser gate on a digital multiplexed oscilloscope.

USER INTERFACE

The User Interface of timing control system has been split into three categories. Each screen utilises the parameter editing tool (PET) tool for displaying critical operational parameters. PET is a customary user interface application developed at CA-D for controlling and measuring data represented as ADOs. First screen is a consolidation of all clients of timing system denoted by a channel on a delay generator and there delay configurations. After the initial testing phase, most of the values on this screen are governed by the master ADO interface rather than the operators. The second screen displayed in Fig. 4, is used to control beam parameters such as power of operation, QE at the start of a day's run, and beam current. Train parameters can be changed at the discretion of the shift leader from this screen. Another screen keeps track of default configuration of the DG645 boxes and lets an engineer control them via the master Timing ADO.

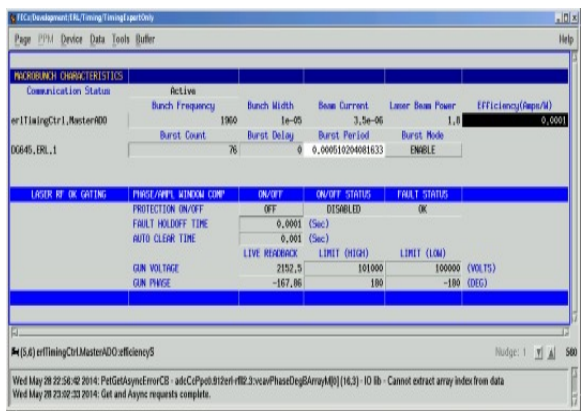


Figure 4: Timing expert only user interface screen.

TERMINATION OF TRAIN

It is possible to terminate the timing signal generation under three conditions:

1. At-will, from the operating console
2. When a finite programmed macrobunch sequence has completed

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

3. When RF system has any faults.

In case (1), the burst from DG645 #1 has to be terminated. This can be accomplished by sending the command 'BURM 0' to the delay generator. This disables burst mode. According to SRS engineers, this will terminate the burst within approximately 50 msec. To start a new acquisition, 'BURM 1' must be sent to re-enable burst mode, so this command should be part of the command sequence preceding each acquisition. In case (2), acquisitions will terminate naturally, as the burst sequence ends, and no further triggers are received by the laser optical switch or the instrumentation. At present there is no provision to communicate the end of the acquisition back to the control system. If this becomes necessary or desirable, it can be achieved by polling the status byte of DG645 #1. In case (3), laser emission will be terminated promptly through the machine protection system (MPS), which gates the signal to the optical switch. Electron beam will also cease, so it is not critical that data acquisition stop immediately. To avoid needlessly filling digitiser buffers, it would be desirable to stop acquisitions more quickly. This is accomplished by logically ANDing three signals: (a) a duplicate MPS signal inverted, and applied to the INHIBIT inputs of both DG645 2, 3 and 4 (applying to #1 will not work, as the INHIBIT inhibits new triggers, and will not terminate a burst that is in progress); (b) a health signal from RF system indicating TTL low, the good state (c) the High Temperature Super Conducting Solenoid's quench status.

COTS DG645

COTS devices were an attractive option for ERL timing controls in order to reduce costs by leveraging off of industrial solutions that could meet system requirements. An ADO class controls the assembly of DG645s at ERL communicating via TCP/IP. There are several key features that led to these devices being selected. The burst mode feature that allows the production of a high frequency train of triggers was a system requirement supported by the DG645. These units were bought with intent to make use of pre-scale trigger option which permits skipping external/internal trigger and prevents a channel from producing an output. This aspect was not utilised in the case of the ERL because the device does not keep track of the start of the train, and a random trigger input that will be skipped instead of a synchronised multiple of programmed pre-scale multiplier. One of the limitations of the device was observed when an output channel was set to large pulse width; this led to generating the rate error light indicating that several triggers were missed. This default grouping of all output channels confines the system to have a maximum delay value to keep operating at high beam frequency. Another drawback of this device is absence of ability to disable an output channel individually. In absence of beam it is desirable to turn off certain instruments whereas some of them must be triggered to avoid saturation of electronics. To overcome this inadequacy, all diagnostics were regrouped so that instruments that have similar trigger needs are connected to same DG645 box.

CONCLUSION

A timing system prototype has been designed and well tested with live beam of 2 KHz rep rate at BNL's Energy Recovery LINAC. A hybrid system composed of a variety of delay generating units with adjustable delays has been implemented. Key milestones for the ERL project have been met, and operations are in the Gun to Dump phase [5]. For future projects, COTS hardware such as MRF timing systems can be considered as an alternative to individual Delay Generator units in case of standalone accelerator projects. Efforts to make the user interface more operator-friendly to non-experts will continue.

REFERENCES

[1] L.T. Hoff, J.F. Skelly, "Accelerator devices as persistent software objects", Proc. ICALEPCS 93, Berlin, Germany, 1993.

[2] B. Oerter, C.R. Conkling, "Accelerator Timing at Brookhaven National Laboratory", Proc. 1995 Particle Accel. Conf., 1995.

[3] I. Ben-Zvi, et al., "The Status of the BNL R&D ERL," ICFA Beam Dynamics Newletter, No. 58, August 2012, p. 151.

[4] Wencan Xu, et al, "SRF Guns at BNL: first beam and other commissioning results.", MOIOB03, Proceedings of SRF2013, Paris, France.

[5] Toby Miller, et al, "Current Measurement and associated machine protection in the ERL at BNL",WEIALH2048, Proceedings of ERL 2015, Stony Brook, USA.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MASSIVE: AN HPC COLLABORATION TO UNDERPIN SYNCHROTRON SCIENCE

Wojtek James Goscinski, Chris Hines, Paul McIntosh,
Monash University, Clayton, Australia
Keith Bambery, Ulrich Felzmann, Chris Hall, Anton Maksimenko,
Santosh Panjikar, David Paterson, Mark Tobin,
Australian Synchrotron, Clayton, Australia
Chris Ryan, Darren Thompson,
CSIRO, Clayton, Australia

Abstract

MASSIVE is the Australian specialised High Performance Computing facility for imaging and visualisation. The project is a collaboration between Monash University (lead), Australian Synchrotron (AS) and CSIRO, and it underpins a range of advanced instruments, including AS beamlines. This paper will report on the outcomes of the MASSIVE project since 2012, in particular focusing on instrument integration, and interactive access for analysis of synchrotron data. MASSIVE has developed a unique capability that supports an increasing number of researchers, including an instrument integration program to help facilities move data to an HPC environment and provide in-experiment data processing. This capability is best demonstrated at the AS Imaging and Medical Beamline (IMBL) where fast CT reconstruction and visualisation is essential to performing effective experiments. A workflow has been developed to integrate beamline allocations with HPC allocation providing visitors with access to a dedicated project space and a CT reconstruction service using a remote desktop environment. The work herein describes the processing and analysis workflows developed for processing AS data at the IMBL, Macromolecular crystallography beamlines (MX), X-ray Fluorescence Microscopy (XFM), and the Infrared Microspectroscopy Beamline (IR).

THE MASSIVE FACILITY

MASSIVE was established to provide data analysis and visualisation services to the imaging and instrumentation community. The facility provides computer hardware, software and expertise to drive research in the biomedical science, materials research, engineering, and neuroscience communities, and it underpins advanced imaging research in synchrotron X-ray and infrared imaging, functional and structural magnetic resonance imaging, X-ray computed tomography (CT), electron and optical microscopy. MASSIVE offers Australian scientists access to two specialized computing facilities at Monash University and AS, plus specialised services offered through the cloud.

MASSIVE underpins the processing and analysis requirements of beamlines at AS. The IMBL is able to produce data at over 500 Mbytes/s and data sets over a terabyte in size. This introduces obvious challenges for researchers to capture, process, analyze and visualise data in a timely and effective manner. Researchers are also increasingly eager to perform data analysis "in-experiment" to make quick decisions.

Hardware

MASSIVE provides two computers, M1 and M2, that operate at over 5 and 30 teraflops respectively, using CPU processing, and over 50 and 120 teraflops, respectively, using Nvidia and Intel co-processors.. The computers are connected using a dedicated network for fast file transfer and management*.

GPUs have proved an important part of the MASSIVE environment. A number of key applications, including the X-TRACT [1] CT reconstruction software, have been parallelized to take advantage of GPUs and optimized on MASSIVE which is critical to fast processing of data in near real-time at the IMBL. Furthermore, GPUs underpin interactive visualisation through the MASSIVE Desktop and through parallel rendering tools such as Paraview [2].

Both systems have a GPFS [3] file system that is capable of a combined 5GB+ per second write speed. Fast file system performance has proved essential to support both the fast capture of data from instruments, and file system intensive image processing.

Instrument Integration

MASSIVE runs a program for the integration of imaging instruments with high performance computing capability. This program helps develop workflows for scientists to process and visualise data as an experiment progresses or immediately after it completes. A number of instruments have been integrated with MASSIVE [4]. The IMBL CT reconstruction workflow (described within) is a major output of this program.

The MASSIVE Desktop

MASSIVE provides users with an easily accessible scientific desktop environment configured for analysis and visualisation of research data. It provides researchers with access to a range of existing tools and software, including commercial and open-source applications used for synchrotron data processing such as X-TRACT [1],

* Detailed specifications available at: https://www.massive.org.au/high-performance-computing/resources

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

GeoPIXE [5], Drishti [6], and Avizo[†]. Many tools are integrated with the HPC scheduling environment [4].

The desktop environment is an effective way for MASSIVE to provide access to tools, without rewriting or wrapping those tools, underpinned by a high performance file system and running on high-end CPUs and GPUs.

The remote desktop uses the CentOS operating system running the KDE or Gnome environments. For remote access, the desktop uses TurboVNC/VirtualGL[‡], an open source VNC client, which supports remote hardware rendering and clients on Windows, Mac and Linux.

To make access as easy as possible the MASSIVE team develops a tool called Strudel[§] (short for Scientific Desktop Launcher) that automates the steps to access a desktop session. Strudel launches an interactive visualisation job through a visualisation queue, and connects using TurboVNC over a secure SSH connection. It is open source, configurable and also being applied at other Australian HPC facilities.

MASSIVE AND APPLICATIONS TO SYNCHROTRON SCIENCE

A number of workflows have been deployed by AS to leverage the M1 system. Five beamlines use MASSIVE to process user data during or shortly after data capture: the IMBL, MX1 and MX2 beamlines, XFM, and IR.

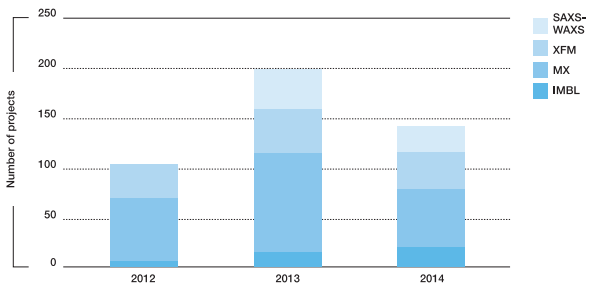


Figure 1: HPC usage by beamline 2012 to 2014 (MX1 and MX2 are shown combined).

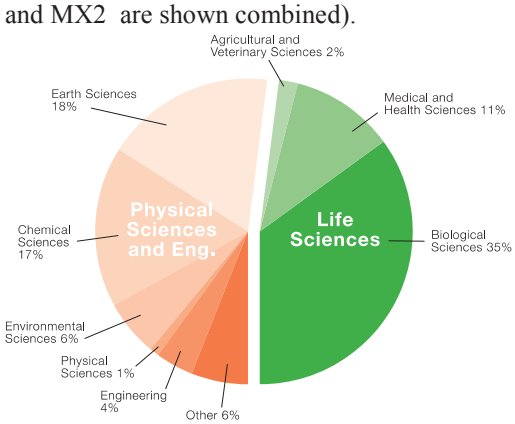


Figure 2: Research areas supported by the M1 and M2 computers during or shortly after beamtime.

During the period 2011 to 2014, approximately 446 beamtime sessions have used M1. Figure 1 shows the number of beamtime allocations that used MASSIVE. Figure 2 characterizes the sciences using the facility.

IMAGING AND MEDICAL BEAMLINE

In order to control the data collection and optimise the experimental conditions at the IMBL, scientists must be able to visualise collected data in near real-time as the experiment is in progress. In particular, groups that are imaging anesthetized animals need to establish whether a previous scan has successfully produced the desired data before proceeding with the next step of an experiment.

The most popular workflow on the IMBL is for CT, which requires collection of a set of X-ray projection images with the specimen being rotated between each. To start a data collection the user sets up the specimen on a rotary table then runs acquisition from a custom GUI on a control computer. The GUI allows specimen information to be recorded and checks the CT parameters and instruments. IMBL motion control hardware produces a train of pulses between a defined start and finish angle of the rotation stage. Each pulse edge occurs at the defined projection angle for the image. The imaging detector responds to pulses by capturing an image at a set exposure time. Image data is rapidly recorded to an image file on the file store. With the IMBL standard 5.5 megapixel X-ray imager the full frame rate is around 30 frames per second. Tagged Image File Format (TIFF) files are currently used but a HDF5 format is being developed.

The computer used to control and collect data from the detector is connected via a high-speed fibre network link to the file system on M1. Following a single CT collection a set of projection images will be saved for either 180 degree or 360 degree rotation of the sample. Since the synchrotron beam has a high aspect ratio, it is often the case that a single specimen will require several segments collected through its height.

The preferred reconstruction software for CT data sets collected on IMBL is X-TRACT from CSIRO [1]. It runs on the MASSIVE desktop and provides a simple and easy to use interface for reconstruction. Each user group which uses IMBL is provided with an individual MASSIVE project, along with both archived data storage space, and scratch space for intermediate calculations (details described in following section). To stitch projection images prior to reconstruction IMBL provides a script which uses ImageMagick[**] and CTAS [7].

The first imaging workflow step is to remove noise and pedestals using standard flat and dark field correction. Sinograms are compiled in memory, then reconstructed into image slices and then written to disk. 32 bit TIFF files are used since the popular output from X-TRACT is the calculated linear attenuation coefficient map. Visualisation of reconstructed data is done using the MASSIVE Desktop running Dristhi, ParaView, or Avizo.

[†] http://www.fei.com/software/avizo3d/
[‡] http://www.virtualgl.org/
[§] https://www.massive.org.au/userguide/cluster-instructions/strudel
[**] http://www.imagemagick.org/script/index.php

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Annually IMBL provides around 100 days of CT imaging experiment time . On average a group will collect 5 to 10 TB of raw data. The scratch space required when processing this can be double that.

HPC Integration at IMBL

In 2015 AS and MASSIVE initiated a project to tightly integrate IMBL beamline allocations and project accounts on MASSIVE. The intent is to provide IMBL users with automatic access to desktop sessions for data processing and collaboration during and after an experiment. The technology to support this project was developed under the NeCTAR Characterisation Virtual Laboratory[††].

The integration workflow is as follows (Figure 3): (1) A user creates an ID in the AS user database and submits a proposal for access using the AS portal; (2) If beamtime is approved a booking is created and collaborators can be added to the booking; (3) The MASSIVE ID system periodically queries the AS user database using a RESTful web service; (4) A new MASSIVE user matching each collaborator and a new project matching each experiment is created in the MASSIVE ID system. Passwords on MASSIVE are not set (as they will never be used) and user names are not published (as they will be determined automatically); (5) At the beginning of a beamline session, an operator resets the system. This archives data captured during the previous experiment and defines the project for new user data on MASSIVE; (6) User visits the beamline; (7) Data captured is written to the defined location on MASSIVE and file ownership set; (8) For processing, the user logs into MASSIVE using Strudel providing their AS user credentials; (9) Strudel authenticates against the AS ID system. On successful authentication a token is returned; and (10) Strudel uses the token to authenticate the user to a MASSIVE web service and to establish a secure desktop instance.

This workflow has a number of attributes: First, it provides AS the ability to automatically create (and discontinue) projects and user identities on a shared HPC system; Second, it provides AS users and their collaborators with a dedicated project space for each experiment; and third AS users are provided with direct access to the MASSIVE Desktop while visiting the IMBL and the same environment after their visit.

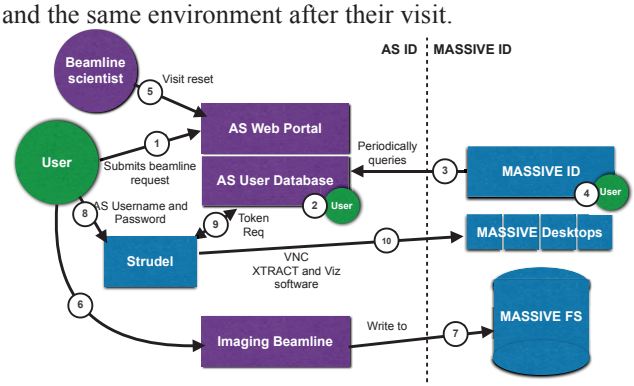


Figure 3: IMBL HPC project and allocation workflow.

This workflow is deployed at the IMBL and since April 2015 it has been used by 35 experiments. There are plans to deploy the same workflow to XFM.

MACROMOLECULAR CRYSTALLOGRAHY BEAMLINES

The AS hosts two MX beamlines: the bending-magnet high-throughput MX1 beamline [8] and the micro-focus MX2 beamline powered by a small-gap in-vacuum undulator. Both beamlines deliver tuneable X-rays in the energy range from 6 to 18 keV for the standard MX and anomalous diffraction experiments for experimental phasing.

The beamlines use Auto-Rickshaw (AR) [9, 10], a data evaluation system, which provides computer coded decision-makers for successive and automated execution of a number of existing macromolecular crystallographic computer programs thus forming a software pipeline for automated and efficient crystal structure determination. AR is triggered by the collection of a diffraction image.

The main AR server (http://www.embl-hamburg.de /Auto-Rickshaw) runs at EMBL-Hamburg and is freely accessible publically. A mirror server has been installed on MASSIVE for automatic data processing for users. The system is triggered from the beamline computers when the auto-processed X-ray data indicates an anomalous signal. The results from AR are automatically directed to the same directory from where the jobs were triggered. The automatic triggering from the beamline computer are available as an opt-in mode for users. Apart from the automatic system, AR can be triggered from the beamline computers for structure solutions using various phasing methods over a web-based user interface by selecting appropriate parameters and by selecting the processed and scaled datasets. This workflow on MASSIVE has helped to produce the structure of the BC component of ABC toxins that explains the general mechanism for protein encapsulation and delivery [11]. Additionally, it helped to determine the crystal structure of the Extracellular Adhesion Domain of the Sialic Acid-binding Adhesin SabA from *Helicobacter pylori* [12] the Mg^{2+} transporter CorA from the Archaea *Methanocaldo-coccus jannaschii* [13], an Indole-3-Acetic Acid Amido Synthetase from *Grapevine* [14] and cyanuric acid hydrolase [15].

X-RAY FLUORESCENCE MICROSCOPY

Remote login to MASSIVE via the portal provides users at the XFM beamline [16] a persistent desktop environment to run the GeoPIXE package from CSIRO [17] for the reconstruction of list-mode data-sets, acquired at XFM using the Maia detector and fluorescence imaging system [18], and the exploration and analysis of these rich data-sets. Maia is a 384 element detector array with dedicated FPGA-based real-time processor, which can acquire data at event rates of up to around 10 MHz, with each event tagged by XY position in a scan, dwell time and incident and transmitted beam flux count for each

[††] https://www.massive.org.au/cvl

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

pixel. First-pass processing is performed in real-time using the Dynamic Analysis method (DA) [5] implemented in the FPGA.

Off-line processing of the list-mode stream and deconvolution of its rich spectral content is performed in GeoPIXE exploiting 10-12 cores on M1. GeoPIXE also generates the DA transform matrix used for the real-time deconvolution. Using the matrix approach and parallel processing on MASSIVE, DA achieves the result at rates of around 10^8 events into 10^4-10^5 pixels per second. Further tools in GeoPIXE permit the reconstruction of XANES image stacks for speciation analysis and their correction, for example for spatial drift. Users then explore this multi-element or speciation image space using GeoPIXE and extract spectra from specific features for further quantitative analysis or to verify the accuracy of imaging. The resulting image data-sets, which can be as large as 1 G pixel showing detail in 20-30 elements, are explored interactively using GeoPIXE or using statistical analysis of the hyperspectral data.

This capability allows super high-definition X-ray fluorescence element images to be obtained with detail at ~2 μm spatial resolution in areas of several square cm. The high pixel count opens up a suite of 3D imaging techniques, which map over 3 dimensions to provide high-resolution X-ray absorption near-edge structure (XANES) images of chemical speciation (scan in XYE) and fluorescence tomography reconstructions of 3D multi-element distribution (scan in XYθ).

GeoPIXE on MASSIVE is used for processing 95% of experiments at XFM and usage after a visit is increasing.

INFRARED MICROSPECTROSCOPY BEAMLINE

The IR beamline manages and maintains a project on MASSIVE to aid in the analysis of infrared spectra recorded from biological samples. Fourier transform infrared (FTIR) microspectroscopy has been applied to diverse areas of biological and medical research. The technique provides a direct and rapid probe of chemical composition without the need for sample pre-treatment with stains and has the potential to identify and validate important biomarkers. However, light scattering effects can hamper the analysis of these FTIR spectra and this is especially problematic for the case of Mie scattering from intact cell samples. MASSIVE is being used to computationally model and correct for these light scattering effects in synchrotron FTIR microscopy data. The algorithm used is Resonant Mie Scattering modelling through Extended Multiplicative Signal Correction (RMieS-EMSC) [19]. Recent studies on cells performed at the beamline include work on developing a diagnosis for malaria [20] and understanding the biological effects of radiation [21]. Parallel computing allows RMieS correction to be performed around 100 times faster than is possible on desktop computers. MASSIVE allows data characterisation to be performed directly on the spectra stored on the synchrotron facilities archive, so results can be shared with collaborators conveniently and securely.

CONCLUSION

The MASSIVE collaboration has significantly enhanced data processing at AS beamlines. The analysis workflows and desktop described within are important AS services. This is most strongly demonstrated when MASSIVE is being used "in-experiment" allowing the visitor make quick decisions, increasing AS productivity. Flexibility in scheduling and HPC management is important and it is essential that capability can be reserved for instrument and interactive use. During these times, the system maybe underutilised however the impact is gained when scientists have access to HPC capability on demand.

ACKNOWLEDGMENT

This work is funded by project partners, the NCI Specialised Facilities, and Victorian Government and the NeCTAR CVL. The authors acknowledge contributions from staff at AS, Monash and CSIRO.

REFERENCES

[1] T. E. Gureyev, *et al.*, "Toolbox for advanced X-ray image processing," *Proc. SPIE8141 B,* vol. 81410, pp. 81410B-14, 2011.

[2] A. Henderson, et al, *The ParaView Guide*: Kitware, NY, 2004.

[3] F. B. Schmuck and R. L. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," in *FAST*, 2002, p. 19.

[4] W. J. Goscinski*, et al.*, "The Multi-modal Australian ScienceS Imaging and Visualisation Environment (MASSIVE) high performance computing infrastructure: applications in neuroscience and neuroinformatics research," *Frontiers in Neuroinformatics,* vol. 8, 2014-March-27 2014.

[5] C. Ryan, "Quantitative trace element imaging using PIXE and the nuclear microprobe," *International Journal of Imaging Systems and Technology,* vol. 11, pp. 219-230, 2000.

[6] A. Limaye, "Drishti: a volume exploration and presentation tool," in *SPIE Optical Engineering+ Applications*, 2012,

[7] A. Maksimenko, M. Ando, S. Hiroshi, and T. Yuasa, "Computed tomographic reconstruction based on x-ray refraction contrast," *Applied Physics Letters,* vol. 86, p. 124105, 2005.

[8] N. P. Cowieson*, et al.*, "MX1: a bending-magnet crystallography beamline serving both chemical and macromolecular crystallography communities at the Australian Synchrotron," *Journal of Synchrotron Radiation,* vol. 22, pp. 187-190, 2015.

[9] S. Panjikar, et al, "Auto-Rickshaw: an automated crystal structure determination platform as an efficient tool for the validation of an X-ray diffraction experiment," *Acta Crystallographica Section D: Biological Crystallography,* vol. 61, pp. 449-457, 2005.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

[10] S. Panjikar, et al, "On the combination of molecular replacement and single-wavelength anomalous diffraction phasing for automated structure determination," *Acta Crystallographica Section D: Biological Crystallography,* vol. 65, pp. 1089-1097, 2009.

[11] J. N. Busby, et al, "The BC component of ABC toxins is an RHS-repeat-containing protein encapsulation device," *Nature,* vol. 501, pp. 547-550, 2013.

[12] S. S. Pang, *et al.*, "The three-dimensional structure of the extracellular adhesion domain of the sialic acid-binding adhesin SabA from Helicobacter pylori," *Journal of Biological Chemistry,* vol. 289, pp. 6332-6340, 2014.

[13] A. Guskov, *et al.*, "Structural insights into the mechanisms of Mg2+ uptake, transport, and gating by CorA," *Proceedings of the National Academy of Sciences,* vol. 109, pp. 18459-18464, 2012.

[14] T. S. Peat, *et al.*, "Cyanuric acid hydrolase: evolutionary innovation by structural concatenation," *Molecular microbiology,* vol. 88, 2013.

[15] T. S. Peat, et al, "Crystal structure of an indole-3-acetic acid amido synthetase from grapevine involved in auxin homeostasis," *The Plant Cell,* vol. 24, pp. 4525-4538, 2012.

[16] D. Paterson, et al., "The X- ray Fluorescence Microscopy Beamline at the Australian Synchrotron," in THE 10TH INTERNATIONAL CONFERENCE ON X- RAY MICROSCOPY, 2011, pp. 219-222.

[17] C. Ryan, *et al.*, "MAIA X-ray fluorescence imaging: capturing detail in complex natural samples," in *Journal of Physics: Conference Series*, 2014, p. 012002.

[18] R. Kirkham, *et al.*, "The Maia Spectroscopy Detector System: Engineering for Integrated Pulse Capture, Low-Latency Scanning and Real-Time Processing," *SRI 2009, 10th International Conference on Radiation Instrumentation. AIP Conference Proceedings,* vol. 1234, pp. 240-243, 2010.

[19] P. Bassan, *et al.*, "Resonant Mie scattering (RMieS) correction of infrared spectra from highly scattering biological samples," *Analyst,* vol. 135, pp. 268-277, 2010.

[20] B. R. Wood, *et al.*, "Diagnosing malaria infected cells at the single cell level using focal plane array Fourier transform infrared imaging spectroscopy," *Analyst,* vol. 139, pp. 4769-4774, 2014.

[21] E. Lipiec, *et al.*, "SR-FTIR Coupled with Principal Component Analysis Shows Evidence for the Cellular Bystander Effect," *Radiation research,* vol. 184, pp. 73-82, 2015.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DATABROKER: AN INTERFACE FOR NSLS-II DATA MANAGEMENT SYSTEM

A. Arkilic, L. Dalesio, D. Allan, D. Chabot Brookhaven National Laboratory, NSLSII
Upton, NY, USA

Abstract

A typical experiment involves not only the raw data from a detector, but also requires additional data from the beamline. This information is largely kept separated and manipulated individually, to date. A much more effective approach is to integrate these different data sources, and make these easily accessible to data analysis clients. NSLS-II data flow system contains multiple back-ends with varying data types. Leveraging the features of these (metadatastore, filestore, channel archiver, and Olog), this library provides users with the ability to access experimental data. The service acts as a single interface for time series, data attribute, frame data access and other experiment related information.

INTRODUCTION

The current and projected NSLS-II beamlines will support a wide range of disciplines from biology to physics with a wide range of techniques. Higher beam quality and new generations of area detectors are increasing data rates quickly. Increasingly all disciplines are using multi-element detectors to measure and record the results of the experiment. Current state-of-the-art detectors are capable of producing megapixel images with frame-rates in the kilohertz range which results in data rates of up to 800 Mb.s-1. Such data-rates are fully compatible with current off-the-shelf RAID storage technology, effectively allowing the data to be written to a storage medium as it is produced. The challenge for NSLS-II, however is to be able to process and analyse the data at this rate, with these volumes to allow the experimenter to not only make the "first-cut" for decision making during the experiment, but to retrieve data and analyse it for publication. NSLS-II middle layer beamline applications (Fig. 1), powered by MongoDB, the General Parallel File System (GPFS), and EPICS provide such capabilities.

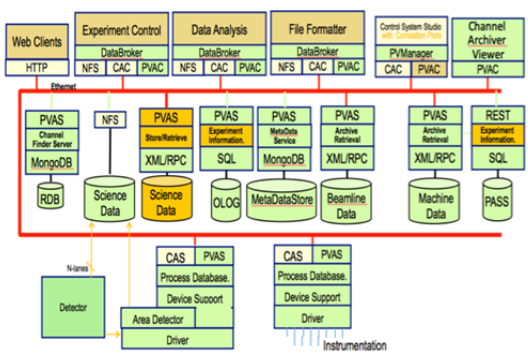


Figure 1: An overview of NSLS-II middle-layer applications.

Given the modular architecture and complexity of these various applications, a need for an easy point of access for all experimental data has occurred. In order to simplify the access to critical experimental data, we have developed databroker.

ARCHITECTURE

databroker is the pathway for data analysis tools to access data without any knowledge of experimental data. The library in production is implemented in Python and provides users with broker objects that contain Pandas data frames, allowing semi and un-structured experimental data to be accessed within scientific Python framework with ease via its narrow interface.

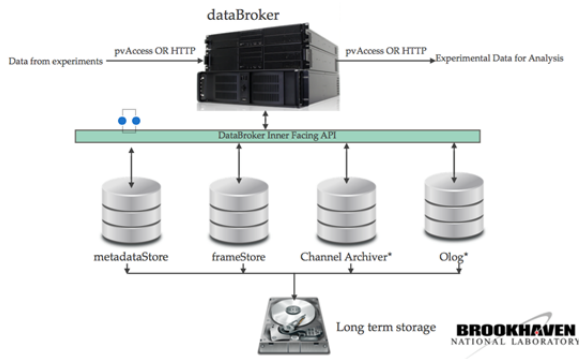


Figure 2: databroker Architecture Overview.

databroker uses the client APIs of metadatastore, filestore, channel archiver, and Olog (Fig. 2). Its primary goal is to provide users with run-based information. A run is defined as an end-to-end scan or sweep that is performed and captured either via NSLS-II run engine, BlueSky, or custom data acquisition framework that utilizes NSLS-II middle layer services. Run based data management is common to all beamlines and it provides users with the desired abstraction.

Deployment Strategy and Access Environment

All python clients that are part of NSLS-II middle layer get deployed to each beamline using the Anaconda python package management toolkit. The sandbox approach allows us to deploy packages that are required but not natively supported by the Debian 8. IPython notebook is also part of this deployment and it is the primary platform to access databroker. In addition to local access to databroker, NSLS-II beamlines provide access to an identical databroker session remotely using jupyter hub.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Searching by ID or Recency¶

Here is a summary of the "Do What I Mean" slicing supported by `DataBroker`.

syntax	meaning
`DataBroker[-1]`	most recent header
`DataBroker[-5]`	fifth most recent header
`DataBroker[-5:]`	all of the last five headers
`DataBroker[108]`	header with scan ID 108 (if ambiguous, most recent is found)
`DataBroker[[108, 109, 110]]`	headers with scan IDs 108, 109, 110
`DataBroker['acsf3rf']`	header with unique ID (uid) beginning with `acsf3rf`

Time-based Queries¶

Runs that took place sometime in a given time interval are also supported.

syntax	meaning
`DataBroker(start_time='2015-01')`	all headers from January 2015 or later
`DataBroker(start_time='2015-01-05', end_time='2015-01-010')`	between January 5 and 10

Complex Queries¶

Finally, for advanced queries, the full MongoDB query language is supported. Here are just a few examples:

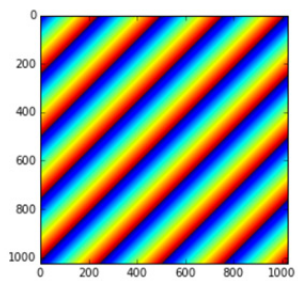
syntax	meaning
`DataBroker(sample={'$exists': True})`	headers that include a custom metadata field labeled 'color'
`DataBroker(scan_type={'$ne': 'DeltaScan'})`	headers where the type of scan was not a `DeltaScan`

Figure 3: Querying capabilities overview.

Using Databroker

As described in Fig. 3, databroker provides different querying methods such as, unique identifier based, time based, and complex/aggregated. This sort of querying approach provides users with immense data mining capabilities that were not available as a part of most legacy systems. Once a query is submitted to databroker yields to a metadatastore query. The results of this query allow databroker to compose a **header** that is composed of a RunStart, RunStop, and a set of EventDescriptor documents [1]. databroker lazily loads the Event documents, which hold the actual experiment metadata, using a generator object. This approach is performance savvy as in most cases users only want to access the header of the experimental data and request the data payload if needed. In addition to data in metadatastore, databroker is capable of locating images that are related to each data point within specific metadatastore events. As shown in the jupyter session in Fig. 4, querying and accessing complex experimental data requires a handful of comments, allowing users to get to analysis stage with very little effort.

```
In [1]: from dataportal.broker import DataBroker

In [2]: from dataportal.muxer import DataMuxer

In [3]: header = DataBroker[-1]

In [4]: events = DataBroker.fetch_events(header)

In [5]: dm = DataMuxer.from_events(events)

In [6]: dm.sources
Out[6]: {u'None_acquire_period': u'PV:ADSIM:cam1:AcquirePeriod_RBV',
         u'None_acquire_time': u'PV:ADSIM:cam1:AcquireTime_RBV',
         u'None_image_lightfield': u'PV:ADSIM:',
         u'None_stats_total1': u'PV:ADSIM:Stats1:Total_RBV',
         u'None_stats_total2': u'PV:ADSIM:Stats2:Total_RBV',
         u'None_stats_total3': u'PV:ADSIM:Stats3:Total_RBV',
         u'None_stats_total4': u'PV:ADSIM:Stats4:Total_RBV',
         u'None_stats_total5': u'PV:ADSIM:Stats5:Total_RBV',
         u'm1': u'PV:LSBR-DEV:m1.RBV',
         u'sclr_ch2': u'PV:AISIM:ai1'}

In [7]: images = dm[u'None_image_lightfield']

In [8]: import matplotlib.pyplot as plt

In [11]: %matplotlib inline

In [12]: plt.imshow(images.values[0])
Out[12]: <matplotlib.image.AxesImage at 0x7f1a24adf290>
```




```
In [ ]:
```

Figure 4: A sample jupyter Session and access to sample experiment via databroker interface.

databroker does not only support experimental data saved in filestore and metadatastore. NSLS-II beamlines archive all process variables within the scope of an experiment in channel archiver. By providing process variable based search, databroker provides access to this experimental data. This allows users to investigate factors they left out during data acquisition stage. Another important aspect of databroker is its capability to access Olog, the rich logging environment [2]. Users can create log entries in which they can denote interesting findings of their experiments. This is very useful as Olog provides users with CS-Studio and web browser views.

CONCLUSION

databroker provides access to complex, high-performance, and modular NSLS-II middle layer framework by providing a well-defined narrow interface that is accessible both locally and remotely. Its intuitive and simple API allows end-users to start analyzing

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

experimental data during their experiment,. This allows them to discover interesting attributes of their samples while they still have the chance to investigate them further.

REFERENCES

[1] metadatastore: A primary data store for NSLS-II beamlines, A. Arkilic, L. Dalesio, D. Allan, W. K. Lewis, Presented at the ICAPLEPCS 2015.

[2] Olog and the CSStudio: A rich logging environment, K. Shroff, L. Dalesio, A. Arkilic, E. Berryman Presented at the 2013 ICALEPCS.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MADOCA II DATA LOGGING SYSTEM USING NoSQL DATABASE FOR SPring-8

A. Yamashita*, M. Kago, SPring-8, Hyogo, Japan

Abstract

The data logging system for SPring-8 was upgraded to the new system using NoSQL database, as part of the MADOCA-II framework. The data logging system has been collecting the log data that is required for accelerator control without any issues since the upgrade. In the past, the MADOCA system powered by a relational database management system (RDBMS) had been operating since 1997. The MADOCA system had grown with the development of accelerators. However, the system with RDBMScould not handle new requirements like variable length data storage, data mining from large volume data and fast data acquisition. Therefore, new software technologies were developed to address these problems. In our proposed system, we adopted two NoSQL databases, Apache Cassandra and Redis, for data storage. Apache Cassandra is utilized for providing a perpetual archive. It is a scalable and highly available column oriented database suitable for time series data. Redis is used for the real time data cache because of a very fast in-memory key-value store. Data acquisition part of the new system was also built based on ZeroMQ message packed by MessagePack. The operation of the new system started in January 2015 after an evaluation term over one year.

INTRODUCTION

The new MADOCA-II data acquisition and storage system took over from the old system in January 2015. Since then, it has been under operation at the SPring-8 accelerator and beam lines control system without major troubles.

In the old MADOCA system [1], one large RDBMS [2] managed not only the equipment and operation parameters but also the log data [3]. It began the production run during the commissioning of SPring-8 in 1997 and has since grown as the accelerator has evolved. The number of signal data increased from 871 at the commissioning in 1997 to 27,626 at the end of 2014 and the size of the database expanded from 17.2GB (1998 one year) to 4 TB. With the progress of the accelerator and for the next generation SPring-8-II [4], we required more scalability, flexibility and maintainability for the database system especially for the log database. In The old MADOCA database system, almost every control application depended on a single database server and the server was required to be reliable and without down time. On the other hand, increase in the number of signals required more performance from one server. Therefore, we employed a fault tolerant database server and SAN (Storage Area Network) storages for reliability. Although, they are very reliable and have worked without trouble in years, they are expensive and pose a difficulty during scale up.

In the previous ICALEPCS, we reported the design and implementation of MADOCA-II database, which used NoSQL databases [5]. The system consisted of two databases, Apache Cassandra [6] for perpetual data store and Redis [7] for the real time data cache. The Apache Cassandra runs on a redundant cluster of commodity servers. It makes the cluster scale-out by simply adding additional server nodes. Data are replicated and distributed to nodes. In our case, up to two simultaneous node downs are tolerable. On the other hand, Cassandra scarifies the consistency of data. A Cassandra cluster distributes data to its nodes. It takes time (about 1 sec in our case) until every node has consistent data. We overcome the inconsistency of Cassandra by writing data into a very fast in-memory data cache; Redis. The data acquisition system writes data into Cassandra and Redis simultaneously. Two Redis servers are run in parallel for redundancy.

We installed the test system in SPring-8 accelerator and beam line control environment. The system acquires and stores the same set of data as MADOCA database system in parallel. During the one year test, we operated the database and data acquisition system with no major trouble. We obtained many know-hows during the test run. From January of 2015, we replaced old data acquisition and database system with new MADOCA-II system with some modifications based on know-hows that were obtained during the test run.

IMPLEMENTATION

Figure 1 shows the entire data acquisition and database system. Our previous paper [8] explained the data acquisition system. We did not make major modifications to it. The data acquisition system operates with the old system in parallel.

We developed a data sender running embedded computer as well as libraries for data reading, web system and alarm system running on client computers.

Data Senders

We developed two types data sender one is called po2m2db which runs on an embedded computer system. It extracts data from the shared memory inside, produces messages, and transmits them to a relay server asynchronously. The other data sender is called cc2m2db for an embedded system that has not enough resources to run po2m2db. The old data acquisition system [9], which runs on a workstations, outputs data to stdout. cc2m2db captures the stdout data, parse them, generates messages and send to the relay server.

* aki@spring8.or.jp

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

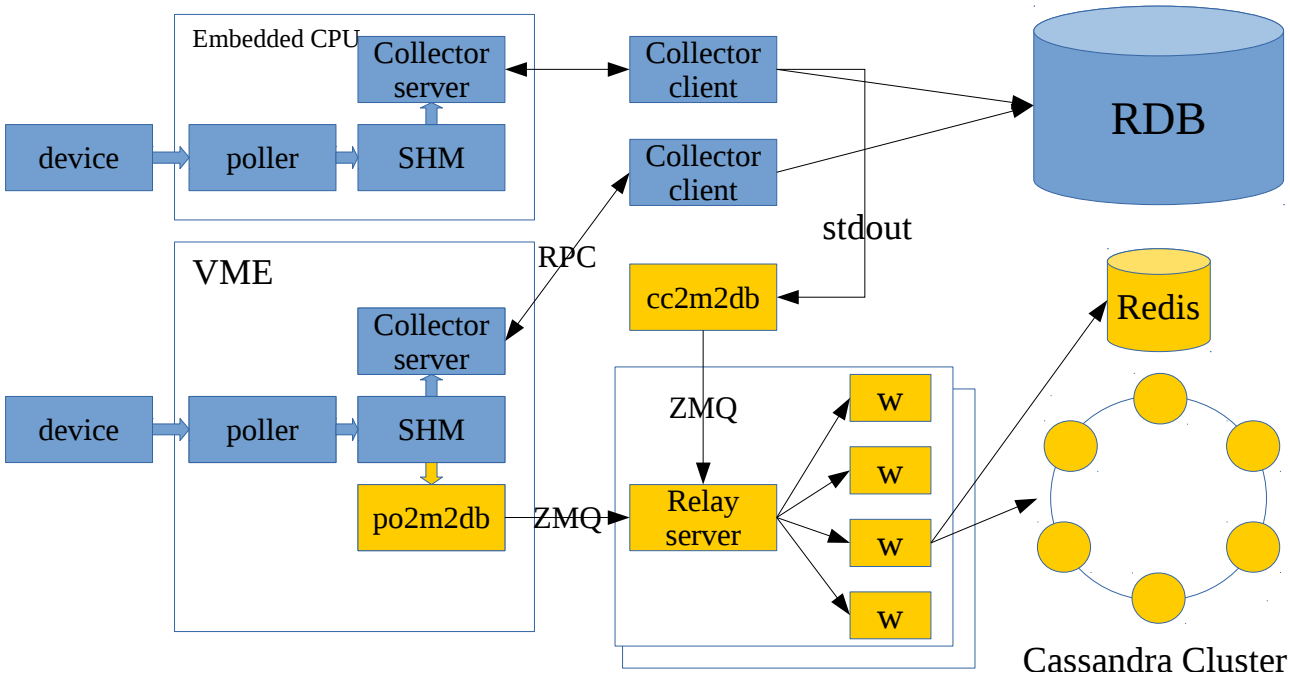


Figure 1: Entire data acquisition system. Blue ones are components of old MADOCA system. Yellow ones are newly developed MADOCA-II system. Both are running in parallel. Two relay servers are running for load balancing and redundancy. "w" means a writer thread.

Client Library

The client libraries that read data from Cassandra or Redis were developed for graphical user interface applications running on workstations. We named the functions identical to the old libraries so that no modifications to the source code of the applications were required. The new libraries did not use the messaging system to access databases but their own database client libraries . The new client library selects one of the Redis servers randomly to read real-time data. Because of the eventual consistency of Cassandra, one may obtain inconsistent values if one accesses only one node of the cluster. The library by default reads data in quorum mode. It returns data if the data from at least two nodes are identical.

Alarm Survey

We re-implemented alarm survey system for the new MADOCA-II database. The old alarm system reads the newest log data from RDBMS, compares them to the nominal values stored in RDBMS, and writes alarm data to RDBMS if an alarm event is detected [10]. We implemented RDBMS in the system for normal value and alarm event stores because those data are suitable for RDBMS. RDBMS has better equipped to manage the relation between data and flexibility of query over NoSQL. The survey application is completely re-written using Python2.7 [11] and PyQt4 [12] for GUI. Casandra, Redis, RDBMS and MessagePack [13] libraries were open source resources. The old survey system was running in parallel for log term test. The results of both systems were compared and examined. and were found to have no difference between them.

MODIFICATION FOR PRODUCTION RUN

Before the production run, we made some modifications to the system after the one completion of the year test run.

Table Structure of Cassandra

Initially , the table structure was changed from one big table to divided smaller tables. We implemented Cassandra with one big table that stored all the data in the test run and observed that a single big table is not equipped to manage large data. Casandra writes data into set of files named SSTables. As the size of SSTables grow, those files are combined into large SSTables. This process is called compaction. Compaction requires temporary disk space with the same size of SSTables. Therefore, the size of SSTables for one big table cannot exceed more than half of the disk space as it an inefficient disk utilization method. In addition, compaction for big SSTables requires large amount of disk i/o and CPU power.

We divided a big table into small tables for each month. Owing to this modification the compaction requires less temporary disk space, disk i/o and CPU resources. Although the database management increases in complexity, the backup job for files become much simpler with smaller files. The reading libraries also need to be modified to read many tables in one function call.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Alive Flag

If an embedded computer stops, the data generator running on it will stop sending messages. The alarm system detects the trouble. A message has a timestamp embedded in its meta-data part. One may detect trouble by comparing the timestamp with the system clock of the alarm system. However, each signal has its own data cycle, therefore differences in the time time required to trigger the alarm should be set for each signals. We added a cycle period to the meta-data part of the message. Alarm system identifies the trouble when the time stamp is delayed more than ten times the cycle.

Time to Live

Many signals keep same value in accelerator control system. Especially signals for status, like on/off, remain unchanged for long period of time. We will not store them if the value of a signal is identical to the previous value to save the total amount of data stored on disk. Cassandra provides a "time to live" option for each column. The columns are marked when the time limit is exceeded and the marked columns are deleted at the next compaction job. We add time to live values to meta data. The writer reads the time to live meta data and add them to Cassandra's insert command. For data that are unchanged for long time, unchanged data are stored in the Cassandra database every five minute. The final structure of the message is shown in Fig. 2.

Cluster Expansion and Upgrade

The six node test cluster was changed to 12 nodes to store large data sets. We constructed the other 12 node production cluster, upgraded Cassandra version from 1.2 to 2.0 which enables virtual nodes (vnodes) and implemented a new table scheme. The vnode builds virtual nodes on physical nodes and enables fast data repair by using the data replicas on other vnodes. The specifications of servers are described in Table 1.

Table 1: Server specifications

Server	Dell PowerEdge R420
OS	CentOS 6.6 (64bit)
CPU	Intel Xeon E5-2420 v2, 6c, 2.2GHz
Memory	16GB
Hard disk (system)	600GB SAS 15Kr/m x1
Hard disk (data)	3TB SATA 7,200r/m x3
Cassandra version	2.0.10
JavaVM	JRE1.7.0-67-b01

Data Migration

The archived log data on old RDBMS were duplicated into the production Cassandra cluster. 4TB of data sets on RDBMS stored from 1997 to the end of 2014 became 0.75TB per Cassndra nodes. The total data size in the Cassandra cluster was 9TB, which includes three replications of data. The raw size of data in RDBMS is 4TB and becomes larger in the real RAID5 file system.

LGsr_mag_ps_b/adc_current:
{"tm":1595588400000000000, "tl":1296000, "cy":1000}
1.234565876

Figure 2: Structure of a message. The first part is a key and not Messagepacked. The second part is meta data. "tm" is timestamp in nanoseconds. "tl" is time to live in miliseconds and "cy" is data acquisition cycle in milliseconds. The last part is data. Data in second and third part is packed by Messagepack.

DIAGNOSTICS SYSTEM

We built and installed several diagnostic systems to ensure the for healthy operation of the system.

Server Resource Monitoring

Server resources are monitored by Zabbix [14]. It not only displays information acquired from SNMP (Simple network management protocol) but also the data acquired from JMX (Java Management Extensions) on Web browsers. JMX monitor the heap status of Cassandra on the Servers.

Data Acquisition Monitors

We build two monitoring systems for relay and writer processes. One is a simple system shown in Fig. 3 that displays alive or dead status of processes and runs on the display wall in the main control room to display status to operators. The other is for system experts shown in Fig. 4. It displays show message per seconds, elapsed time to write into database, and other information. The application also send/receives messages to/from relay servers for the test. Both monitor applications were written in C++ with Qt4.

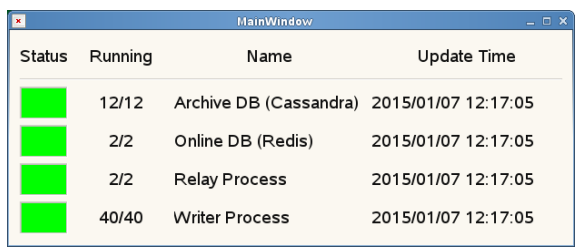


Figure 3: Screen capture of monitoring tool for operators.

Mail Notification System

Although the system has high reliability with no single point of failure, we set up a mail notification system for system troubles. System manager also received message received check mail other than trouble notifications once a day.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

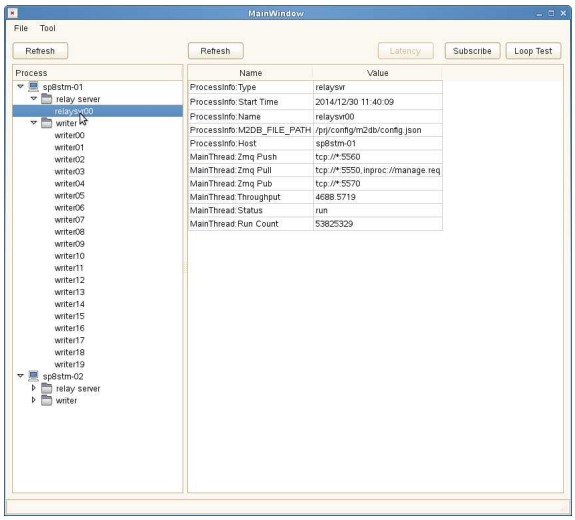


Figure 4: Screen capture of monitoring tool for expert.

CURRENT STATUS

We began operating the production system in January 2015. We have had no major trouble since then. We manage 27,000 signals on the system and handle 9,000 messages per second. We also monitor system resources like CPU loada and disk and memory usage and found no abnormal status.

The MADOCA-II database still remains a part of the old system. The system on embedded computers and cc2m2db are temporal ones that send data to old system in parallel. We will implement a new MADOCA-II dedicated data generation application for embedded computers. The data registration files for embedded computers and database systems are now separated into two files. The segregated data registration files were problematic for the old system, sometimes they are inconsistent. Therefore, we aim to unify the files in our proposed system for simplification and troubleshooting the inconsistency.

SUMMARY AND FUTURE PLANS

We started SPring-8 operation using the new MADOCA-II database. Before production, we modified the system using the knowledge obtained from long term test operation. We installed a monitoring system and re-wrote the alarm system, client libraries and Web systems. We are now planning to extend the dedicated MADOCA-II system to embedded computers and signal registration system.

REFERENCES

[1] R.Tanaka, et al., "The first operation of control system at the SPring-8 storage ring", Proceedings of ICALEPCS 1997, Beijing, China, (1997).

[2] A.Yamashita, et al., "The Database System for the SPring-8 Storage Ring Control", Proceedings of ICALEPCS 1997, Beijing, China, (1997).

[3] A.Yamashita, et al., "Data archiving and retrieval for SPring-8 acceleratorl complex", Proceedings of ICALEPCS 1999, Trieste, Italy, (1999).

[4] http://rsc.riken.jp/pdf/SPring-8-II.pdf

[5] M.Kago, et al., "Development of a Scalable and Flexible Data Logging System Using NoSQL Databases", Proceedings of ICALEPCS 2013, SanFrancisco, USA, (2013).

[6] http://cassandra.apache.org/

[7] http://redis.io

[8] A.Yamashita, et al., "A New Message-Based Data Acquisition System for Accelerator Control", Proceedings of ICALEPCS 2013, San Francisco, USA, (2013).

[9] T.Masuda, et al., "Data Acquisition System with Database at the Spring-8 Storage Ring", Proceedings of ICALEPCS 1997, Beijing, China, (1997).

[10] A.Yamashita, et al., "The alarm system for the SPring-8 storage ring", Proceedings of ICALEPCS 1997, Beijing, China, (1997).

[11] http://www.python.org

[12] http://riverbankcomputing.com/

[13] S. Furuhashi, Master Thesis, University of Tsukuba, Japan, (2012).

[14] http://www.zabbix.com/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

HDB++: A NEW ARCHIVING SYSTEM FOR TANGO

L. Pivetta, C. Scafuri, G. Scalamera, G. Strangolino, L. Zambon,
Elettra Sincrotrone Trieste, Trieste, Italy
R. Bourtembourg, J.L. Pons, P. Verdier, ESRF, Grenoble, France

Abstract

The TANGO release 8 led to several enhancements, including the adoption of the ZeroMQ library for faster and lightweigh event-driven communication. Exploiting these improved capabilities, a high performance, event-driven archiving system written in C++ has been developed. It inherits the database structure from the existing TANGO Historical Data Base (HDB) and introduces new storage architecture possibilities, better internal diagnostic capabilities and an optimized API. Its design allows storing data into traditional database management systems such as MySQL or into NoSQL database such as Apache Cassandra. The paper describes the software design of the new HDB++ archiving system, the current state of the implementation and gives some performance figures and use cases.

INTRODUCTION

The TANGO archiving system is a tool that allows to store the readings coming from a TANGO based control system into a database. The archived data are essential for the day by day operation of large facilities, such as long term monitoring of subsystems, statistics, correlation of parameters or comparison of operating setups over time.

To take advantage of the fast and lightweight event-driven communication provided by TANGO release 8 [1] with the adoption of ZeroMQ [2], a novel archiving system for the TANGO Controls framework [3] has been designed and developed in collaboration between Elettra and ESRF.

DESIGN GUIDELINES

A number of requirements have been taken into account during the design phase. The HDB++ archiving system must fully comply to the TANGO device server model, with two immediate benefits. First, all the required configuration parameters are stored to and retrieved from the TANGO database; some of these parameters are, for user convenience, duplicated into a dedicated table of the HDB++ schema by a mechanism that guarantees the consistency of the copy. Second, the HDB++ archiving system inherits the TANGO scaling capability: any number of EventSubscriber instances can be deployed according to the desired architecture and overall performance.

The publish/subscribe paradigm is available in TANGO via the event subsystem. More in detail, the *archive* event is provided for archiving purposes and can be triggered on threshold comparison and/or periodic basis. The HDB++ architecture is fully event based; therefore, a part of HDB++ setup consists of conveniently configure TANGO device servers to send events as required.

Two TANGO device servers have been developed. The EventSubscriber, also referenced as archiver, is in charge of gathering the values from the TANGO devices and storing them into the historical database. To address the requirements coming from large systems the need to distribute the workload over a number of archivers shows up. A ConfigurationManager TANGO device server will assist in the operations of adding, editing, moving and deleting an Attribute to/from the HDB++ archiving system. A specific library, exposing a suitable API, addresses the historical data extraction from the archive.

The task of each HDB++ archiving system component is sharply defined; low layer devices, e.g. archivers, have no dependency against the ConfigurationManager, the extraction library or the graphical user interfaces and, possibly, can be deployed standalone. Also, the HDB++ architecture has been designed to easily support different SQL and NoSQL database engines: an abstraction library decouples the interface to the database back-end from the implementation. Adding a new back-end is just matter of writing the code for the specific implementation; this has been done, as an example, during last year to introduce the support for Cassandra [4].

EVENT SUBSCRIBER

The EventSubscriber TANGO device server is the core of the HDB++ archiving system. It subscribes to archive events for the specified Attributes list, stored into a Property in the TANGO database, as well as a number of additional parameters, such as the hostname and port number where the back-end is running, the name of the database and the username and password to be used.

A dedicated thread is in charge of event subscription and callbacks execution; the callbacks, acting as producers, put the complete data structure of the received events in a FIFO queue, protected by a suitable locking mechanism. The thread and the callbacks must be able to handle an arbitrary number of events, possibly limited just by the available memory and the required performances. One additional thread, acting as consumer of the FIFO, is in charge of writing the data into the database. Moreover, a high-mark threshold is setup on the FIFO queue to alert for an overloaded EventSubscriber.

The EventSubscriber device server allows to perform the following operations:

- add/remove an Attribute to/from archiving
- start/stop the archiving for all Attributes
- start/stop the archiving for one Attribute
- read the status of an Attribute
- read the list of Attributes currently archived (started)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- read the list of Attributes currently not archived (stopped)
- read the number/list of Attributes in charge
- read the configuration parameters of each Attribute
- read the number/list of working Attributes
- read the number/list of faulty Attributes
- read the number/list of Attributes pending in the FIFO

Working at the EventSubscriber level implies that the database entry and the archive event parameters have to be already configured. Besides, no action is performed on the archived data when removing an Attribute, which means that the data remain available in the historical database.

Special care has been reserved to the error management. One NULL value with time stamp is inserted whenever the archiving of an Attribute is stopped due to error. Moreover, if an error occurred, the corresponding Attribute is marked as faulty in the archiving engine and the error description stored. In case the archiving was suspended due to an error, it is automatically resumed when valid data is available again. The quality factor of the Attribute is also stored into the historical database. Exploiting these features, a client could be fully aware of the archiving status of an Attribute; in addition, dedicated alarms can be configured in the TANGO Alarm System to asynchronously inform about the status of the archiver device.

The EventSubscriber TANGO device server also exposes some additional figures of merit, such as:

- for each instance, total number of records per time
- for each instance, total number of failures per time
- for each Attribute, number of records per time
- for each Attribute, number of failures per time
- for each Attribute, time stamp of last record

These numbers can sum up in a counter, which can be reset every hours/days/weeks, to rank each Attribute in term of data rate, error rate etc. This allows preventive maintenance and fine tuning, detecting, for instance, when an Attribute configuration is wrong because the variation threshold is lower than the noise level. These statistics are a key element for qualifying the health of the system. All these Attributes are archived themselves to enable a follow-up versus time. For each Attribute, the EventSubscriber TANGO device server also computes the minimum and maximum processing and storing times, which helps discovering possible bottlenecks.

CONFIGURATION MANAGER

Adding an Attribute to the archiving system may require creating the new entry into the database tables, setting up the Attribute archive event configuration and assigning the Attribute to one of the archivers. Moreover, large systems may need to distribute the workload over several EventSubscriber device servers. A special TANGO device server, the ConfigurationManager, has been developed to simplify the above steps and to help monitoring the whole HDB++ archiving system. Adding an EventSubscriber device to the ConfigurationManager pool enables the management. This leaves open the possibility of deploying un-managed archivers, if needed.

The ConfigurationManager device server is able to perform the following operations on the managed EventSubscriber pool:

- handle the request of archiving a new Attribute
  - create an entry in the database if not existing
  - setup the Attribute archive event configuration
  - assign the new Attribute to one of the archivers
- move an Attribute from one archiver to another
- show the Attribute/archiver coupling
- start/stop the archiving of an Attribute
- remove an attribute from archiving

The ConfigurationManager also exposes some Attributes to keep trace of the global statistics:

- total number of EventSubscribers
- total number of working/faulty attributes
- total number of events per second
- overall minimum and maximum processing and storing time

These attributes could be themselves archived to enable a follow up versus time. The statistics window GUI for MySQL back-end at the ESRF is shown in Fig. 1.

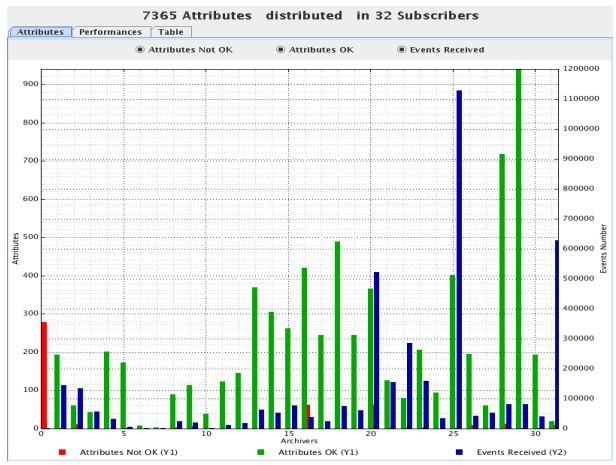


Figure 1: MySQL back-end statistics. X: archivers, Y1: Attributes [red faulty/green ok], Y2: events number [blue].

To guarantee the consistency of the archiving setup, the ConfigurationManager implements a strict sequence when adding an Attribute to the archiving system. More in detail, at first the historical database entry for the Attribute is created, then the archive event parameters are configured, or checked if already existing, and finally the Attribute is assigned to the desired archiver.

HISTORICAL DATABASE

Currently the HDB++ archiving system supports MySQL [5], a relational database management system, and Cassandra as back-ends. Cassandra is a distributed NoSQL database where the data is spread over a number of nodes. Its architecture provides replication, high availability with no single point of failure and linear scalability, meaning that

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

higher performances can be achieved simply adding more nodes. A detailed description of the Cassandra support in HDB++ is available in [6].

Some shared libraries provide the methods for writing to the database back-end. These libraries, written in C++, are addressed to the EventSubscriber TANGO device server and their main purpose is to provide an abstraction layer. Actually, some shared objects are available implementing the abstraction layer and the specific interface:

- *libhdb++*: database abstraction layer
- *libhdbmysql*: legacy HDB schema support for MySQL back-end
- *libhdb++mysql*: HDB++ schema support for MySQL back-end
- *libhdb++cassandra*: HDB++ schema support for Cassandra back-end

These libraries allow reusing the EventSubscriber, the ConfigurationManager and the GUIs without changes. The HDB++ archiving system can be easily extended to support additional database engines, such as Oracle, PostgreSQL or other NoSQL databases, just writing the specific support library.

The database schema characteristics are common to both MySQL and Cassandra back-end. The *att_conf* table associates the attribute name with a unique ID and selects the data type; it's worth noting that the *att_name* row always contains the complete FQDN, e.g. with the hostname and the domain name. The *att_history* table stores the timestamps of the operations made on each Attribute, such as adding, removing, starting or stopping the archiving. Each TANGO data type has a dedicated table containing the Attribute ID, the Attribute data timestamp, the event timestamp, the database insert timestamp and the data value. Comparing the timestamps, any possible performance bottleneck can be easily detected. With respect to the legacy HDB schema, the new HDB++ schema introduces some relevant changes:

- μs timestamp resolution
- no per-attribute additional tables; the number of tables used is fixed and does not depend on the number of archived attributes
- specific TANGO data type support

CONFIGURATION TOOLS

A graphical user interface for the ConfigurationManager has been developed. Written in Java, the HdbConfigurator GUI presents a Jive-like interface, showing on the left side the device tree and on the right side the selected archiver with the lists of the relevant started and stopped Attributes. On bottom left, the archive event parameters of the selected Attribute appear. A screenshot of the HdbConfigurator GUI is shown in Fig. 2.

To use the HdbConfigurator, the `HdbManager` environment variable, containing the ConfigurationManager TANGO device server to be used, has to be exported. Then, once the desired archiver has been selected, the device tree can be browsed for the requested Attribute. A right-click on

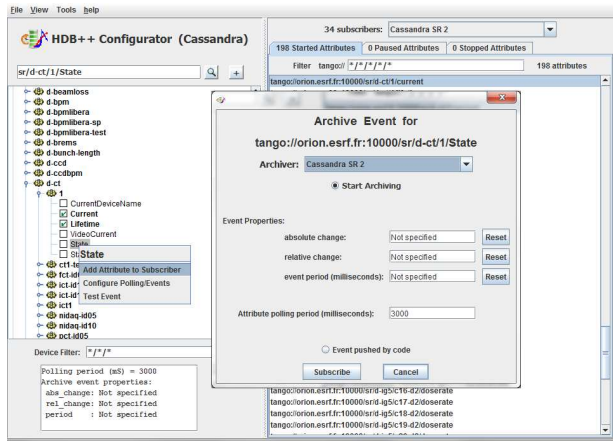


Figure 2: HdbConfigurator GUI.

the Attribute name opens a pull-down menu; in the popup window the user can specify the archive event parameters, if needed, or mark the event pushed by code flag. A list of Attributes, stored in a file, can be added to the archiving system using the File/Open menu of the HdbConfigurator GUI. Moreover, whenever a large number of Attributes has to be added to the archiving system, using a programmatic approach could be convenient. A client application for the ConfigurationManager TANGO device server can consist of a few lines of Python or Java and address in a very efficient and effective way the above requirement, especially when also the archive event parameters for each Attribute have to be specified.

DATA EXTRACTION

A specific API has been defined to address the historical data extraction. The data extraction library shall be able to deal with event based archived data. The possible lack of data inside the requested time window shall be properly managed:

- returning some *no-data-available* error: in this case the reply contains no data and an error is triggered;
- enlarging the time window itself to comprehend some archived data: the requested time interval is enlarged to include some archived data. No fake samples are introduced to fill the values in correspondence of the requested timestamps;
- returning the value of the last archived data anyhow: the requested time interval is kept and the last available data sample is returned. The validity and the consistency of the data is guaranteed when the archive event change threshold is configured; care must be taken with the dataset in case of periodic archiving event;

Two libraries have been developed implementing this API: the first, written in C++ is dedicated Qt/Qtango based GUIs or to C++ TANGO device servers; the second, written in Java, has been used for the HdbViewer GUI and is a native choice for Java device servers.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The HdbExtractor++ multithread library allows fetching the data from the legacy HDB and the new HDB++ MySQL schema in a simple Object Oriented way. An additional module provides a Qt interface to the HdbExtractor++ and a dedicated GUI, exploiting the MathGL framework, aimed at displaying mono and bidimensional data over time. Also, possible errors conveniently stored in the database can be found an displayed. Figure 3 and 4 show a multiline plot and a surface plot.

Figure 3: HdbExtractor++ multiline plot.

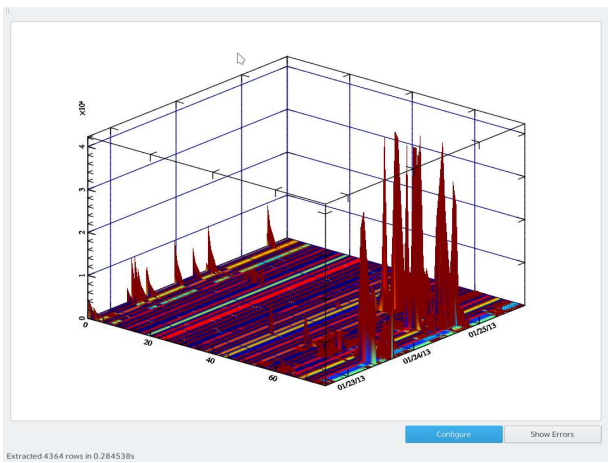


Figure 4: HdbExtractor++ surface plot.

The HdbViewer Java framework, in addition to the legacy ESRF historical database support, allows retrieving the data from the new Cassandra back-end as well as managing the Cassandra partitioning period. The classic table display of the HdbViewer GUI is shown in Fig. 5. Multiline bidimensional plots are also supported.

PERFORMANCE

Currently more than 6800 Attributes are archived with the HDB++ at Elettra, on both the legacy HDB schema and the new HDB++ schema using MySQL as back-end; respectively 30 and 20 instances of the EventSubscriber have been

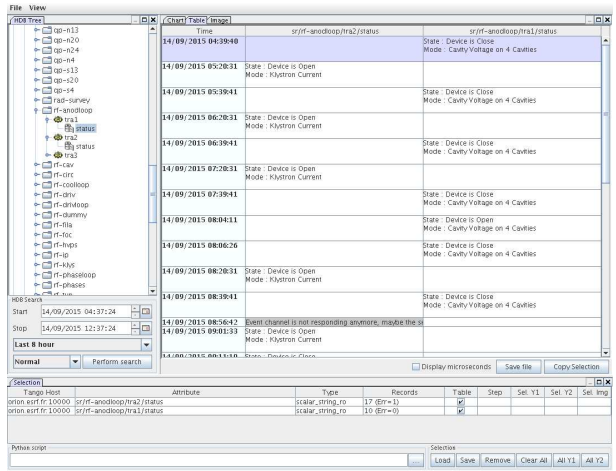


Figure 5: HdbViewer, classic table display.

deployed, each one managing a very different number of Attributes, spanning from just one to more than one thousand per archiver. A single EventSubscriber device server, running on an unloaded machine, is capable to handle thousands of events per second, sustained rate, with easy; peaks of a few tens of thousands can also be handled exploiting the FIFO for caching. However, in this scenario, or at even higher rates, care has to be taken with respect to the database back-end that can become the bottleneck.

Similarly, more than 7300 Attributes are archived with the HDB++ at the ESRF using the new HDB++ schema on both MySQL back-end, exploiting 32 archivers, and Cassandra back-end, using 34 archivers.

CONCLUSION

The HDB++ archiving system, though still under development, is in production at both Elettra and ESRF sites since almost two years. The ConfigurationManager device server in conjunction with the HdbConfigurator GUI greatly simplify the utilization. The administration of an HDB++ archiving system is quite easy, especially for anyone with some TANGO experience, although the installation is still somehow tricky. Some Debian packages are foreseen to simplify the installation procedure for selected platforms and will be made available in the next future.

REFERENCES

[1] A. Götz et al., "TANGO V8 - Another turbo charged major release", ICALEPCS'13, San Francisco, USA (2013).

[2] ZeroMQ: `http://zeromq.org`

[3] TANGO Controls: `http://www.tango-controls.org`

[4] Apache Cassandra: `http://cassandra.apache.org`

[5] MySQL: `http://dev.mysql.com`

[6] R. Bourtembourg et al. "How Cassandra improves performances and availability of HDB++ TANGO archiving system" WEM310, *These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

BIG DATA ANALYSIS & ANALYTICS WITH MATLAB®

D. Willingham#, MathWorks, Sydney, Australia

Abstract

In today's world, there is an abundance of data being generated from many different sources in various industries across engineering, science & business. For example, DESY's 1.7 mile-long PETRA III accelerator is capable of generating 20 gigabyte's of data per second [1]. Using Data Analytics to turn large volumes of complex data into actionable information can help improve design and decision making processes. Big data sets may often be too large for the available memory, take too long to import and process, or just stream too quickly to store. Standard algorithms are usually not designed to process big data sets in reasonable amounts of time or memory usage. MATLAB® is the high-level language and interactive environment used by millions of engineers and scientists worldwide. It lets you explore and visualize ideas and collaborate across disciplines including signal and image processing, communications, control systems, and computational finance. Using a generic case study we demonstrate efficient ways to manipulate, compute and visualise on large, multidimensional data sets on light weight machines. The techniques presented here can also be used to develop predictive models, scale up for high performance computing on clusters, or in the cloud and deployable within databases, Hadoop and other Big Data environments.

INTRODUCTION

Big data describes a massive volume of data that is so large or complex that it's difficult to manage and process using traditional database and software techniques [2]. Big Data is commonly described by the 3 V's

- Volume – which references the amount of data.
- Velocity – is the speed at which data is generated or needs to be analyzed.
- Variety – which describes the range of data types and sources from which data is obtained.

There are a number of challenges that arise when analyzing big data sets, especially for domain experts who are not necessarily experienced software programmers. These include:

- How to get started and gain insight into the basic structure and format of the data especially when the volume exceeds memory limits?
- Rapid exploration of the data, and development of algorithms which can easily be scaled for use with big data
- Using big data algorithms within business systems

As an interesting example of the volume and velocity of data that is being generated in the world. In 2000, the world generated 2.5 Exabyte's of data a day [3]

In this paper, we present a generic case study identifying and predicting patterns of the domestic flights in the US between 1987 and 2008, which is 12gB in size [4]. We address the challenges mentioned previously by demonstrating two capabilities of MATLAB®, namely, datastore and mapreduce and the links with Big Data storage platforms such as Hadoop.

This paper is organised as follows. We highlight 3 techniques that can be used to handle importing and processing big data;

1. Scale the problem down
2. Parallelize the Problem
3. Scale up if needed along with the MATLAB functions that support these.

TECHNIQUES FOR HANDLING BIG DATA

Scale the Problem Down

For simpler problems where data can be too big to be analyzed at once, but where distinct sub-problems can be calculated and the results easily aggregated together, we can simply loop through the various files or records, load the subset of data we're interested in, analyze it, store the result and repeat. MATLAB's `datastore` enables this to be achieved in the easy efficient way.

What is *datastore*?

A *datastore*[4] is an object for reading a single file or a collection of files or data. The *datastore* acts as a repository for data that has the same structure and formatting. For example, each file in a *datastore* must contain data of the same type (such as numeric or text) appearing in the same order, and separated by the same delimiter. A *datastore* is useful when:

- Each file in the collection might be too large to fit in memory. A *datastore* allows the ability to read and analyze data from each file in smaller portions that do fit in memory
- Files in the collection have arbitrary names. A *datastore* acts as a repository for files in one or more folders. The files are not required to have sequential names.
- Not all the data needs to be loaded into memory. For example, you may only wish to analyse 10 columns out of 50 from a data file, so only import the 10 that are needed.

david.willingham@mathworks.com.au

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Create and Read from a Datastore

Use the *datastore* function to create a *datastore*. For example, create a *datastore* from the sample file, airline.csv [5]. This file includes departure and arrival information about individual airline flights.

```
ds = datastore('airline.csv')
```

After creating the *datastore*, the data can be previewed without having to load it all into memory. Variables can be specified as columns of interest using the SelectedVariableNames property to preview or read only those variables.

```
ds.SelectedVariableNames=
('DepTime','DepDelay'};
preview(ds)


ans =
    DepTime    DepDelay
    _______    ________
     642        12
    1021         1
    2055        20
    1332        12
     629        -1
    1446        63
     928        -2
     859        -1
```

Values can be specified in the data which represent missing values. In airline.csv, missing values are represented by NA.

```
ds.TreatAsMissing = 'NA';
```

If all of the data in the *datastore* for the variables of interest fit in memory, it can be read using the readall function.

```
T = readall(ds);
```

Otherwise, read the data in smaller subsets that do fit in memory, using the read function. By default, the read function reads 20000 rows at a time. However, this value can be changed by assigning a new value to the ReadSize property.

```
ds.ReadSize = 15000;
```

Reset the *datastore* to the initial state before re-reading, using the reset function. By calling the read function within a while loop, we can perform intermediate calculations on each subset of data, and then aggregate the intermediate results at the end. This code calculates the maximum value of the DepDelay variable.

```
reset(ds)
X = [];
while hasdata(ds)
    T = read(ds);
    X(end+1) = max(T.DepDelay);
end
maxDelay = max(X)
maxDelay =
        1438
```

If the data in each individual file fits in memory, you can specify that each call to read should read one complete file rather than a specific number of rows.

```
reset(ds)
ds.ReadSize = 'file';
X = [];
while hasdata(ds)
    T = read(ds);
    X(end+1) = max(T.DepDelay);
end
maxDelay = max(X);
```

Parallelize the Problem

As the number and type of data acquisition devices grows annually, the sheer size and rate of data being collected is rapidly expanding. These big data sets can contain gigabytes or terabytes of data, and can grow on the order of megabytes or gigabytes per day. Most algorithms are not designed to process big data sets in a reasonable amount of time or with a reasonable amount of memory. *MapReduce* allows us to meet many of these challenges to gain important insights from large data sets, and importantly can be run using parallel processing on multiple cores, processors or clusters which can reduce computational time.

What is *MapReduce*?

MapReduce[6] is a programming technique for analyzing data sets that do not fit in memory.

mapreduce uses a *datastor*e to process data in small chunks that individually fit into memory. Each chunk goes through a Map phase, which formats the data to be processed. Then the intermediate data chunks go through a Reduce phase, which aggregates the intermediate results to produce a final result. The Map and Reduce phases are encoded by map and reduce functions, which are primary inputs to *mapreduce*. There are endless combinations of map and reduce functions to process data, so this technique is both flexible and extremely powerful for tackling large data processing tasks.

mapreduce lends itself to being extended to run in several environments, on a single PC, on a cluster and or integrated with Hadoop®.

Prepare Data

The first step to using *mapreduce* is to construct a *datastore* for the data set. Along with the map and reduce

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

functions, the *datastore* for a data set is a required input to *mapreduce*, since it allows *mapreduce* to process the data in chunks.

```
ds = datastore('airline.csv');
```

Write Map and Reduce Functions

The *mapreduce* function automatically calls the map and reduce functions during execution, so these functions must meet certain requirements to run properly.

1. The inputs to the map function are data, info, and intermKVStore:

- data and info are the result of a call to the read function on the input *datastore*, which *mapreduce* executes automatically before each call to the map function .
- intermKVStore is the name of the intermediate KeyValueStore object to which the map function needs to add key-value pairs. The add and addmulti functions use this object name to add key-value pairs. If none of the calls to the map function add key-value pairs to intermKVStore, then *mapreduce* does not call the reduce function and the resulting *datastore* is empty.

A simple example of a map function is:

```
function MeanDistMapFun(data, info,
intermKVStore)

    distances =
data.Distance(~isnan(data.Distance));

    sumLenValue = [sum(distances)
length(distances)];

    add(intermKVStore, 'sumAndLength',
sumLenValue);

end
```

2. The inputs to the reduce function are intermKey, intermValIter, and outKVStore:

- intermKey is for the active key added by the map function. Each call to the reduce function by mapreduce specifies a new unique key from the keys in the intermediate KeyValueStore object.
- outKVStore is the name for the final KeyValueStore object to which the reduce function needs to add key-value pairs. *mapreduce* takes the output key-value pairs from
- outKVStore and returns them in the output *datastore*, which is a KeyValueDatastore object by default. If none of the calls to the reduce function add key-value pairs to outKVStore, then *mapreduce* returns an empty *datastore*.

A simple example of a reduce function is:

```
function MeanDistReduceFun(intermKey,
intermValIter, outKVStore)

    sumLen = [0 0];

    while hasnext(intermValIter)

        sumLen = sumLen +
getnext(intermValIter);

    end

    add(outKVStore, 'Mean',
sumLen(1)/sumLen(2));

end
```

This reduce function loops through each of the distance and count values in intermValIter, keeping a running total of the distance and count after each pass. After this loop, the reduce function calculates the overall mean flight distance with a simple division, and then adds a single key to outKVStore.

Run MapReduce

After you have a *datastore*, a map function, and a reduce function, you can call *mapreduce* to perform the calculation. To calculate the average flight distance in the data set, call *mapreduce* using ds, MeanDistMapFun.m, and MeanDistReduceFun.m.

```
outds = mapreduce(ds, @MeanDistMapFun,
@MeanDistReduceFun);
********************************
*      MAPREDUCE PROGRESS      *
********************************
Map   0% Reduce   0%
Map  16% Reduce   0%
Map  32% Reduce   0%
Map  48% Reduce   0%
Map  65% Reduce   0%
Map  81% Reduce   0%
Map  97% Reduce   0%
Map 100% Reduce 100%
```

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

View Results

Use the readall function to read the key-value pairs from the output *datastore*.

```
readall(outds)

ans =

     Key          Value

    _____      __________

    'Mean'     [702.1630]
```

SCALE UP IF NEEDED

Scale Up if Needed

By default, if the Parallel Computing Toolbox is installed with MATLAB, the *MapReduce* will run in parallel on multiple cores. However in big data problems it's common for the scale of the problem to go beyond what a single machine can handle. In such situations, users should look to process the problem on a cluster (or cloud) that link up to big data storage framework, e.g. Hadoop. Note that this method does require a cluster, and works best if the analytics are deployed on the same cluster as where the data is stored.

Using the *MapReduce* and *Datastore* functionality built into MATLAB, we can develop algorithms on our desktop and directly execute them on Hadoop. To get started, access a portion of the big data stored in HDFS with the MATLAB *datastore* function, and use this data to develop *MapReduce* based algorithms in MATLAB on our desktop. We then use MATLAB Distributed Computing Server to execute the algorithms on a cluster within the Hadoop *MapReduce* framework against the full data set stored in HDFS. Additionally we could integrate MATLAB analytics with production Hadoop systems by using the–MATLAB Compiler to create applications or libraries from MATLAB MapReduce based algorithms.

CONCLUSION

While Big data represents an opportunity to gain greater insight and make more informed decisions, but it also presents introduces a number of challenges with no one size fits all solution. In this paper we presented three capabilities of MATLAB to help address these. Using *datastore* to import big data sets efficiently so that they can fit into available memory, Using m*apreduce,*to reduce the computational time to process analytics on the data through parallelization. And finally how to combine the first 2 approaches and scaling up and integrate the analytics with computational clusters and Hadoop.

Looking to the future, Big Data computing is a field that is constantly evolving due to the ever increasing amounts of data being generated each day. MATLAB's *datastore*, *mapreduce* and deployment capabilities will enable users to evolve their analytics to meet this ongoing challenge.

REFERENCES

[1] "DESY and IBM Develop Big Data Architecture for Science" 21 Aug, 2014. https://www-03.ibm.com/press/us/en/pressrelease/44587.wss

[2] Beyer, Mark. "Gartner Says Solving 'Big Data' Challenge Involves More Than Just Managing Volumes of Data". Gartner. Archived from the original on 10 July 2011.

[3] IBM press release, "IBM Expands PureSystems Family to Help Clients Tame Big Data" https://www-03.ibm.com/press/us/en/pressrelease/39039.wss, Oct 9, 2012.

[4] MathWorks Website "Getting Started with Datasore": www.mathworks.com/help/matlab/import_export/what-is-a-datastore.html

[5] United States Department of Transportation website: http://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp

[6] MathWorks Website "Getting Started with MapReduce": www.mathworks.com/help/matlab/import_export/getting-started-with-mapreduce.html

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

TIMING SYSTEMS FOR ATNF TELESCOPES

S. A. Hoyle, CASS, Parkes, NSW, Australia
P. Mirtschin, CSIRO ATNF, Narrabri, NSW, Australia

Abstract

Radio Telescopes require precise time and timing signals for accurate telescope pointing, synchronisation of signal processing instrumentation and offline manipulation of observation data. We provide an overview of the timing system in use at our observatories; briefly describing the main features of the hardware, firmware and software.

INTRODUCTION

The Australia Telescope National Facility, (ATNF) comprises the observatories located near Parkes and Narrabri in NSW and the Murchison Radio Observatory (MRO) in remote Western Australia. The Commonwealth Scientific Industrial Research Organisation (CSIRO) division of Astronomy and Space Science (CASS) manages and operates the ATNF primarily for astronomical research conducted by scientists from institutions in Australia and around the world.

Our radio telescopes vary in age and architecture: from the Parkes 64m single dish (1961), to the Australia Telescope Compact Array (ATCA) (1988) near Narrabri NSW, an array of 6 x 22m diameter moveable antennas, to the Australian Square Kilometre Array Pathfinder (ASKAP) at the MRO, an array of 36 x 12m antennas currently in the outfitting and early commissioning stage of construction. While ASKAP employs the latest technology, our older telescopes have been kept at the forefront of astronomical research due to continual upgrades to their instrumentation and extension of their operating capabilities.

Observations are made with the Compact Array and Parkes 64m telescopes both separately and together to form the Long Baseline Array. Occasionally, the array is extended further with radio telescopes in Tasmania and New Zealand. The technique of using widely separated telescopes in concert provides increased resolution and hence greater detail in the final astronomical images and is known as Very Long Baseline Interferometry (VLBI). Precise time and timing signals are required to synchronise instruments within a single antenna, between antenna elements of an array and between widely separated sites.

Figure 1 illustrates the basic instrumentation of a radio telescope and the points where time and timing signals are applied. Referring to Fig. 1: extremely weak radio frequency energy from a celestial source is converted to an electrical signal and amplified by a multi-band receiver mounted at the focal point of the antenna. The signal is then frequency down-converted and band-pass filtered before being sampled and digitised and passed to the correlator for correlation with itself, and, in the case of an array, with the signals from other antennas. The frequency conversion system requires local oscillators to 'tune' the receiver to the chosen centre frequency within each band and usually other fixed local oscillators for further stages of down conversion to an Intermediate Frequency (IF) suitable for digital sampling. All local oscillators must be precisely controlled to maintain constant phase and phase coherence between the converted signals. Lack of phase coherence results in a reduced output level at the correlator and therefore reduced sensitivity. Phase jitter reduces the quality of the final astronomical image [1]. To this end, the antenna-based oscillators are phase-locked to a highly stable central reference oscillator. In the case of an antenna array, small variations in reference phase at an antenna can occur due to movement and temperature changes in the distribution cabling. These are compensated by measuring the round-trip phase from a central point to each antenna and back; applying corrections either to the correlator output data or to the phase of the local oscillator at the antenna [1].

To attain sufficient dynamic range, the correlator accumulates the correlated input data streams for an 'integration cycle' period, of typically 5 or 10 seconds, prior to transformation to the frequency domain [1]. The integration cycle waveform, along with other real-time sequencing events supplied to the correlator, is generated by an ATNF 'Event Generator'. A Master Clock supplies precise current time to the Event Generator which is pre-programmed to set or clear its digital output event lines at absolute times corresponding to waveform transition edges.

The Antenna Control Computer also uses an Event Generator to generate the 'Start of Integration' events for the Digitiser and the 'Calibration Cycle' waveforms for the Receiver and Conversion System. Start of Integration is encoded in the IF data streams produced by the Digitiser and used to implement variable delays between the signals from an antenna array. The delays are added to compensate for the different arrival times of a celestial signal at each antenna and varied as the earth rotates to ensure alignment of samples from the same wave-front [1]. This concept is illustrated in Fig. 2 below.

The Calibration Cycle is used to determine the overall system gain and hence the true flux of the celestial source. (The system gain itself varies automatically to provide a constant input level to the samplers.) The Calibration Cycle waveform modulates a noise source of known power amplitude at the input to the receiver. The Conversion System then measures the noise on / off power levels and the difference is used to determine the gain [1].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

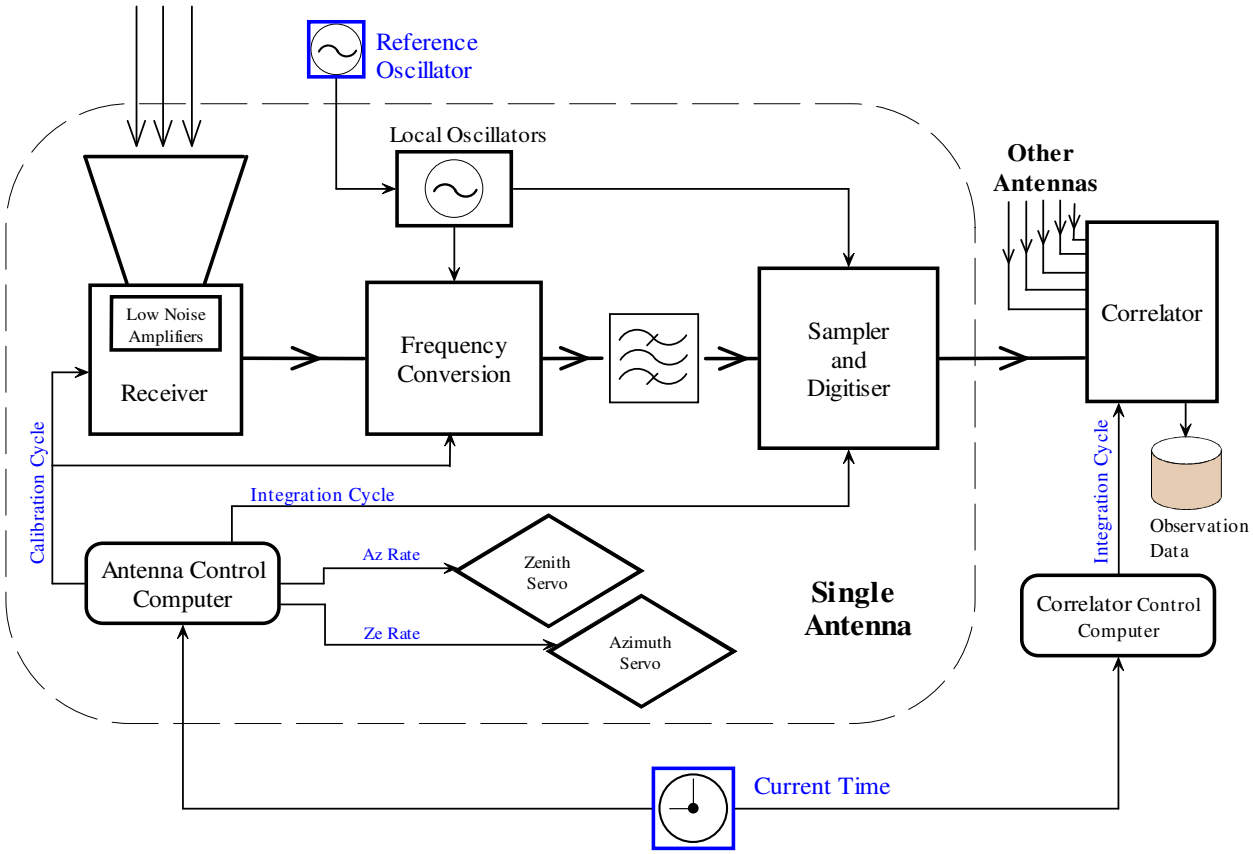


Figure 1: Radio telescope basic instrumentation.

An Event Generator is also used to synchronise a software control loop that supplies the antenna axis drive servos with azimuth and elevation coordinates to execute an antenna pointing trajectory in scanning or tracking a source. The antenna pointing coordinates themselves are determined from the current Universal Coordinated Time, the geographic coordinates of the observatory site and the Right Ascension / Declination of the source [1].

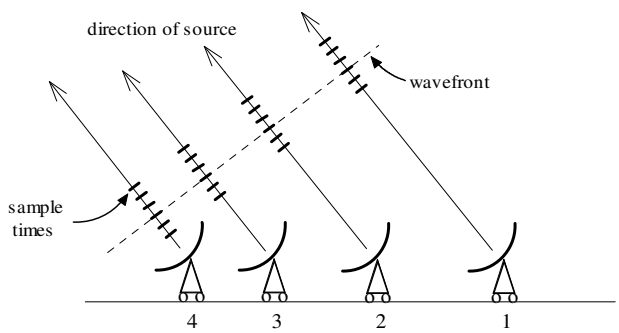


Figure 2: The wave-front concept [1].

[1] Reproduced from a drawing by Wilson W., (CASS, retired), published in [1]

TIMING SYSTEM BASIC ARCHITECTURE

Figure 3 below illustrates the basic architecture of the timing system in use at each ATNF telescope site.

A central Hydrogen Maser / Reference Oscillator provides the highly stable reference frequency, (typically 5 MHz), to phase-lock the antenna local oscillators and an 'ATNF Distributed Clock' (ATDC). The purpose of the ATDC is to generate and distribute time signals to time dependent equipment throughout an observatory site [2]. Time information is encoded in a bit stream, the 'Clock Frame', and transmitted once per millisecond over fibre optic cable or balanced copper pairs and is commonly referred to as 'Clock Bus'. The ATDC divides the input reference oscillator frequency in stages to produce a one pulse per second (1pps) output. An external 1pps from a GPS Receiver[2] provides a reference 'tick' input to the ATDC for comparison with its own 1pps. The difference, the 'Tick Phase', provides a fine-grain measure of the accuracy of the clock. In short, the GPS 1pps is the common timing reference between sites and the Reference Oscillator is the source of precision timing at each site.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

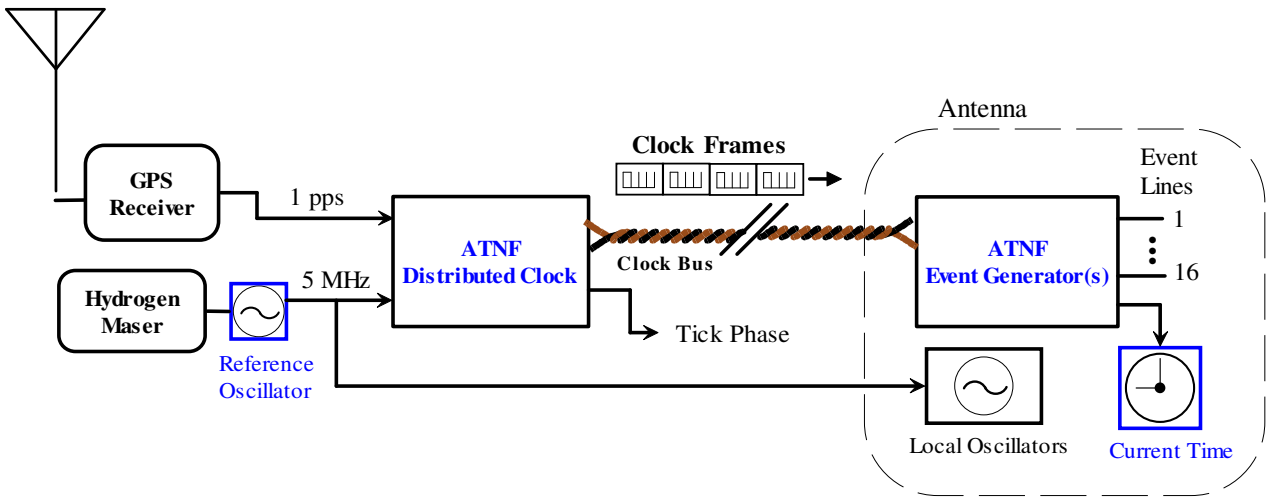


Figure 3: Basic architecture.

An 'ATNF Event Generator' (ATEG) receives and decodes the Clock Frame data. An ATEG can be pre-programmed to change the state of any of its sixteen digital output lines at an absolute time with microsecond precision [3]. Through software control it can generate pulse trains and single events with high precision and stability.

H-Maser / Reference Oscillator

There are one or more H-Masers at each observatory site. Parkes has a 'Smithsonian Astrophysical Observatory' (SAO) model VLG-10 on loan from the US Naval Observatory, while the ATCA and MRO use Russian made 'Vremya VCH-1005' H-Masers. Of the 10 or so SAO VLG-10s that were hand-made circa 1974, the one at Parkes has proven to be amongst the best in terms of long term stability. Long term timing stability is crucial for the pulsar timing observations conducted at Parkes, where pulse time-of-arrival (ToAs) for various pulsars is measured at intervals spanning years.

Variations in the thermal and magnetic environment affect the oscillation frequency of a maser cavity. In addition to their own thermal and magnetic isolation, our masers are housed in a temperature controlled room, away from the moving antenna structure. At Parkes, the temperature is maintained within a +/- 0.5 degree range.

Australia Telescope Distributed Clock

The original stand-alone version by designed Geoff Crapps[3] in the early 1980's to address the need to synchronise time between the soon-to-be-built ATCA and Parkes. The current hardware/firmware implementation is in the form of an extended Peripheral Component Interconnect (PCIe) card in a Linux host computer with kernel driver. Software utilities provide full control of the clock and retrieval of current earth and atomic time in various forms over the local area network.

The initial time need only be set to within 1 second. The clock then sets its internal counters to the correct microsecond count. Additionally, for fine adjustment, there is a facility to 'slide' the clock in 200 nanosecond steps, (1 cycle of the 5 MHz reference), to minimise the tick phase, i.e. the phase between the 1 pulse-per-second (1pps) of the external GPS signal and the clock's 1pps.

The ATDC transmits a binary sequence of time information, a clock frame every millisecond. Each clock frame consists of 55 x 16 bit words which are pulse width modulated or Manchester encoded onto a 1 MHz carrier. A clock frame includes Binary Atomic Time (BAT), Universal Coordinated Time (UTC), Universal Time (UT1), Local Mean Sidereal Time (LMST), Tick Phase as well as local time and the accumulated number of leap seconds added to UTC, (DUTC).

Binary Atomic Time is a 64 bit number representing the count of micro seconds since the epoch, MJD 0.0, i.e. 00:00hrs on Nov 17 1858. BAT is an approximation to International Atomic Time (IAT) [4], where

$$\mathrm{IAT} = \mathrm{BAT} + \mathrm{offset} + \mathrm{rate} * \Delta\mathrm{BAT}$$

Rate is the error in the frequency of the reference oscillator, offset is the tick phase. The change in tick phase is closely monitored over time and the maser cavity fine-tuned to keep rate close to zero. Since the tick phase is transmitted with each clock frame, IAT can be recovered from BAT by the receiving ATEG.

Both DUTC and daily time corrections for earth 'wobble', (DUT1), are published in the International Earth Rotation and Reference Systems Service Bulletin A (IERS-A). IERS-A corrections are downloaded once per week and applied automatically each day at midnight UTC.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The Parkes Pulsar Timing Array project (PPTA) has measured the time-of-arrival (ToA) of numerous millisecond pulsars over many years and compared the results to the predicted ToA of each pulsar [5]. The difference, known as the pulsar timing residual, for four pulsars is reproduced in the graphs in App. A [6]. The timing residuals of generally less than +/- 2 microseconds over more than a decade as shown in the graphs are only achievable due to the high stability and end-to-end precision of the ATNF timing system.

Australia Telescope Event Generator

The ATEG is currently implemented either as a PCI card in a Linux host computer, or, in the case of ASKAP, entirely in firmware embedded in instrument sub-systems, e.g. the ASKAP Digital Receivers and Correlator. A kernel driver provides the low-level software interface for the PCI card, whereas the embedded version is controlled via Ethernet with commands encoded in Universal Datagram Protocol (UDP) packets.

The main functions of an ATEG are to maintain accurate current time by 'grabbing' clock frames as they are transmitted via the clock bus and to generate timed digital events with microsecond precision at any of the sixteen digital output lines. Current BAT, UTC, LMST, etc. read from the latest clock frame are available to a host process. Additionally, interrupts can be pre-programmed to occur at any BAT, by assigning one of the output lines as a 'self-interrupt' line [7], thus allowing fine-grain timing control of software tasks and loops.

Internally, an ATEG stores event requests as paired elements of BAT and output line states in a 256 element First-In-First-Out (FIFO) register. Higher level software supplies the FIFO with a sequence of required events ahead of time. Both simple and complex pulse trains can be generated continuously provided the number of output lines in use and the pulse width are such that the host process is able to refill the FIFO before it becomes empty.

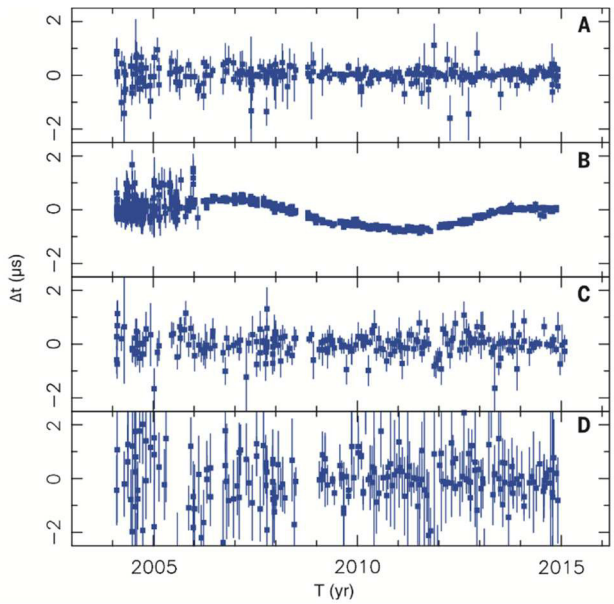
The most recent implementation of a higher level software application for the ATEG is the Event/Timing control software (EVT) developed for ASKAP. ASKAP uses the Experimental Physics Industrial Control System (EPICS) framework for its instrument monitor and control software. The EVT is an EPICS Input / Output Controller (IOC) that provides a simple, parameter based interface to produce multiple, independently controlled waveforms and soft events for synchronising remote systems via EPICS Channel Access.

ACKNOWLEDGEMENT

Thanks to Preisig B. (CASS, Parkes), Reynolds J. (CASS, Marsfield) and Phillips C. (CASS, Marsfield) for their answers to the authors questions.

APPENDIX

App A: Residual pulse times of arrival, Δt, of four pulsars observed at Parkes since 2004 [4]



REFERENCES

[1] IREE Journal of Electrical and Electronics Engineers Australia "Special Issue: The Australia Telescope" Vol 12, No. 2 June 1992

[2] Hunt A. J., "Australia Telescope Distributed Clock (MkIV)", (1994)

[3] Davis E., "ATNF PC Event Generator – Version 2: PCI Bus", (2009)

[4] Kesteven M. J., "The AT Clocks", ATNF Internal Memo AT/25.1/028, (Oct 1986)

[5] Hobbs, G., "Development of a Pulsar-Based Timescale", (2014)

[6] Shannon et al. "Gravitational waves from binary supermassive black holes missing in pulsar observations", Science, 349, 1522-152, (2015)

[7] Davis E., "ATNF PC Distributed Clock – Hardware Description", (2000)

[2] In the early years of the LBA (circa 1990), compact, low-cost GPS receivers were not available. The Masers at Parkes and the ATCA were synchronised by monitoring an ABC TV transmission signal based on a Rubidium clock.

[3] Retired, CASS, Marsfield

[4] As published in [7]. Reproduced here with kind permission of Shannon R. (CASS, Marsfield)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

VIRTUALISATION WITHIN THE CONTROL SYSTEM ENVIRONMENT AT THE AUSTRALIAN SYNCHROTRON

Ulrich Felzmann, Andrew Starritt, Nicholas Hobbs,
Synchrotron Light Source Australia, 800 Blackburn Road, Clayton 3168, Victoria, Australia

Abstract

Virtualisation technologies significantly improve efficiency and availability of computing services while reducing the total cost of ownership. Real-time computing environments used in distributed control systems require special consideration when it comes to server and application virtualisation. The EPICS environment at the Australian Synchrotron comprises more than 500 interconnected physical devices; their virtualisation holds great potential for reducing risk and maintenance.

INTRODUCTION

The Australian Synchrotron is a 3 GeV third-generation synchrotron light source built in Clayton near the Monash University Campus in Melbourne, Victoria and opened on 31 July 2007. Since 2012, the Australian Synchrotron is operated by the Australian Nuclear Science and Technology Organisation (ANSTO) under the name Synchrotron Light Source Australia (SLSA).

The core of the control system for the 100 MeV linac, the 100 MeV to 3 GeV booster, the 3 GeV storage ring and the 9 beamlines consists of more than 500 Input Output Controllers (IOCs) running EPICS. Most of those IOCs are Linux hosts (mostly CentOS), but also ~ 100 Libera systems, some Windows hosts, VME-bus computers and PLC-based controllers with EPICS embedded.

Each beamline reside in a separate subnet and VLAN and uses its dedicated set of isolated services with one service per host. Due to that philosophy, many of the IOCs do not have any special hardware attached. Those so-called "Soft IOCs" and IOCs with only serial devices attached, that can be easily connected via a Ethernet to serial converter, can be easily converted into virtual machines (VM).

VIRTUAL ENVIRONMENT

Virtualisation, which refers to the creation of an environment that acts like a real computer with an operating system, offers a great way to increase the efficiency and availability of computing services while reducing risk and the total cost of ownership..

In June 2012, the Australian Synchrotron purchased its original hardware for the virtualisation platform. Since then, the initial hardware configuration has been extended substantially and consists currently of the following individual components:

- 4 x Dell R620 hypervisors, providing in total 8 Intel Xeon CPUs (E5-2650 @ 2 GHz) and 768 GB RAM
- 3 x clustered Dell EqualLogic iSCSI devices (104 TB storage group in total), hosting the virtual disk images and frequently changing data for corporate applications that require daily replication (backup)
- 5 x EMC Isilon NL400 nodes (674 TB in total), hosting the virtual disk images for IOCs via NFS as well all the user data from the beamlines
- 2 x CISCO Nexus 5596 switches, connecting the beamlines, corporate and storage networks
- 2 x Juniper EX4300 switches, connecting the machine
- 1 x CISCO Catalyst 3750 switch, connecting the DMZ
- VMware ESXi 6 with resource pooling enabled so "Infrastructure as a service (IaaS)" can be offered to the individual beamlines.

The main components of the virtualisation environment are depicted in Figure **1**.

Apart from services for the control system environment, the virtualisation platform also hosts database and application servers for the corporate business units (email, file server, www, etc.) as well as network management (DHCP, DNS, NTP, PXE, etc.) and monitoring (Observium, Zabbix) services for the facility.

APPLICATIONS

Currently more than 200 virtual machines are active, of which approximately 50 are virtual IOCs (vIOC). In addition, there are 19 EPICS gateways, 3 EPICS CA Archivers (Figure **2**), ~ 15 build boxes, as well as VMs hosting version control, bug tracking and continuous integration testing systems.

Many of the vIOCs have been created from physical machines using the vCenter Converter from VMware, which works successfully most of the time. Some physical hosts with an old Linux operating systems such as. CentOS4 or Debian 3 required a manual conversion.

The median run-time configuration parameters for a vIOC are: 1 vCPU (9.5 MHz), 672 MB RAM allocated, 13 GB disk space used (20 GB thin-provisioned).

Further to hardware virtualisation, some VMs provide further nested virtualisation in the form of operating-system-level virtualisation**.**

- 19 EPICS Gateways providing Channel Access between the different subnets are operated as LXC instances on a single VM to reduce resource consumption.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

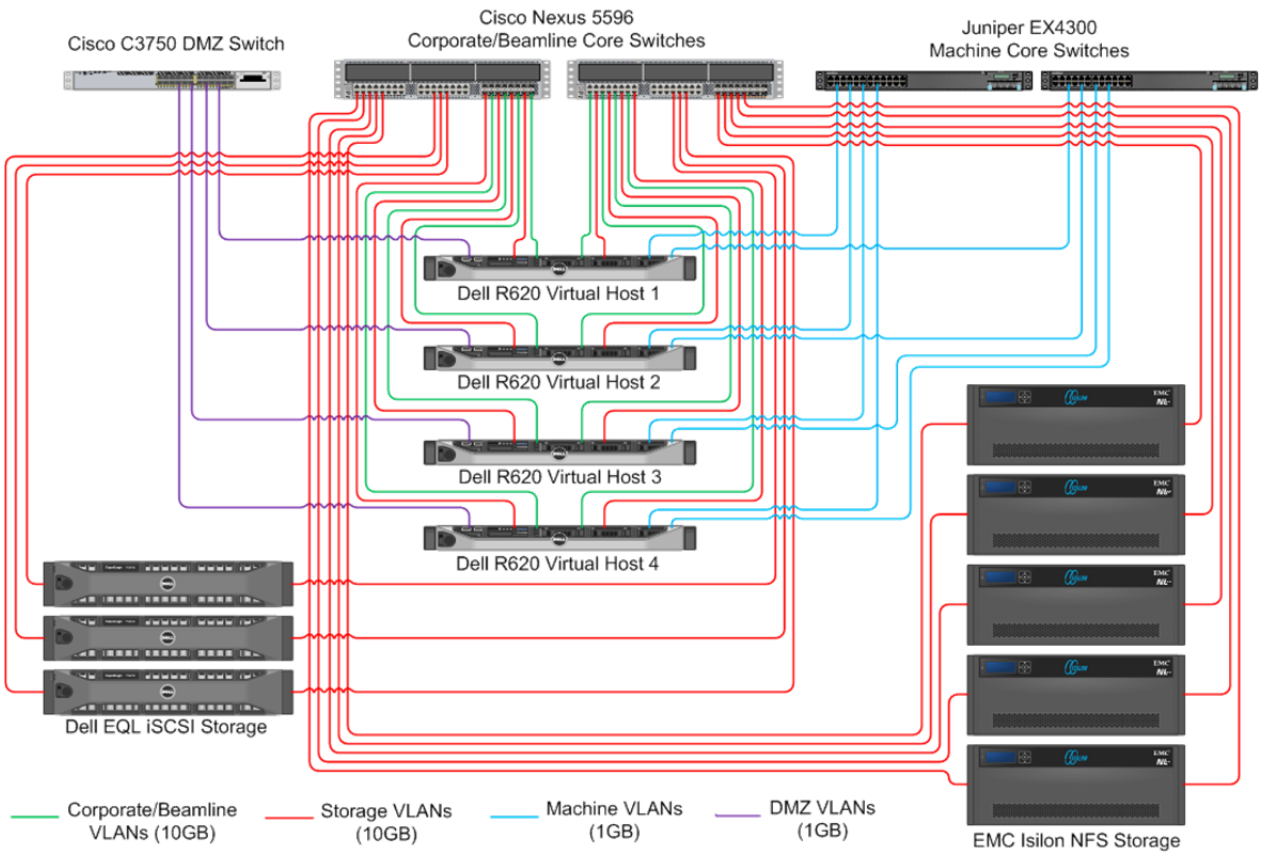


Figure 1: Schematic of the virtualisation platform of the Australian Synchrotron. The separate backup systems for the iSCSI and the NFS storage systems are omitted.

- About 10 former Cosylab µIOCs (MCS8) driving various beamline optics and originally built on Debian 3 are now run within a chroot environment on individual CentOS 6 (32bit) VMs. Due to the unavailability of the source code of the EPICS applications, copying the entire file-system into a chroot jail proved to be the most suitable method to replace those critical µIOCs, that started failing recently.

LESSONS LEARNT

With the introduction of a new technology, there are lessons to be learnt as issues and defects with the new setup and configuration are identified.

- VMs are abstract for many users and administrators, and problems with the VM itself are often wrongly attributed to the hypervisor. End users often find comfort in the fact that they can 'see and touch' physical hardware, and are resistant to migrate to a virtual platform, which is perceived as being beyond their control.

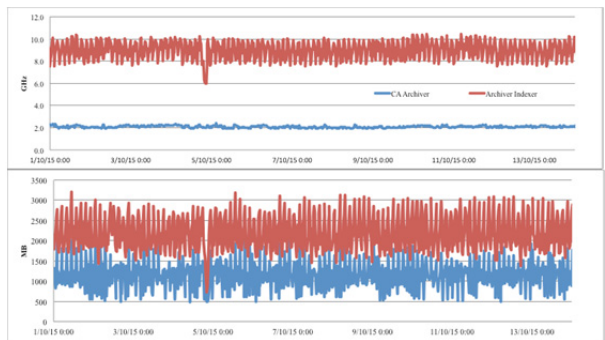


Figure 2: The most resource intense VMs are the EPICS Archiver Indexer (red) and the EPICS CA Archiver (blue).

- A failure in one hypervisor can crash many services, and the recovery can take a considerable time. Also, expert knowledge and presence is often required after a disruptive event involving the hypervisors.
- Dell Equallogic iSCSI storage is very cost-effective and provides useful features like the ability to mount iSCSI volumes within a VM, and replication and snapshotting for disaster recovery planning. However, it does not provide good redundancy and the iSCSI protocol is not very tolerant of network disruptions. EMC Isilon storage provides a higher

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

level of redundancy and the NFS protocol is more tolerant of network disruptions. The EMC Isilon cluster has proved to be a more robust storage platform for virtual control systems.

- Initially we deployed VMware vCenter Server on MS Server, with the database containing all the information about running VMs being hosted on a separate MS SQL server. Since switching to the latest OpenSUSE based vCenter Server Appliance with inbuilt vProgress database, the stability of the Virtualisation Environment has greatly improved.
- Virtual machine snapshots are very useful for rapid rollback and are the traditional method for backup processes to gain access to virtual disks. However VMware snapshotting has proved to be detrimental to EPICS processes. Instead, we utilize storage level snapshots as a means of gaining access to virtual disks for backup purposes without disrupting VM operation.

CONCLUSION AND OUTLOOK

Virtualisation has significantly improved the resilience of the control systems at the Australian Synchrotron while removing effectively high risk IOCs and reducing the overall hardware footprint tremendously. New services can be deployed and tested very easily and in a very cost-effective matter. The live migration of control systems between different hypervisors works well and VMware's resource scheduler manages the allocated resources effectively so a self-service (IaaS) is offered without the risk of one beamline consuming all the resources. To further reduce the efforts required to backup all the control system VMs frequently, a configuration management solution is being investigated so VMs can be rebuild from scratch on a short time-scale.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

STATUS MONITORING OF THE EPICS CONTROL SYSTEM AT THE CANADIAN LIGHT SOURCE

Glen Wright, Michael Bree, Canadian Light Source, Saskatoon, Saskatchewan, Canada

Abstract

The Canadian Light Source (CLS) uses the EPICS Distributed Control System (DCS) for control and feedback of a linear accelerator, booster ring, electron storage ring, and numerous x-ray beamlines. The number of host computers running EPICS IOC applications has grown to over 200, and the number of IOC applications exceeds 700. The first part of this paper will present the challenges and current efforts to monitor and report the status of the control system itself by monitoring the EPICS network traffic. This approach does not require any configuration or application modification to report the currently active applications, and then provide notification of any changes. The second part will cover the plans to use the information collected dynamically to improve upon the information gathered by process variable crawlers for an IRMIS database, with the goal to eventually replace the process variable crawlers.

BACKGROUND

The Canadian Light Source is Canada's national centre for synchrotron research and a global centre of excellence in synchrotron science and its applications. Located at the University of Saskatchewan in Saskatoon, the CLS is a world-class, state-of-the-art facility that is advancing Canadian science, enhancing the competitiveness of Canadian industry and contributing to the quality of life of people around the world.

EPICS is used at the CLS to monitor and control the electronic devices for both electron beam components and photon beam components. While there are many tools that monitor individual Process Variables (PVs) for conditions that indicate problems with individual devices, there are no tools that give an overall picture of the health of the control system itself. Early attempts at the CLS to reflect the status of the control system using an alarm handler meant that the control system was attempting to report on its own health – not an easy feat to accomplish.

Since the CLS first began operation, the monitoring of the control system health was provided by the machine operators in the control room. This was not done as a specific task, but rather occurred when a particular system was unavailable or unresponsive. The natural challenge was not all systems are being used by the operators at all times. A system failure could go unnoticed if it was not in use or directly connected to the PV alarm system. There was a strong need to improve the control system monitoring.

AVAILABLE DATA

It is important to emphasize that the information being collected does not depend on a database of running applications or available PVs. Rather, the purpose of this exercise is to provide maximum status information without having to preconfigure anything.

EPICS uses broadcast packets to indicate the continued running of server processes and when establishing communication between processes. Although the CLS uses individual Virtual Local Area Networks (VLANs) to partition the network, there are currently two systems used for EPICS gateway communication that use 802.1Q packets directly and establish numerous virtual Network Interface Cards (NICs). These two systems can then see all the broadcast packets off of all the configured networks.

The first of the two types of broadcast packets provides the following information:

- Server IP
- Server unique port number
- Packet sequence number

This information is collected to provide a summary as follows:

- The identity of each host running an EPICS server
- The number of EPICS server applications running on any host
- An estimate of how long each server application has been running
- Notification when the number of unique server applications changes
- The timestamp of when the latest packet from a server application had been processed

These packets are sent at regular intervals. These intervals can be configured differently at different sites, but the default is 30 second intervals (more frequent on start-up). Because they are UDP packets, there is a risk that the operating system will drop a packet on a congested network. A single missed packet can't be taken as a problem, but continued missed packets are an alert that something could be wrong.

The second type of broadcast packets come from EPICS client applications, and provides the following information:

- Client IP
- Client Unique TCP port number
- Requested PV name

Collecting this information provides the following summaries:

- Number of times a PV has been requested
- List of clients requesting a PV
- All PV names that have been requested

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The third type of monitored broadcast packet is the CA_PROTO_SEARCH responses from the EPICS server. These packets contain the following information:

- The server IP address
- The TCP port number of the serving application

By itself, this isn't a lot of information. By combining this with PV search requests and responses from the monitoring application itself, it can be determined which (if any) server application may have responded to the request. This now extends the information available:

- Unserved PVs
- PVs served by multiple applications

INFORMATION DISPLAY

The information collected is available at the CLS through an internal web page. A single page gives a list of all hosts known to have run or be running as a server.

INFORMATION PERSISTENCE

Currently, the information is stored at regular intervals as JSON strings.

PROGRAMMING

The system is programmed in Python. A python module (bottle.py) is used to generate the web pages.

FUTURE WORK

There is still a lot more information that is missing. The current work only finds the PVs that are used outside of an application on a regular basis. The exact identity of a client or server is not known. Performance issues with older hardware can cause the UDP beacons to not appear. Combining what is currently known with other information leads to exciting possibilities.

It is possible to run a query on a Linux host and obtain a list of EPICS clients and servers, along with their related TCP and UDP port numbers, running directories, boot time, whether they're using the ProcServ application, and the port number for connecting to the application console through ProcServ. Making this information available for simple query would allow collecting a full PV listing for any server application, and also allow tracking what client applications are referencing different PVs. This is useful when a major revision to a system changes a PV name.

The main intention of this system is as an aid to the on-call Controls staff at the CLS. While still in the early stages, this shows a great deal of promise to provide a quick diagnostic tool for determining when the control system itself has failed.

The secondary objective of this system is to eventually be able to populate an IRMIS or PV finder database. If it is simple to track when an application restarts, and what application has restarted, it follows that this would be the best time to compare the currently used server database against what is stored in a persistent database, and to note any updates. Further, having a unique identity for all running server applications and client applications would allow tracking the PV connections that are made, simplifying finding affected applications when there is a change to an EPICS record name or record type.

CONCLUSION

The information gathered by a relatively simple process has proved valuable to see the overall status of the control system. It is expected that further effort will allow this approach to provide timely information to speed system recovery from major outages, and to help identify the source of problems relating to the overall system operation.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

LabVIEW INTERFACE FOR MADOCA II WITH KEY-VALUE STORES IN MESSAGES

T. Matsumoto#, Y. Furukawa, Y. Hamada, T. Matsushita, JASRI/SPring-8, Hyogo, 679-5198, Japan

Abstract

As the next generation of the Message and Database-oriented Control Architecture (MADOCA), MADOCA II is a message-driven distributed control framework that is more accommodating, e.g., it allows control on Windows operating system as well as messages consisting of data of variable [1]. A prototype of MADOCA II was developed in the Laboratory Virtual Instrument Engineering Workbench (LabVIEW) interface in 2013 and implemented on a control system in SPring-8 [2]. However, it was recognized that the interface should be easy-to-use in order to be able to implement MADOCA II in a wide variety of LabVIEW applications, especially in synchrotron radiation experiments. In this paper, we propose a redesigned LabVIEW interface to respond to this challenge. In this interface, messaging is managed through a unified method with virtual instruments (VIs) using key-value stores. As a result, applications for the client program and the equipment management server can be easily constructed. The VIs are based on a dynamic link library (DLL) developed in C++. The interface is easily upgraded by replacing the DLL, which was also applied to the Python interface.

INTRODUCTION

The Message and Database-oriented Control Architecture (MADOCA)-II is the next generation of MADOCA. It was successfully implemented in accelerator controls in the SPring-8 and the SPring-8 Angstrom Compact Free Electron Laser (SACLA) data acquisition (DAQ) system in 2013, as reported at the last International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS) [1]. Like its predecessor [3], MADOCA II is based on a message-oriented control scheme. A message is composed of a text in subject/verb/object/complement (S/V/O/C) syntax. In this context, “S” denotes the program that is automatically defined by the framework and “V” denotes the action of a command, for which “put” or “get” is primarily used. “O” is an object name, which identifies the target of the message, and “C” is an action parameter. Figure 1 shows the structure of MADOCA II messaging. A graphical user interface (GUI) sends a message with “put/camera/init” as V/O/C (S is abbreviated as described), and an equipment manager (EM) of device control responds to the message in the S/V/O/C format. If the initialization of the camera is successful, C is set to “ok” in the response message. Thus, this messaging scheme is easy to understand.

Moreover, MADOCA II is more flexible than MADOCA. In MADOCA II, data strings of varying length, such image data, can be attached to a message, and the Windows environment can be used for device control. These features were utilized in control applications in SPring-8.

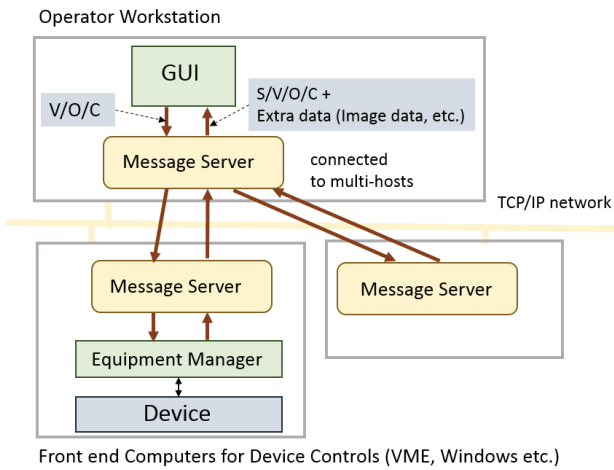


Figure 1: The structure of MADOCA II messaging. Messages are exchanged between a GUI and an equipment manager (EM) thorough a Message Server (MS) in each host.

A motivation underlying the development of MADOCA II was the ability to use the Laboratory Virtual Instrument Engineering Workbench (LabVIEW) interface by using the above flexibilities because LabVIEW is used for some parts in the accelerator control and many applications in synchrotron radiation experiments. If these LabVIEW applications are updated to contain an interface for MADOCA II, we can unify its control procedures in order to save on management costs.

In 2013, we developed a prototype of the LabVIEW interface for MADOCA II, and applied it to a beam position monitor (BPM) with NI PXI-5922 digitizers [2]. In the application, waveform data was attached to a message. A total of 5,000 samples of the waveform could be exchanged between a client application and an EM of the BPM within one second, and the BPM application functioned stably.

We intend to apply the MADOCA II LabVIEW interface to other LabVIEW applications. However, we realize that the interface should be revised for ease of use and better maintainability.

#matumot@spring8.or.jp

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The prototype of the MADOCA II LabVIEW interface was implemented by porting the algorithm of MADOCA II with LabVIEW, excluding LabVIEW-zmq binding [4] and Message Pack for LabVIEW [5]. We occasionally needed to control the version of ZeroMQ and MessagePack used in MADOCA II. However, this was difficult because we did not maintain LabVIEW-zmq binding and Message Pack for LabVIEW. Furthermore, updating the library required elaborate work because we needed to care both for the library for C++ and LabVIEW. We redesigned the LabVIEW interface for MADOCA II to solve these problems. In this paper, we describe this redesigned MADOCA II LabVIEW interface and its applications.

DESIGN OF MADOCA II LabVIEW INTERFACE

We redesigned the MADOCA II LabVIEW interface for better maintainability and ease of use. With regard to maintainability, we developed the LabVIEW interface using a Dynamic Link Library (DLL). The virtual instruments (VIs) call the DLL to use functions in MADOCA II. Since the DLL is constructed with the MADOCA II library in C++, the algorithm for MADOCA II can be shared between LabVIEW and other applications in C++. The MADOCA II library can be easily upgraded by replacing the DLL. We can also manage the versions in ZeroMQ and the Message Pack at the same time.

With regard to ease of use of the interface, we decomposed the messaging steps in MADOCA II and implemented each step through a corresponding VI. We can easily construct an application by arranging these VIs. To simplify the interface, we introduced VIs with key-value store to manage information contained in the messages. We prepared an internal buffer in the DLL. When we build message information using a VI, the information is stored in the internal buffer. The contents can be obtained from other VIs by fetching the information in the buffer.

In MADOCA II, we can treat messages containing data of variable length, such as image data. This feature is useful, but developing an easy-to-use interface for this is challenging. Using the key-value stores, we can develop a unified method to access this information, and it becomes very easy to handle various kinds of data with this messaging system.

The DLL used in the LabVIEW was applied using C++ and Python as well. MADOCA II applications can be easily built from other languages in a similar manner. The Python interface was prepared by using ctypes modules. MADOCA II can be easily used through the script language.

The details of the LabVIEW interface are shown below.

Implementation of VIs for Applications

We now describe the implementation of VIs for a client application and an equipment management (EM) application. Figure 2 shows a flowchart of a client application. The steps involved in messaging are shown with corresponding icons for the VIs. “MS2_OPEN” first connects to an MS. At this time, an internal buffer is created, followed by a handle to share information with other VIs. Following this, the internal buffer is initialized with “MSG_BUILD_INIT.” The message is then built using “MSG_BUILD_INF.” For example, we can set the key as “VOC” and the value to “put/camera/init.”

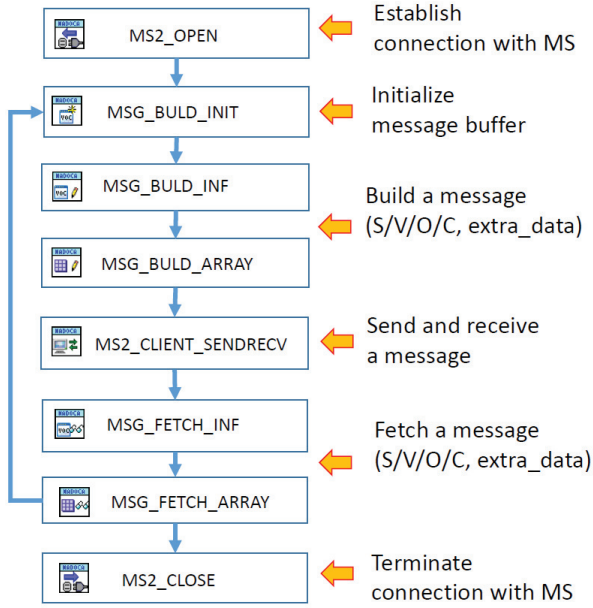


Figure 2: Flowchart of a client application in MADOCA II.

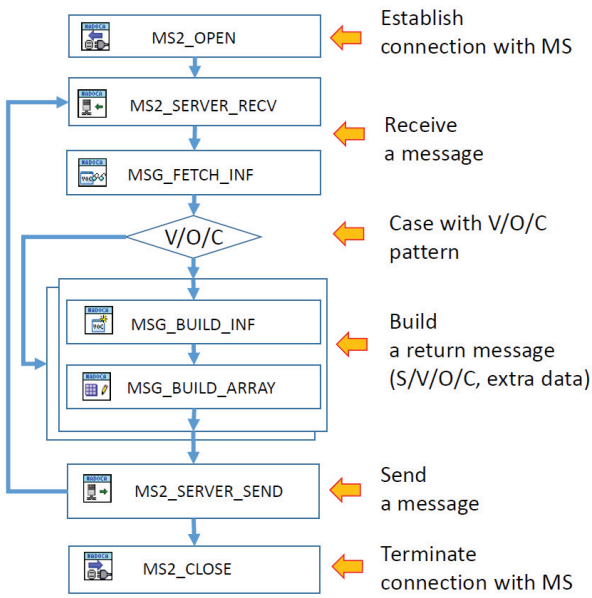


Figure 3: Flowchart of an equipment manager server application in MADOCA II.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The extra data attached to the message can be built using "MSG_BUILD_ARRAY." For example, we set the key as "image_data" and a value for the relevant array data. Various data types can be specified, such as integer and float. Following the construction of the message, we send and receive messages using "MS2_CLIENT_SENDRECV." The received messages are stored in the internal buffer in the DLL. "MSG_FETCH_INF" and "MSG_FETCH_ARRAY" can then be used to fetch required information. Finally, "MS2_CLOSE" terminates the connection with the MS.

The case of a flowchart for an EM application is shown in Figure 3. In case of an EM, "MS2_OPEN" is also used to form a connection with an MS, but the object list managed in the EM is set at the same time. "MS2_SERVER_RECV" is then used to receive messages from client applications. Following the receipt of a message, information is stored in a buffer in the DLL and fetched through "MSG_FETCH_INF." The response of an EM is determined by checking the V/O/C pattern of the relevant message, and the reply is constructed using "MSG_BUILD_INF" and "MSG_BUILD_ARRAY." Finally, "MS2_SERVER_SEND" sends the response to client applications.

Thus, we can easily construct LabVIEW programs for client applications and EM applications.

VI Components of MADOCA II LabVIEW

A total of 23 VIs were implemented for the LabVIEW interface of MADOCA II. Some of these VIs are shown in Figure 2 and Figure 3. Icons were prepared for each VI in order to easily identify its use. We also prepared VIs to treat various types of extra data, including hierarchical structures.

An error handle was attached to each VI of the interface in order to identify the error. Once we find the error, the error content can be obtained through text from other VIs to understand its cause during message routing in MADOCA II.

Robustness against Control Troubles

We designed the MADOCA II LabVIEW interface to be robust against control issues for an easy-to-use interface. In MADOCA II, messages are exchanged by using object information registered in the MSs. Object information was registered to an MS when an EM started. However, if the EM terminated unexpectedly, we often found that object information was not cleared from the relevant MS. In this case, we could not restart the EM unless we manually cleared object information from the MS, since a duplication of the object in an MS is forbidden. To solve the problem, we first cleared object information related to the application, and then restart the EM. Thus, we avoided problems related to use in MADOCA II.

Available Data Format in MADOCA II

In MADOCA II, extra data such as image and waveform data can be attached to a message. The extra data can be built and fetched with key-value stores in the MADOCA II LabVIEW interface. However, it is important to have unified data format to share information with different applications. Therefore, we defined data formats for several data types, such as image, waveform, camera controls, etc. An example of the data format for image data is shown in Table 1. In case of image data, the data type was defined by the key "image_data_type," and image size was defined by "image_width," "image_height," and "image_depth." The data type of the image was defined by "image_num_type," and the image array data was defined by "image_data." Data format was also applied to the MADOCA II library with C++ and used in a control application for image transfer in a two-dimensional interferometer in SPring-8 [6].

Table 1: An Example of Data Format for Image Data

Key	Data type	Value
image_data_type	string	"MONO","RGB","RGBA"
image_width	int32_t	
image_height	int32_t	
image_depth	int32_t	
image_num_type	string	"uint8_t","uint16_t","uint32_t","uint64_t","int16_t","int32_t","int64_t","float","double"
image_data	defined by image_num_type	
image_pixel_order	string	"lefttop," "leftbottom" ("lefttop" if not defined)

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

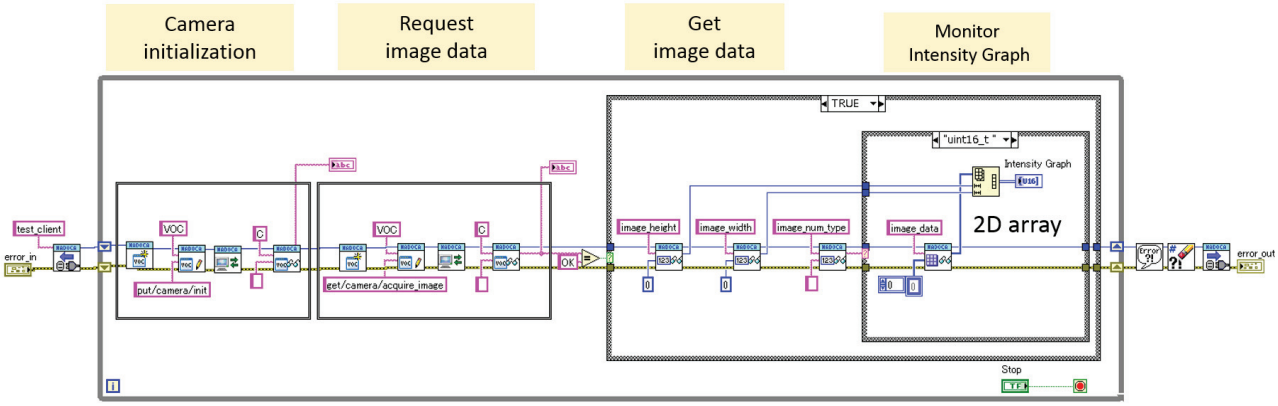


Figure 4: An example of a client program to monitor a camera image with MADOCA II LabVIEW interface. The camera images were remotely obtained through MADOCA II messaging from a camera server.

CAMERA IMAGE VIEWER WITH MADOCA II LabVIEW INTERFACE

Figure 4 shows an example of a client application for a camera image viewer using the MADOCA II LabVIEW interface. The application performs remote control of a camera and monitors the intensity graph of the resulting image. In the application, the camera is initialized with a "put/camera/init" command, and image data is requested with a "get/camera/aquire_image" command. The obtained data is stored in an internal buffer in the DLL and fetched by specifying the corresponding keys: "image_height," "image_width," "image_num_type," and "image_data." Then, the obtained data array is monitored as an intensity graph with LabVIEW. Such an application can be easily programmed with the MADOCA II LabVIEW interface.

For synchrotron radiation experiments, we tested an application using a high-sensitivity camera, the Hamamatsu ORCA-Flash 4.0. We prepared a camera server with an EM for ORCA-Flash and monitored the camera image from another client PC with the same application as shown in Figure 4. The PCs were connected to a 1 GbE network switch under a local network. A camera image of 4 MP could be monitored with a few Hz rate.

In the MADOCA II LabVIEW interface, we used an internal buffer in the DLL for ease of use with key-value stores. The control application may be not suitable for quick control because we the internal buffer is mediated during messaging. However, we can apply the interface for slow control applications, such as monitoring and device controls in SPring-8.

SUMMARY

We redesigned MADOCA II LabVIEW interface for ease of use and better maintainability. We applied key-value stores to VIs to easily manage various data in messaging with a unified method. The LabVIEW interface is based on the DLL and can be easily upgraded by replacing it. The DLL is also applicable to other languages, such as C++ and Python. With the redesigned interface, MADOCA II can be easily applied to LabVIEW and enables rapid programming. We will apply the redesigned LabVIEW interface to several control applications in SPring-8 in future research. We are preparing a LabVIEW application with an interface for a monitoring system for the NewSUBARU accelerator. We also plan to add functions to the LabVIEW interface for greater flexibility in control applications.

ACKNOWLEDGEMENT

We are grateful to Chuo Electric Works Ltd. for developing the LabVIEW interface for MADOCA II.

REFERENCES

[1] T. Matsumoto et al., "Next-generation MADOCA for SPring-8 control framework", Proceedings of ICALEPCS 2013, San Francisco, USA, (2013) p. 944.
[2] Y. Furukawa et al., "MADOCA II Interface for LabVIEW", Proceedings of ICALEPCS 2013, San Francisco, California, USA, (2013) p. 410.
[3] R.Tanaka et al., "The first operation of control system at the SPring-8 storage ring", Proceedings of ICALEPCS'97, Beijing, China (1997) p.1.
[4] http://zermq.org/
[5] http://msgpack.org/
[6] A. Kiyomichi et al., "Development of MicroTCA-based image processing system at SPring-8", Proceedings of ICALEPCS 2013, San Francisco, California, USA, (2013) p. 78.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CUSTOM HARDWARE PLATFORM BASED ON INTEL EDISON MODULE*

D. Pedretti, D. Bortolato, F. Gelain, M. Giacchini, D. Marcato, M. Montis,
S. Pavinato, J. A. Vasquez, INFN-LNL, Legnaro, Italy
M. Bellato, R. Isocrate, INFN Sez. di Padova, Padova, Italy

Abstract

The Computer-on-Module (COM) approach makes cutting edge technology easily accessible and lowers the entry barriers to anyone prototyping and developing embedded systems. Furthermore, it is possible to add all the system specific functionalities to the generic PC functions which are readily available in an off-the-shelf core module, reducing the time to market and enhancing the creativity of system engineers. The purpose of this paper is to show a custom hardware platform based on the tiny and low power Intel Edison compute module [1], which uses a 22 nm Intel processing core and contains connectivity elements to ensure device-to-device and device-to-cloud connectivity. The Intel Edison carrier board designed is expected to act as a local intelligent node, a readily available custom EPICS IOC [2] for extending the control reach to small appliances in the context of the SPES [1] project. The board acts as an Ethernet to RS232/RS422 interface translator with Power-Over-Ethernet supply and network booting as key features of this platform. The x86 architecture of the Edison makes standard Linux software deployment straightforward. Currently the board is in advanced prototyping stage.

INTRODUCTION

Nowadays embedded systems are commonly found in consumer, industrial, automotive, medical, commercial and military applications. An embedded system is hardware and software designed to perform specific tasks but we introduce here a custom hardware platform that is based on a commodity computing platform, thus partially bridging the gap between custom development and commercial off-the-shelf personal computers. The possibility to build an embedded system around a Computer on Module, for example a commercial and highly compact computing platform, represents a step towards a general purpose embedded system.

BOARD OVERVIEW

Particle accelerators need a distributed control network capable to reach devices of different types and requirements. In this context a certain effort is required for embedding the control of a single appliance or a small group of appliances. The custom hardware platform object of this paper has been inspired by the necessity to extend the control reach to small groups of magnet power supplies around the SPES accelerator, actually in construction at LNL (Laboratori Nazionali Legnaro) [3]. Magnet power supply boxes are accessible via serial buses, so we need a low power and low cost microprocessor board running the EPICS IOC software together with one or more RS232/RS422 interfaces. Although the magnet control is the target application, we realized that a more general approach opens the possibility of embedding different instrumentation around the accelerator, preserving the investment.

A Desktop PC

This embedded system is built around the Intel Edison compute module which integrates a 22 nm Intel Atom Processor dual-core 500 MHz. Lot of general purpose PC functionalities like a Wi-Fi dual-band (IEEE 802.11a/b/g/n) controller, an USB 2.0 OTG (On The Go) controller and 40 GPIO (General Purpose Input Output) configurable as I2C, I2S, SPI, SD card, UART, PWM are readily available via a 70 pin board to board connector. This low power Computer on Module is easily found on the market and we decided to use it as the core of our custom platform.

Figure 1: Intel Edison.

Figure 1 shows the tiny processor. Its x86 architecture brings lot of benefits to the embedded controller, which becomes in all respects a custom, low cost, low power and easy to use PC.

Main Features

Figure 2 shows a block diagram resuming the key features of this custom hardware platform. The SPES control system is an Ethernet based distributed network and implements the client-server model foreseen by EPICS architecture. Hence Ethernet connectivity is a key feature of the board. Since Edison Module does not integrate an Ethernet physical layer and neither an Ethernet data link layer we added an Hi-Speed USB 2.0 to 10/100/1000 Gigabit Ethernet controller

* Thanks to Prime Elettronica for assembly the prototype and to DPE Elettronica and Link Engineering for supporting the Place&Route.

[1] SPES (Selective Production of Exotic Species) is a second generation ISOL radioactive ion beam facility. The aim of SPES is to provide high intensity and high-quality beams of neutron-rich nuclei to perform forefront research in nuclear structure, reaction dynamics and inter-disciplinary fields.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

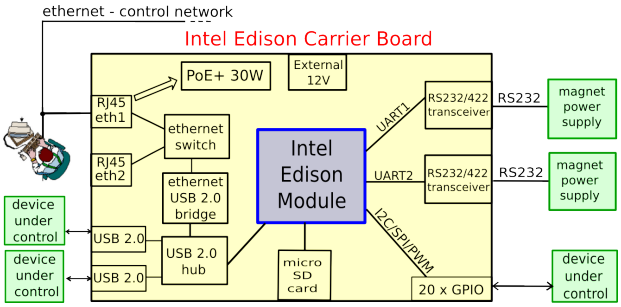


Figure 2: Block diagram.

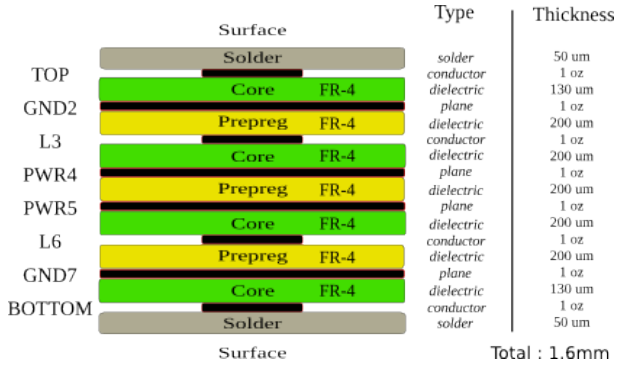


Figure 3: Stack-up.

providing high-performance and cost-effective connectivity solution. Power Over Ethernet (PoE) [4] is a standard for power sourcing devices through Ethernet cabling, up to 30 W (PoE+), that is widely used in IP phones and security cameras. Thanks to the PoE+ capability, this hardware platform can be powered via Ethernet network avoiding to lay cables around the complex and avoiding any modification to the power supply system of devices under control. PoE+ is intended as a 30 W primary DC power source. A secondary power source is provided via a standard 12 V ATX P4 connector. This connector provides a bidirectional DC power way; by powering the carrier via PoE+ the user can extract up to 24 W from the ATX-P4 connector. This is really useful if one needs to power up a non PoE+ device located at the proximity of the Edison carrier board. Furthermore a switch Ethernet ensures the possibility to daisy-chain two Edison carrier boards. Two RS232/RS422 transceivers, directly linked via UART to the processor, have been integrated in the carrier to implement the magnet control system. To minimize the area, no mechanical disk is provided. A flash memory in micro SD form factor is available as a boot device or for logging selected data. A dual stacked type A USB 2.0 connector together with the 20 GPIO header make the system general purpose. An USB hub extends the number of ports available to the processor (Edison provides only one native USB port). Moreover Wi-Fi connectivity makes the custom hardware suitable for applications requiring galvanic isolation.

LAYOUT

The board outline has been set to 132 mm x 72 mm. This small form factor complies with DIN-RAIL mounting, making the embedded controller easy to install on the field. The planning of the Printed Circuit Board (PCB) stack-up took some time and care due to the area and density constraints. A draft of the placement, together with an overview of the distribution of power nets and differential pairs, leaded us to route the PCB on 8 layers as shown in Figure 3. Since there are not special requirements in terms of dielectric, FR-4 glass epoxy has been used (dielectric constant 4.5, loss tangent 0.035). By importing the stack-up in an impedance simulator tool, it is possible to simulate high speed signals requiring a controlled differential impedance.

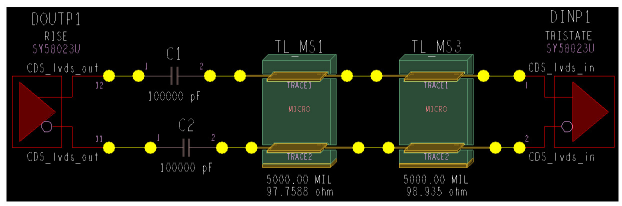


Figure 4: Impedance simulation.

Figure 4 shows a differential pair routed in part on the TOP layer and in part on an inner layer. Stack-up parameters are essential to tune the line width in order to minimize the impedance mismatch seen by the signal crossing the layer change. Importing IBIS models of the driver and receiver stages, provided by manufacturers, together with the proper source or load terminations we could simulate and perform a behavioral validation of the design. Figure 5 depicts the final board layout; only conductor layers are shown. (Top, L3, L6, Bottom). Vias in pad as well as buried vias and blind vias technologies have been avoided.

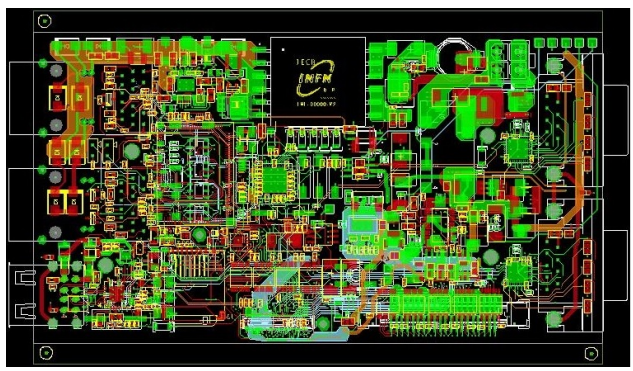


Figure 5: Layout.

Signal and Power Integrity Analysis

Before hardware prototyping we carried out post-layout simulations. This step is crucial in order to foresee signal and power integrity problems [5]. We imported the layout in specialized tools that, by means of full-wave finite element algorithms, are able to compute trace characteristics, discontinuity reflections and cross coupling. Both time and

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

frequency domain analysis have been performed on the high speed differential traces. In particular each differential pair has been modeled as a two-port circuit with the related scattering matrix. S-parameters analysis highlights reflections and thus losses due to impedance discontinuities. This analysis is especially useful in multilayer PCB having fragmented reference planes. Afterwards, resonant modes have been investigated. A resonance may lead a net to perform differently from our expectation. Figure 6 shows the areas concerned by natural resonances at 4.8 GHz in the Intel Edison carrier board. The image plots the voltage difference between GND2 and PWR4 planes, where a red color indicates higher voltage amplitude. Since the working frequencies are much below the first resonant mode and are confined in not crucial areas, we decided to tolerate them.

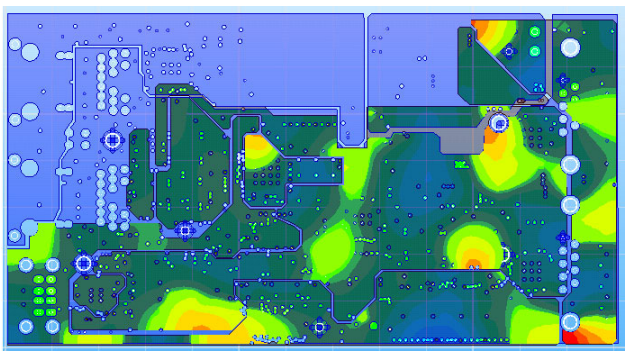


Figure 6: Cavity resonant plot.

Signal and power integrity analysis highlight potentially risky performance due to physical structures on the PCB. However there are also functional problems that may afflict system performance and cannot be included in these analysis. An example follows.

Network Performance

To meet low power requirements and to optimize the thermal profile of the board we used switching regulators and faced noise problems during the debug of the first prototype. We observed poor network performance and dropped packets due to voltage ripple on the 1.2 V power supply of the Ethernet switch's core. A revision of the DC distribution with an improved filtering stage fixed the communication problems. Figure 7 shows the filtered voltage together with the 100 Mbps Ethernet physical layer and the final network test as a proof of the increased data throughput.

SOFTWARE DEVELOPMENT

Intel Edison makes application development straightforward thanks to its full x86 compatibility. Linux or Windows applications can run out of the box without the need to recompile them. By flashing the Edison Module with the latest firmware release (provided by Intel), which is a Yocto [6] complete image, one can bootstrap the target Linux distribution and (by means of "chroot" system call) change the apparent root directory for the current running process to the

Figure 7: Network performance.

chosen Linux distribution. The original image may be replaced with a custom Linux-based system built using Yocto Project tool-set. Another possibility is to make a bootable micro SD card. On the embedded controller we successfully run an EPICS software IOC which loads a database collecting all the local process variables. This fully meets our requirements on the field. The application control GUI is remote in the control room by means of the EPICS client-server access channel structure.

Real Time Peripherals Control

Real time requirements are often met in the control of a particle accelerator. To accommodate this need we have two options in our board: either installing a real time operating system (as Wind River VxWorks for instance) with EPICS support or using the dedicated MCU (microcontroller) integrated in the Edison Module besides the Atom processor. The MCU provides developers with simple and real-time peripheral control capabilities for GPIOs, UARTs, and I2C interfaces further reducing the power consumption. The MCU application runs on the Viper kernel and independently controls the peripherals that connect to the MCU. The microprocessor can communicate with the host Intel Atom processor and wakes it up when needed.

General Purpose I/O - Bit Banging

Twenty digital I/O with selectable voltage level are available on the card and directly mapped to the Edison Module. We used them as I/O pins for digital signals acquisition and as control pins emulating I2C and SPI buses through a simple bit banging technique in python modules.

CONCLUSION

Computer-on-Module approach greatly reduces the development time and costs of embedded systems. The prototype features low consumption (below 3 W with no USB devices attached), and it proved to be an adequate solution for embedding the control of different devices in our accelerator complex: oscilloscopes, magnet power supplies, stepper motors and further appliances accessible via RS232/RS422,

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

SPI, I2C, USB, Ethernet. Figure 8 and Figure 9 depict the Intel Edison carrier board.

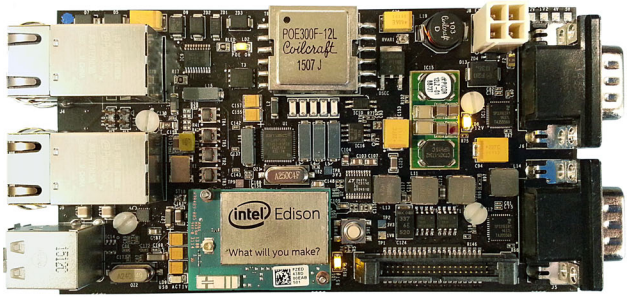


Figure 8: Intel Edison carrier board – top view.

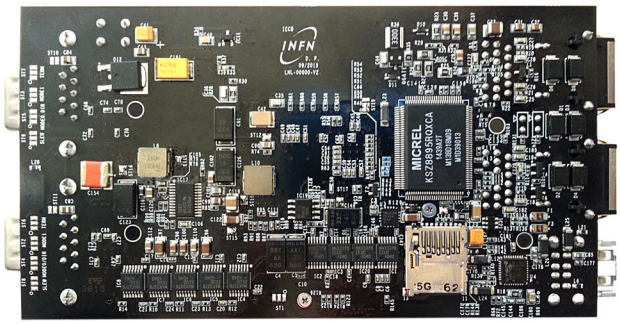


Figure 9: Intel Edison carrier board – bottom view.

REFERENCES

[1] Intel, Intel Edison Module - Hardware Guide, September 2014, Revision 002.

[2] EPICS website: `http://www.aps.anl.gov/epics` `http://www.lnl.infn.it/~epics/archiver.html`

[3] INFN - LNL, SPES project website: `http://www.lnl.infn.it/~spes/`

[4] Microsemi, Understanding 802.3at - PoE Plus Standard Increase Available Power, June 2011.

[5] H. J., M. G., High Speed Digital Design: A Handbook of Black Magic, January 1 1993.

[6] Yocto project website: `https://www.yoctoproject.org`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A MULTI-MODAL HUMAN-MACHINE-INTERFACE FOR ACCELERATOR OPERATION AND MAINTENANCE APPLICATIONS

R. Bacher, DESY, Hamburg, Germany

Abstract

The advent of advanced mobile, gaming and augmented reality devices provides users with novel interaction modalities. Today's accelerator control applications do not provide features such as speech, finger- and hand-gesture recognition or even gaze detection. Their look-and-feel and handling are typically optimized for mouse-based interactions and are not well suited for the specific requirements of more complex interaction modalities. This paper describes the conceptual design and implementation of an intuitive single-user, multi-modal human-machine interface for accelerator operation and maintenance applications. The interface seamlessly combines standard actions (mouse), actions associated with 2D single- / multi-finger gestures (touch sensitive display) and 3D single- / multi-finger and hand gestures (motion controller), and spoken commands (speech recognition system). It will become an integral part of the web-based, platform-neutral Web2cToGo framework belonging to the Web2cToolkit suite and will be applicable for desktop and notebook computers, tablet computers and smartphones, and even see-through augmented reality glasses.

INTRODUCTION

In a control room the mouse is the standard user input device to interact with accelerator control applications. Being well accepted by the operators it provides a very accurate pointing capability even onto complex applications with a wealth of graphical widgets. However, hardware commissioning and maintenance use cases might profit from novel interaction capabilities (modalities). For instance the alignment of mirrors mounted on an optical table to adjust a laser beam spot often requires a "third hand". Interacting with control applications via spoken commands could be an appropriate alternative. Likewise remote-controlling a head-mounted display showing some documentation or control application panels might be considerably simplified by recognizing spatial gestures such as clenching a fist or snapping fingers.

This paper reports the conceptual design and the implementation of a single-user human-machine-interface (HMI) for accelerator operation and maintenance applications in the context of the Web2cToolkit Web service collection [1,6]. Web2cHMI defines a set of common user interactions associated with various modalities including flat or spatial gestures, spoken commands, and ordinary mouse or touch actions. This approach applies to desktop or notebook computers with passive or touch-sensitive display, tablet computers or smartphones, and even see-through Augmented Reality glasses.

Web2cToolkit is a collection of Web services including among others Web2cViewer and Web2cToGo. Both Web applications implement Web2cHMI. The Web2cViewer Web service provides a user-configurable interactive synoptic live display to visualize and control accelerator or beam line equipment. The Web2cToGo client application is especially designed for mobile devices and is capable of running simultaneously up to 15 multi-page Web2cToolkit-compliant Web applications such as Web2cViewer applications. At any particular time the selected page of the selected user application is displayed while other pages of the same application and the pages of the other applications are hidden. The Web2cToGo client application provides three different views including Explorer View, Navigation View, and Operation View.

CONCEPTUAL DESIGN AND BASIC FEATURES

Web2cHMI is a set of JavaScript (JS) classes to be used in Web client applications executed by a Web browser. A Web2cHMI-compliant browser must implement the Web Sockets API and the getUserMedia / Stream API (required for speech recognition only). Table 1 summarizes the implementation status [2] by major Web browsers (most recent desktop or mobile version).

The recognized user input from each supported modality is handled by its own modality-specific class notifying a common interface class which translates the modality-specific user actions into common unique application-specific commands for further execution.

Table 1: Implementation Status of Required JS API

Web Browser	Web Sockets API	getUserMedia / Stream API
Edge	yes	yes
Firefox	yes	yes
Chrome	yes	yes
Safari	yes	no
Opera	no	yes
iOS Safari	yes	no
Opera Mini	no	no
Android Browser	yes	yes
Chrome for Android	yes	yes

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Supported Modalities

Web2cHMI supports five different modalities which can be used simultaneously:

- 1D/2D flat gestures including single-finger actions (**mouse**) and single- or multi-finger gestures (**touch-sensitive display**)
- 2D/3D spatial gestures including hand-gestures (**LEAP Motion controller** [3]) and hand- or arm-gestures (**Myo gesture control armband** [4])
- English Spoken commands (**Sphinx speech recognition** [5]).

Table 2 summarizes the support status of the various modalities on major operating systems (most recent version).

Table 2: Support Status of Web2cHMI Modalities

OS	Mouse	Touch	LEAP	Myo	Speech
Win	yes	yes	yes	yes	yes
Linux	yes	yes	yes	no	yes
MacOS	yes	yes	yes	yes	yes
Android	no	yes	no	yes	yes
iOS	no	yes	no	yes	yes

Web2cHMI recognizes various primitive, i.e. native or input device-specific gestures including

- Mouse: Click, Move
- Touch-sensitive display: Tap, Move / Swipe, Pinch (two fingers)
- LEAP Motion controller: Key-Tap, Swipe, Open-Hand, Closed-Hand, Circle
- Myo gesture control armband: Double-Tap, Wave-Out / Wave-In, Fingers-Spread, Fist

In addition, enriched gestures, i.e. primitive gestures followed by moves, rotations etc. are supported. If applicable or required by ergonomics principles, different gestures may be applied by right or left handed individuals.

Spatial Gestures

The LEAP Motion controller (Figure 1) and the Myo gesture control armband (Figure 2) are connected locally through USB and Bluetooth, respectively. The raw sensor signals are processed by a local Web server which communicates with the Web2cHMI via Web Sockets.

Unlike mice or touch-sensitive displays gesture recognition devices such as LEAP or Myo do not allow an accurate positioning of a cursor.

The LEAP Motion controller can also be used in upside-down orientation.

To avoid unwanted responses to unintentional gestures recognition ability must be armed or disarmed explicitly by the user. Arming and disarming the LEAP Motion controller is performed by the Key-Tap gesture. While armed the application displays a moving label (yellow = finger is not in the sensor's active range, green = finger is in the sensor's active range) indicating the position of the pointing finger within a virtual frame (width = 200 mm, height = 200 mm) located 100 mm above the sensor. The active sensor range covers ± 100 mm before and behind the sensor, respectively.

Figure 1: LEAP Motion sensor

In order to arm and disarm the Myo gesture control armband the user has to execute a Double-Tap gesture. While armed the application displays a moving green label indicating the position of the hand of the arm wearing the sensor within a virtual frame (width = 500 mm, height = 500 mm). During arming the arm direction is taken as the zero-reference and the label position is set arbitrarily to the center of the application window.

If a gesture is successfully recognized the next gesture recognition is inhibited for approximately 1s while the label is fixed to the center of the application window.

Figure 2: Myo gesture control armband

Spoken Commands

Phonetic analysis of locally recorded audio sequences is performed by the Web2cToGo server application [6] which notifies the Web2cHMI via Web Socket communication if a spoken command has been recognized. In order to preserve privacy audio recording can be paused or resumed by clicking the speech recognition icon at the left border of the Web2cToGo application window. To avoid unwanted responses to unintentional spoken commands the user must say "Ok" or "Sleep" to arm or disarm the ability to recognize spoken commands, respectively.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Figure 3: Web2cToGo Explorer View

EXPLORER VIEW HMI

The Web2cToGo Explorer View (Figure 3) displays a set of icons to select a user application. It provides launching or displaying of already launched but currently hidden user applications. In addition it indicates the currently available modalities and the corresponding arming status (L = LEAP, M = Myo and S = speech recognition).

Amongst other user input Web2cHMI recognizes for instance the following corresponding actions to launch or display the selected application:

- Mouse: **Click** selected application icon
- Touch-sensitive display: **Tap** selected application icon (right or left hand)
- LEAP Motion Controller: **Open-Hand** (right or left hand)
- Myo gesture control armband: **Fingers-Spread** (right or left arm)
- Speech Recognition: **Say** "Display"

NAVIGATION VIEW HMI

The Web2cToGo Navigation View (Figure 4) previews the selected user application. It allows switching between user applications, switching to Explorer or Operation View, closing the current user application, browsing between application pages, and zooming, resizing or scrolling pages of user applications.

Among other user input Web2cHMI recognizes for instance the following corresponding actions to zoom in the current application page:

- Mouse: **Click** magnifier icon
- Touch-sensitive display: **Tap** magnifier icon (right or left hand) or Right/Left-Tilted Open-**Pinch** (right/left hand)
- LEAP Motion Controller: **Open-Hand** & Clockwise Rotation (right or left hand)
- Myo gesture control armband: **Fingers-Spread** & Clockwise rotation (right or left arm)
- Speech Recognition: **Say** "Zoom In"

Figure 4: Web2cToGo Navigation View

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

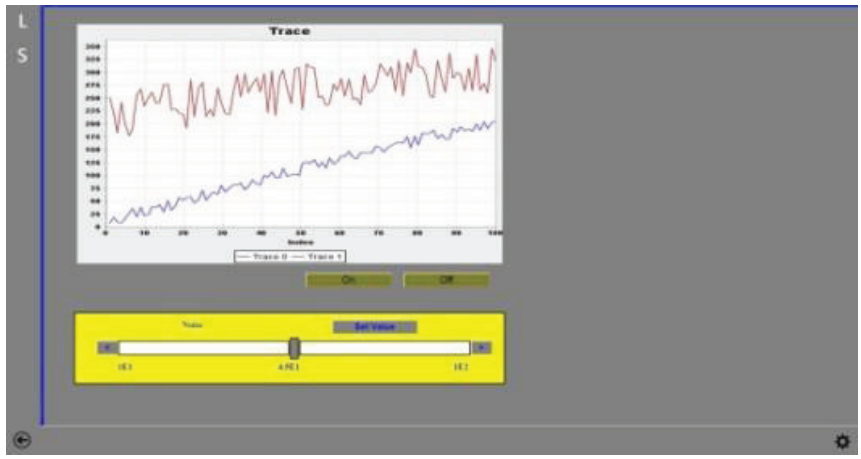


Figure 5: Web2cViewer Operation View

OPERATION VIEW HMI

While switched to Operation View (Figure 5) the Web2cToGo client application displays a Web2cToolkit-compliant Web application. The Web2cViewer application implements Web2cHMI. Each Web2cViewer application page might contain one widget instance of each of the following interactive widget types, only. According to its type, an interactive widget is capable of performing a specific, predefined user action such as opening a vacuum valve or changing a set value of a power supply:

- On-type Button (user action = "On")
- Off-type Button (user action = "Off")
- Slider (user action = "Set Value")
- Chart (user action = "Zoom Data")

In addition a page might contain an unlimited number of passive Web2cViewer widgets such as labels or value fields.

Among other user input Web2cHMI recognizes for instance the following corresponding actions to increase a set value of an attached controls device in small steps using the slider widget:

- Mouse: **Click** ">"-button and **Click** "Set Value"-button of the slider widget
- Touch-sensitive display: **Tap** ">"-button and **Tap** "Set Value"-button of the slider widget (right or left hand)
- LEAP Motion Controller: Clockwise **Circle** (right or left hand)
- Myo gesture control armband: **Fist** & Clockwise Rotation (right or left arm)
- Speech Recognition: **Say** "More"

PROJECT STATUS AND OUTLOOK

While being fully implemented by Web2cToGo and Web2cViewer Web2cHMI is still experimental. An implementation by Web2cArchiveViewer is envisaged. In general the performance and reliability of gesture and speech recognition has to be improved. The defined gestures or spoken commands will likely be changed in future based on usability experiences gained in a real accelerator environment.

REFERENCES

[1] Web2cToolkit; http://web2ctoolkit.desy.de
[2] http://www.caniuse.com
[3] LEAP; https://www.LEAPmotion.com
[4] Myo; https://www.myo.com
[5] Sphinx-4; http://cmusphinx.sourceforge.net/sphinx
[6] R. Bacher, "Renovating and Upgrading the Web2cToolkit Suite: A Status Report", PCaPAC'14, Karlsruhe, Germany, October 2014, FCO2012, p. 234 (2014); http://www.JACoW.org

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A GRAPHICAL TOOL FOR VIEWING AND INTERACTING WITH A CONTROL SYSTEM

J. Forsberg, V. Hardion, D. Spruce
MAX IV Laboratory, Sweden

Abstract

We present a graphical interface for displaying status information and enabling user interaction with the Tango based control system for the MAX IV synchrotron. It focuses on bringing an intuitive view of the whole system, so that operators can quickly access the controls for any hardware based on its physical location.

The view is structured into different layers that can be selectively shown, and various live updated information can be displayed in the form of e.g. color or text. Panning and zooming is supported, as well as invoking commands. The interface is defined by an SVG (Scalable Vector Graphics) drawing which can be edited without programming expertise.

Since our system is based on modern web technologies, it can be run as a web service accessible by standard browsers, but it can also be integrated in GUI applications.

INTRODUCTION

The MAX IV laboratory is a synchrotron facility under construction in Lund, Sweden. It has a TANGO [1] based control system consisting of thousands of different pieces of software that provide users with access to equipment. The control system tends to be structured in a way that makes sense to the software developers that build it, but not necessarily to accellerator physicists and synchrotron operators. This project, called "SVG synoptic", is an attempt at making a control system interface that is intuitive to the users.

The basic idea is to display live control system information in the form of a picture that somehow represents the physical layout of the machine, in a schematic way. Items in the picture may contain live information about corresponding control system variables, as e.g. color or text. The user is able to iteract with the drawing in order to bring up specific controls in a window, or to perform specific actions.

It may not be realistic to fit every detail of a complex system into a single image. By allowing the synoptic to contain several different detail levels, we can provide a seamless interface where the user can choose what to see simply by zooming and panning using the mouse.

A control system is usually logically split into several separate sub-systems such as vacuum, cooling, and so on, and for many use cases only some of these may be of interest to the user. SVG synoptic provides a way to structure the information into layers that can be toggled on or off depending on what subsystems are relevant at the time.

The size of the control system has to be taken into account from a performance point of view. Communicating with thousands of items is time consuming, and since most of the information is not visible at any given time it is not helpful. Therefore, SVG synoptic has the ability to only keep updated about the parts currently visible to the user.

CHOICE OF TECHNOLOGY

The development of this project was initially motivated by requirements from the users that were not easily achievable using available tools. The choice was between extending an existing system or creating a new one.

SVG

The standard tool for displaying synoptics in the TANGO community is based on a specialized Java based drawing tool called JDraw (part of the ATK [2] suite of tools). It has some of the features of SVG-synoptic, but the current viewers can only provide a static view. The JDraw format, while simple, is also fairly limited. Since it is not a widely used standard, there is not much tooling to work with the file format.

SVG (Scalable Vector Graphics) [3] has the following characteristics:

- Being a vector format (like JDraw) it produces resolution independent images suitable for free zooming.
- It is a mature, widely used and open standard, adopted by web browsers (currently all major browsers except IE 8 or older) and many graphics software packages.
- The file format is XML and therefore straightforward to generate and manipulate programmatically.
- SVG can easily be accessed from javascript since it becomes part of the DOM (Document Object Model) once loaded. It can also be styled through CSS (Cascading Style Sheets).
- The FOSS (Free and Open Source Software) drawing application Inkscape [4] uses SVG as a native format.

Considering these features, we chose to go with SVG instead of extending JDraw.

JavaScript

Part of the reasons SVG was chosen as an image format was that it enables the use of a web browser as viewer, even if embedded in an application. This cut down development time, but it did mean that the user interaction had to be implemented in JavaScript [5], the "native" programming language supported by web browsers.

JavaScript has a large community around it, resulting in a lot of high quality libraries. In particular, for interacting

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

with SVG, D3.js [6] is very powerful. Although mostly used for generating data visualisations, it can also be used to manipulate an existing SVG. It is used for implementing the interactive features of the synoptic.

An added benefit of having most of the application running inside a web browser is that it is straightforward to implement it as a web service.

Python

The Python [7] programming language was an obvious choice for the "back-end" part of the application, since it is already the main language used for control system work at MAX IV. Through the PyTango [8] bindings it can access the control system at the low level, and the Taurus library provides Qt4 [9] widgets and other functionality on top.

Python also has good representation in web frameworks, which means that it's also a reasonable choice considering future developments.

ARCHITECTURE

The result of this project is a library, "svgsynoptic", that can be used to easily build visualisations as described above. The implementation is as a Qt widget communicating with the underlying control system and embedding a WebKit component to render the view.

The QWebView widget in Qt makes it possible to embed a full WebKit browser environment in a Qt widget, and the PyQt bindings allow intercommunication between Python and JavaScript.

In the following section, the general architecture of the library is laid out. A few concepts will be used in this description:

model Items in the drawing must be connected to control system entities. For this we use *models*. A model is simply a string, representing some control system aspect in a standard way. For TANGO, a model may currently be the name of a device (e.g. `"sys/tg_test/1"`), or a device attribute (e.g. `"sys/tg_test/1/ampli"`). It's likely to be more generalised in the future. A drawing item may have several models.

section A section is an item in the SVG that is not necessarily directly represented in the control system, such as a logical part of a machine. The idea is to allow the user to navigate the synoptic by e.g. clicking a section in order to move the view to that part.

layer The drawing may be structured into different **layers**, each representing some global grouping of control system items, e.g. "Vacuum" or "Magnet". The user is able to toggle each of these layers on or off at any time, in order to restrict the visible information to what's relevant at the moment.

zoom level If needed, the drawing may be further structured into an arbitrary number of **zoom levels** which successively present more detailed views of the system. The synoptic will automatically switch between these levels as the user changes the scale, displaying only the current one.

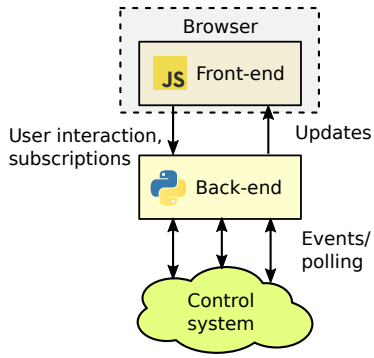


Figure 1: Architecture overview.

In order to separate concerns, the system is composed of two major pieces (Fig. 1):

- a "back-end" that communicates directly with the control system
- a "front-end" which presents a user interface to the user.

We will be discussing TANGO specific details as an example. The current implementation runs both of these parts in the same process.

Back-end

The back-end handles all communication with the control system. The synoptic itself is only concerned with presenting information about the underlying system, relying on opening panels or external applications for direct user interaction such as writing attributes or running commands. Therefore, the information flow is essentially one-way, from control system to back-end.

The back-end receives a list of models currently visible in the synoptic from the front-end (see below). This list is intended to be used to keep the front-end updated about changes in the control system. The TANGO back-end dynamically sets up change event subscriptions to these models, pushing updates to the front-end. It also unsubscribes to attributes no longer in view.

The back-end received notification of user interactions with the front-end, e.g. mouse clicks and hover. It can take appropriate actions, such as visually selecting a clicked item, opening an application when right-clicking certain models, or updating the tooltip with relevant information about the item hovered over, using the front-end's API if needed. The TANGO implementation reacts to right mouse button clicks on models corresponding to devices, by bringing up an appropriate window for direct interaction.

For the web-based implementation, a small part of the back-end will run in the browser, to handle communication across HTTP, using SSE (server-sent events) to push updates.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Front-end

The front-end (Fig. 2) is written in JavaScript and runs entirely in a web browser, usually embedded as a Qt widget. It is concerned with loading the SVG and presenting it to the user in an interactive way, reporting control system related actions to the back-end. The front-end is not directly aware of the control system and can only associate model names with items. It relies on the backend knowing what to do with the models. It has a simple API which can be used from the back-end, e.g. to inform the user about changes in the control system state of a particular model.

The front-end needs to be configured by the developer. At least an SVG image must be supplied, using the correct structure.

The UI can be extended with various optional "plugins" that add functionality. There are currently plugins for adding information "popups", a "thumbnail" view, and buttons to toggle the visibility of each layer.

Embedding

Because of its implementation as a Qt widget, it is easy to use the synoptic as part of another application. For example, it has been used as part of a specialised vacuum application (Fig. 3).

CURRENT STATUS

Applications based on the library is in use at MAX IV since around one year, for the linear accellerator, the 3 GeV storage ring and several beamlines. It has gone through several revisions in order to handle the demands of a growing control system.

Currently the storage ring synoptic contains roughly 2000 magnets, 1000 sensors, 200 beam position monitors, 85 ion pumps and 80 vacuum valves. The SVG for this synoptic is too large and complex to be drawn manually, so it is created by a script taking equipment lists as input.

The current focus of development is making the library a stable and flexible tool. The project source code is available at `https://github.com/maxiv-kitscontrols/lib-maxiv-svgsynoptic`

CONCLUSION

We have provided a library that allows creating interactive visualisations of a control system by drawing an SVG file and inserting references to control system entities. The system has been successfully used during the commissioning of the MAX IV synchrotron facility and several beamlines.

ACKNOWLEDGMENT

The authors wish to acknowledge the entire team at MAX IV, including controls and software, IT, accellerator group and operators for invaluable feedback.

REFERENCES

[1] The TANGO Control system website: `http://www.tango-controls.org`

[2] F. Poncet, J.L. Pons, 10th ICALEPCS Int. Conf. on Accelerator, & Large Expt. Physics Control Systems. Geneva, 10 - 14 Oct 2005, FR2.1-6O (2005).

[3] W3C, *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*, `http://www.w3.org/TR/SVG/`

[4] Inkscape website: `https://inkscape.org/en/`

[5] Ecma International, *ECMAScript 2015 Language Specification*, `http://www.ecma-international.org/publications/standards/Ecma-262.htm`

[6] D3.js website: `http://d3js.org/`

[7] Python Software Foundation. *Python Language Reference, version 2.7*, Available at `http://www.python.org`

[8] PyTango website: `http://www.esrf.eu/computing/cs/tango/tango_doc/kernel_doc/pytango/latest/index.html`

[9] Qt website: `http://www.qt.io`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

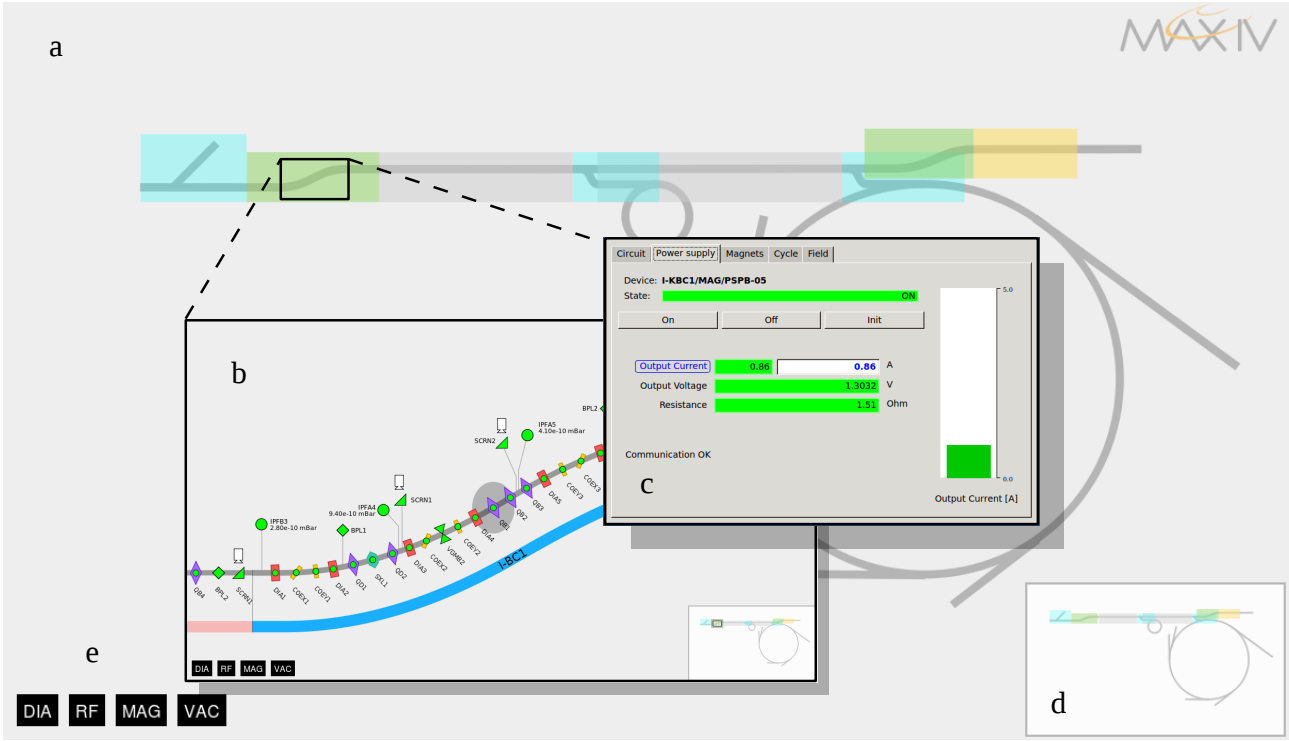


Figure 2: Synoptic for the linear accellerator of MAX IV. Initially an overview of the entire machine is shown (a). The user can zoom in on a particular region (b) to reveal individual equipment such as magnets. To access a particular device, a specific GUI (c) can be opened. A "thumbnail" (d) shows the current view in relation to the whole. A row of buttons (e) allow the user to control the visibility of individual subsystems.

Figure 3: An example of embedding the SVG synoptic widget in a dedicated vacuum monitoring application. The widget responds to user actions such as selecting a device in the tree to the left, by displaying the chosen device, and vice versa.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

HOW CASSANDRA IMPROVES PERFORMANCES AND AVAILABILITY OF HDB++ TANGO ARCHIVING SYSTEM

R. Bourtembourg, J.L. Pons, P. Verdier, ESRF, Grenoble, France

Abstract

The TANGO release 8 led to several enhancements, including the adoption of the ZeroMQ library for faster and lightweight event-driven communication. Exploiting these improved capabilities, a high performance, event-driven archiving system, named Tango HDB++, has been developed. Its design gives the possibility to store archiving data into Apache Cassandra: a high performance scalable NoSQL distributed database, providing High Availability service and replication, with no single point of failure. HDB++ with Cassandra will open up new perspectives for TANGO in the era of big data and will be the starting point of new big data analytics/data mining applications, breaking the limits of the archiving systems which are based on traditional relational databases. The paper describes the current state of the implementation and our experience with Apache Cassandra in the scope of the Tango HDB++ project. It also gives some performance figures and use cases where using Cassandra with Tango HDB++ is a good fit.

INTRODUCTION

The TANGO archiving system is a tool allowing TANGO users to store the readings coming from a TANGO based control system into a database. The archived data are essential for the day by day operation of complex scientific facilities. They can be used for long term monitoring of subsystems, statistics, parameters correlation or comparison of operating setups over time.

To take advantage of the fast and lightweight event-driven communication provided by TANGO release 8 [1] with the adoption of ZeroMQ [2], a novel archiving system for the TANGO Controls framework [3], named HDB++ [4], has been designed and developed, resulting from a collaboration between Elettra and ESRF.

HDB++ design allows TANGO users to store data with microsecond timestamp resolution into traditional database management systems such as MySQL [5] or into NoSQL databases such as Apache Cassandra [6].

APACHE CASSANDRA

Apache Cassandra is an open source distributed database management system available under the Apache 2.0 license. Cassandra's masterless ring architecture where all nodes play an identical role, is capable of offering true continuous availability with no single point of failure, fast linear scale performance and native multi-datacentre replication. All these advantages over the traditional relational databases made Cassandra a very good candidate to store the historical data coming from the ESRF TANGO control systems.

HDB++ DESIGN

HDB++ is a novel TANGO Controls archiving system. Its architecture is mainly based on the two following Device Servers and takes advantage of the TANGO events mechanism:

- The Event Subscriber Device Server subscribes to TANGO archive events, which are ZeroMQ events in the latest TANGO releases, and stores the received events in the historical database. It provides diagnostics data as well.
- The Configuration Manager Device Server configures the attributes to be archived and defines which Event Subscriber is responsible for a set of TANGO attributes to be archived. It provides diagnostics data as well.

An abstraction library named *libhdb++* decouples the interface to the database back-end from the implementation.

To be able to store data into Apache Cassandra, a C++ library named *libhdb++cassandra*, implementing the methods from the *libhdb++* abstract layer for Cassandra, has been developed at the ESRF. It was inspired by the work done by Elettra who implemented the HDB++ MySQL back-end support shared library. The HDB++ Device Servers design is shown in Fig. 1.

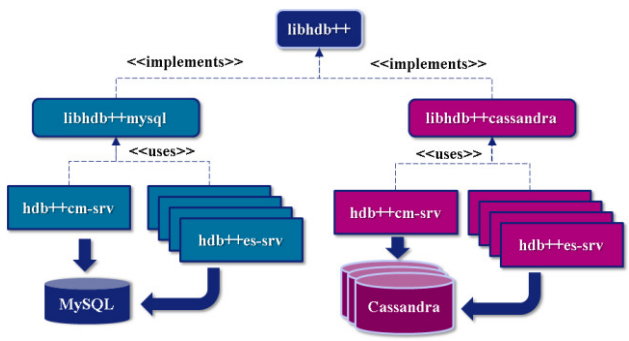


Figure 1: HDB++ Device Servers design.

The Cassandra HDB++ libraries allow us to reuse the Event Subscriber and Configuration Manager Device Servers as well as the Graphical User Interfaces (GUI) interacting with these Device Servers (HDB++ Configurator GUI shown on Fig. 2 and HDB++ Diagnostics GUI), without changing their source code. Linking the two Device Servers with *libhdb++cassandra* was sufficient to be able to store data into Cassandra.

For the data extraction part, new Java classes were developed for HDB++ Cassandra, extending the features provided by an abstract layer hiding the database back-end to the Java applications.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

A Java GUI named HDB++ Viewer (Fig. 3) was developed, using these Java classes, to visualize the data stored in the historical database.

The GUI can be configured to read data from HDB++ Cassandra, from HDB++ MySQL or from the ESRF Oracle old Historical Database.

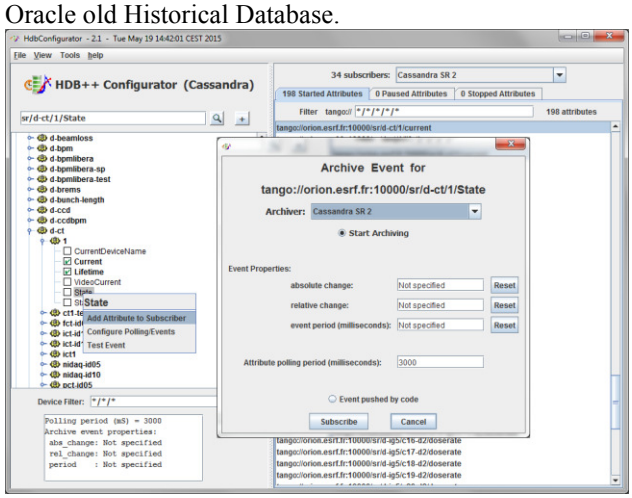


Figure 2: HDB++ Configurator GUI.

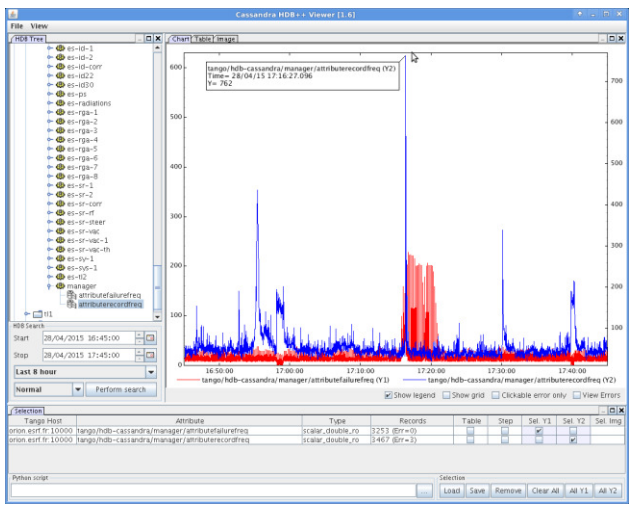


Figure 3: HDB++ Viewer GUI showing the biggest events peak (762 events/s) observed in 2015.

The HDB++ Cassandra Java classes are retrieving the data directly from the Cassandra cluster and are written in a way to take advantage of the replication. A data request is split into several smaller requests to ensure each query targets one and only one Cassandra partition to maximize performances. The requests are sent asynchronously and in parallel, taking advantage of the fact that the data is distributed over several Cassandra nodes. The data model has been designed in such way that there is one partition per attribute per period of time. This period of time is adjustable in the Event Subscriber Device Server code and is currently set to one day. In the future, this period will be set to one hour, meaning one partition will contain one hour of data of a specific attribute.

A C++ extraction library for HDB++ Cassandra will be implemented soon, inheriting from the C++ HDB++ abstract extraction library already developed by Elettra.

THE CASSANDRA EXPERIENCE

The Cassandra basics were learnt very quickly thanks to the numerous tutorial online videos and to the Datastax academy web site [7].

Since the Event Subscriber and Configuration Manager Device Servers were written in C++, the *libhdb++cassandra* library described above had to be implemented in C++. To do this, it was necessary to use the Cassandra C++ driver, which was still in Beta version at that time. The Cassandra C++ driver appeared to be stable for our usage. The version 2.0.1 of the driver, which provides useful features like latency-aware routing, is currently in use. Newer versions, which will be used soon, are providing retry policies, making the code simpler and more robust to failures.

After a few tests archiving a few attributes, it was decided early in the project to store real data coming from the accelerator control system. So the system was configured to archive all the accelerator control system TANGO attributes which were already being archived in the old HDB ESRF database. The HDB++ Configuration GUI (Fig. 2), really helped to simplify this task.

Following the Apache Cassandra website advices, the project was started with Cassandra 2.1.0, with 3 recycled servers (See Table 1). The problem is that useful data was stored almost from the start of the project and Cassandra 2.1.x branch appeared to be a development branch with some remaining bugs so an upgrade was necessary each time a new version was released until version 2.1.7. The bugs didn't cause any loss of data. The most annoying bug was CASSANDRA-8321 [8] which was filling up the disks with temporary files not removed after use. A cron job had to be created in order to remove the oldest of these temporary files until a new version fixing the bug was released. The upgrade was always smooth with no downtime of the HDB++ system. To upgrade a Cassandra cluster, one needs to do a rolling upgrade, meaning each node is upgraded one after the other so there is always only one node down at a given moment. Missing data is then streamed by the other nodes once the node is back online thanks to Cassandra consistency repair features [9].

Table 1: Recycled Servers' Specifications

Item	Description
CPU	Intel Xeon E5440, 2.83GHz, 4 cores, 2 CPUs
Memory	48 GB
OS	Debian 7 (Wheezy) - 64 bits
Storage 1 (OS)	SAS 10K RPM 2x72 GB – RAID 1
Storage 2 (Data)	SAS 10K RPM 2x146 GB - RAID 0 SAS 10K RPM 1x146 GB
Network	1 Gb Ethernet x 1 port

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Data Model

The schema and tables names were inspired by the HDB++ MySQL tables.

Cassandra provides very high performances but in order to maintain these performances, some operations which are allowed in traditional relational databases are forbidden in Cassandra, like the JOIN queries for instance or SELECT queries with a WHERE clause not containing the partition key. For this reason, it was not possible to reuse directly the same database schema as the one created for the HDB++ MySQL version and some additional tables had to be created to overcome this difficulty. Cassandra Query Language collection types were also used to store the Tango array types.

Partition keys

The partition key is a part of the primary key which will be used by Cassandra to decide on what nodes data having a primary key containing this partition key will be stored. It was decided to use the HDB++ attribute configuration ID and a text field named *period* as partition key for all the tables storing historical values. At the beginning, the name of the current month was used for this *period* field (e.g. "2015-10"). The partition sizes were too big and this was degrading the performances, causing long garbage collector pauses when users were requesting a full month of data for an attribute, making a Cassandra node unresponsive during the garbage collection process period, which could last for more than one minute sometimes. It was decided to insert the current day in the *period* field to reduce the partitions sizes. This improved the performances but is still not perfect. This is why one partition per attribute and per hour will be created in the future (and the already stored data will be migrated) to keep the partition sizes below 100MB, following the Cassandra experts' recommendations.

Cassandra Administration

At the ESRF, Datastax OpsCenter Community Edition, shown in Fig. 4, is used to monitor the Cassandra cluster. OpsCenter is a web-based visual monitoring tool for Apache Cassandra.

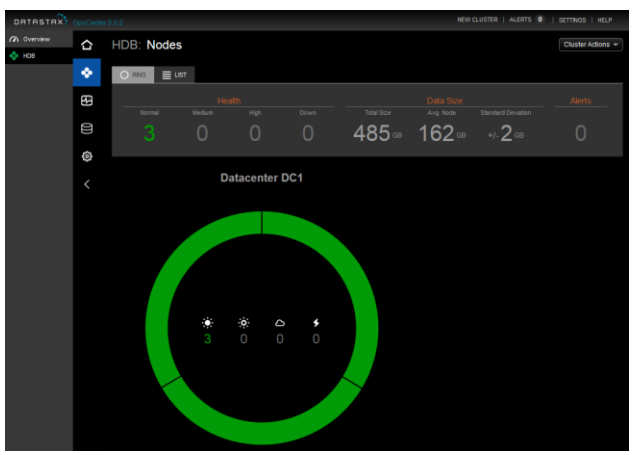


Figure 4: Datastax OpsCenter monitoring tool showing HDB++ Cassandra cluster status at the ESRF.

CURRENT STATUS

The HDB++ Cassandra version at the ESRF is currently storing data coming from about 7440 TANGO attributes managed by 34 Event Subscriber Device Server instances. The maximum number of attributes managed by a single Event Subscriber at the ESRF is currently 940.

The Figure 5 is showing the attributes distribution among the Event Subscriber instances as well as the number of received events per event subscriber during a period of about 1 day 1 hour and 20 minutes, on 12 -13 October 2015. The maximum number of events received for a single event subscriber during that period was a bit more than 1.5 million on that typical day.

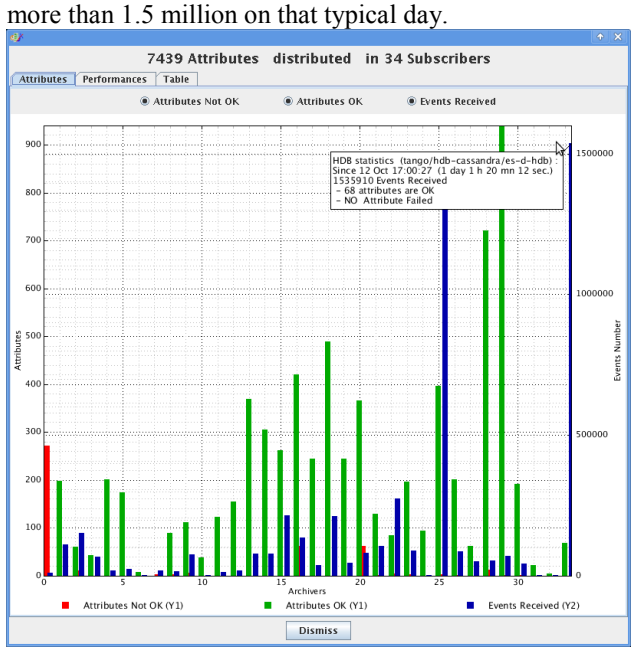


Figure 5: Distribution window from the HDB++ diagnostics GUI.

The Cassandra cluster is currently composed of 3 nodes running Cassandra 2.1.7. The average data size per node is about 162GB. The recycled servers described in Table 1 are still used for the moment.

The architecture of the HDB++ Cassandra system at the ESRF is described in Fig. 6 as well as the future evolution of the cluster.

FUTURE PLAN

New Nodes

3 new servers have been bought and will be replacing soon the 3 current nodes after having upgraded Cassandra to the latest stable version (probably 2.1.10).

Analytics Datacentre

3 additional servers with SSD disks have been bought to constitute a new datacentre dedicated to analytics. They will be located in a different room than the production datacentre for a better protection of the data, in case of fire for instance. The specifications of all these new servers are described in Table 2.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

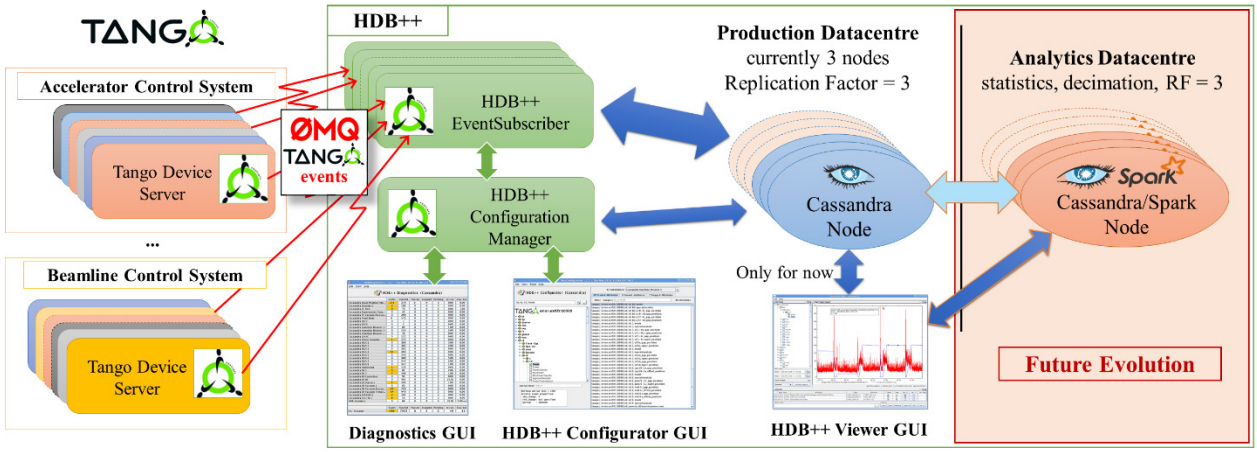


Figure 6: HDB++ Cassandra system architecture at the ESRF and future evolution.

Table 2: New Servers' Specifications

Item	Production DC	Analytics DC
CPU	Intel Xeon E5-2630, 2.4GHz, 8 cores	Intel Xeon E5-2630, 2.4GHz, 8 cores, 2 CPUs
Memory	64 GB	128 GB
OS	Debian 7 (Wheezy) - 64 bits	
Storage 1 (OS)	SAS 15K RPM 300GB x2 – RAID 1	
Storage 2 (data directories)	1x 1TB, SATA, 7200 RPM 1x 2TB NL-SAS 7200 RPM	SSD SATA MLC 800GBx4
Network	1 Gb Ethernet x 4 ports	

Apache Spark [10] will be installed on the *Analytics* datacentre Cassandra nodes. It will be used to compute statistics (minimum, maximum, average, received events count, errors count) per attribute and will compute decimation data which will be inserted into new Cassandra tables from a new keyspace dedicated to analytics data.

The HDB++ Viewer tool will be improved in order to return decimated data and statistics first when a user will want to retrieve a big amount of data. That way, the user will get quickly an overview of the data and might be able to narrow down the region of interest by zooming in and reducing the time interval. At some point, once the amount of data to retrieve will be below a reasonable number (to be defined), the user will see the real data.

CONCLUSION

HDB++ Cassandra proved it can replace the old ESRF Historical Database. It demonstrated it is easy to develop additional database back-end support for HDB++.

It might also become a good alternative to the current TANGO Temporary Database (TDB).

Apache Cassandra proved it is able to deliver continuous availability as long as the partitions sizes stay small to avoid long garbage collector cycles.

Using Apache Cassandra as HDB++ database back-end is a good fit for projects requiring big storage capacities and continuous availability, as well as for projects where more storage capacities and performances might be needed in the future, like at the ESRF for instance with the coming upgrade of the accelerator. For smaller projects, the HDB++ MySQL version is enough and cheaper because less servers are needed.

The analytics cluster using Apache Spark should allow to bring better user experience and read performances thanks to the new decimation tables and SSD disks.

ACKNOWLEDGMENT

The Elettra HDB++ team for their work on the HDB++ project and for their open mind.

DuyHai Doan, technical advocate at Datastax, for his advices and Cassandra expertise.

REFERENCES

[1] A. Götz et al., "TANGO V8 – Another turbo charged major release", ICALEPCS'13, San Francisco, USA (2013), http://jacow.org
[2] ZeroMQ: http://zeromq.org
[3] TANGO Controls: http://www.tango-controls.org
[4] L. Pivetta et al., "HDB++: A New Archiving System for TANGO", *These Proceedings,* WED3O04, ICALEPCS'15, Melbourne, Australia (2015).
[5] MySQL: http://dev.mysql.com
[6] Apache Cassandra: http://cassandra.apache.org
[7] Datastax Academy: https://academy.datastax.com
[8] CASSANDRA-8321 bug: https://issues.apache.org/jira/browse/ CASSANDRA-8321
[9] Cassandra consistency repair features: http://docs.datastax.com/en/cassandra/2.1/cassandra/dml/dmlConsistencyRepair.html
[10] Apache Spark: https://spark.apache.org

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE INSTRUMENT CONTROL ELECTRONICS OF THE ESPRESSO SPECTROGRAPH @VLT

V. Baldini, G. Calderone, R. Cirami, I. Coretti, S. Cristiani, P. Di Marcantonio, P. Santin, INAF-OATs, Trieste, Italy
D. Mégevand, OG, Versoix, Switzerland,
F. Zerbi, INAF-OAB, Merate, Italy

Abstract

ESPRESSO, the Echelle SPectrograph for Rocky Exoplanet and Stable Spectroscopic Observations, is a super-stable Optical High Resolution Spectrograph for the Combined Coudé focus of the Very Large Telescope (VLT). It can be operated either as a single telescope instrument or as a multi-telescope facility, by collecting the light of up to four Unit Telescopes (UTs). From the Nasmyth focus of each UT the light is fed, through a set of optical elements (Coudé Train - CT), to the Front End Unit (FEU) which performs several functions, as image and pupil stabilization, inclusion of calibration light and refocusing. The light is then conveyed into the spectrograph fibers.

The whole process is handled by several electronically controlled devices. About 40 motorized stages, more than 90 sensors and several calibration lamps are controlled by the Instrument Control Electronics (ICE) and Software (ICS). The technology employed for the control of the ESPRESSO subsystems is PLC-based, with a distributed layout close to the functions to control. This paper illustrates the current status of the ESPRESSO ICE, showing the control architecture, the electrical cabinet's organization and the experiences gained during the development and assembly phase.

INTRODUCTION

ESPRESSO is the Echelle SPectrograph for Rocky Exoplanet and Stable Spectroscopic Observations that will be installed in the Combined Coudé Laboratory (CCL) of the ESO VLT. The main goals of ESPRESSO will be the measurement of high precision radial velocities of solar type stars for search for rocky planets, the measurement of the variation of the physical constants and the analysis of the chemical composition of stars in the nearby galaxies [1]. To reach these goals an extremely stable, accurate, efficient and high resolution spectrograph is required.

ESPRESSO will be able to operate either as a single telescope instrument or as a multi-telescope facility, collecting the light of up to four telescopes. In fact the light of each UT is fed from the Nasmyth focus, through a tunnel made up of optical elements (Coudé Train), to a Front End Unit (FEU) in the CCL of ESO VLT.

The ESPRESSO Instrument Control Electronics aims to control all moving parts that allow the light to follow that path and the lamps that calibrates the final spectrum, assuring also the safety of the instruments and people through a large number of sensors and alarms.

SYSTEM ARCHITECTURE

Each of the FEU four arms (see Figure 1) is composed by an Atmospheric Dispersion Corrector (ADC), a focus translational stage and a system that performs the field and pupil stabilization. The latter is composed by a set of piezo tip-tilt stages to modify the light path, two technical CCD (TCCD) and a neutral filter. The FEU provides, through a calibration slide, the means to inject the calibration light (white and spectral sources) into the spectrograph fiber when needed.

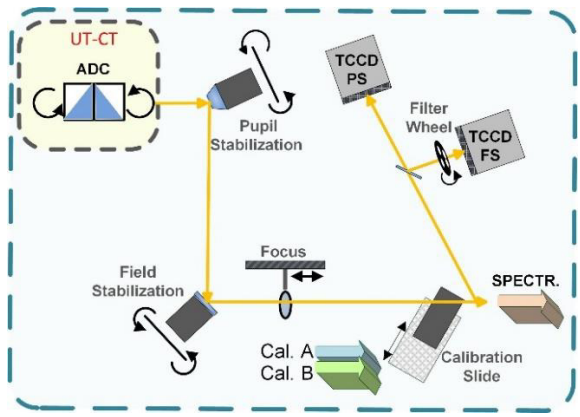


Figure 1: FEU functions in each of the four arms.

Moreover, the FEU provides a mode selector (see Figure 2) mounted on a rotary stage whose task is to feed the spectrograph fibers with the light coming from the selected telescopes and passing through the corresponding FEU arms in one of the specific instrument mode (single telescope High Resolution, single telescope Ultra High Resolution or multi telescope Medium Resolution).

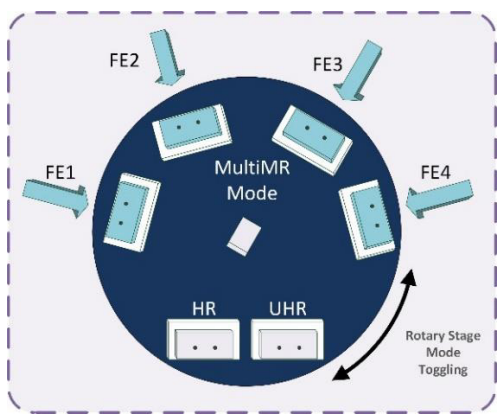


Figure 2: ESPRESSO mode selector.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Two fibers feed the spectrograph simultaneously: the target fiber and the sky/calibration fiber.

Finally, two ESO Next Generation Controllers (NGCs) control two scientific detectors: one for the red arm and one for the blue one.

ESPRESSO INSTRUMENT CONTROL SYSTEM

The control electronics of an instrument like ESPRESSO must satisfy a large number of requirements. It must be reliable, real time capable, accurate in motor positioning and must be easy to maintain on the long term. ESPRESSO ICE aims to satisfy these requirements using mainly COTS components and Beckhoff PLCs as control system [2].

In ESPRESSO the workload is divided between two Beckhoff CPUs (see Figure 3 and Figure 4). The first CPU purpose is to control all the FEU related functions while the second one manages the Thermal Control Systems, the Calibration Unit functions and the interface with the NGCs.

The Beckhoff CPUs used for ESPRESSO belong to the CX2030 series. An EK1100 bus coupler allows the EtherCAT fieldbus communication continuity among the different electronic sub-racks. The functional modules used for I/O and other tasks required by the instrument can be classified in four types:

- Digital I/O – for devices with discrete control signals and status output.
- Analog I/O – for devices with continuous control signals and status output and measurement devices.
- Communication Modules – for devices with high level interface (e.g. serial interface).
- Motion control modules – for the motorized functions.

ESPRESSO has 8 motorized functions for the ADCs, 14 stages in the FEU, and 8 in the calibration unit. All the motorized stages were chosen from the MICOS PI supplier and have a DC motor.

The Beckhoff modules selected for building the basic motion control block are:

- EL7342 - 2 channel DC motor output stage 50 V DC, 3.5 A.
- EL5101 - Incremental Encoder Interface.
- EL1084 - 4 Channel input, 24 VDC, switching to negative potential.

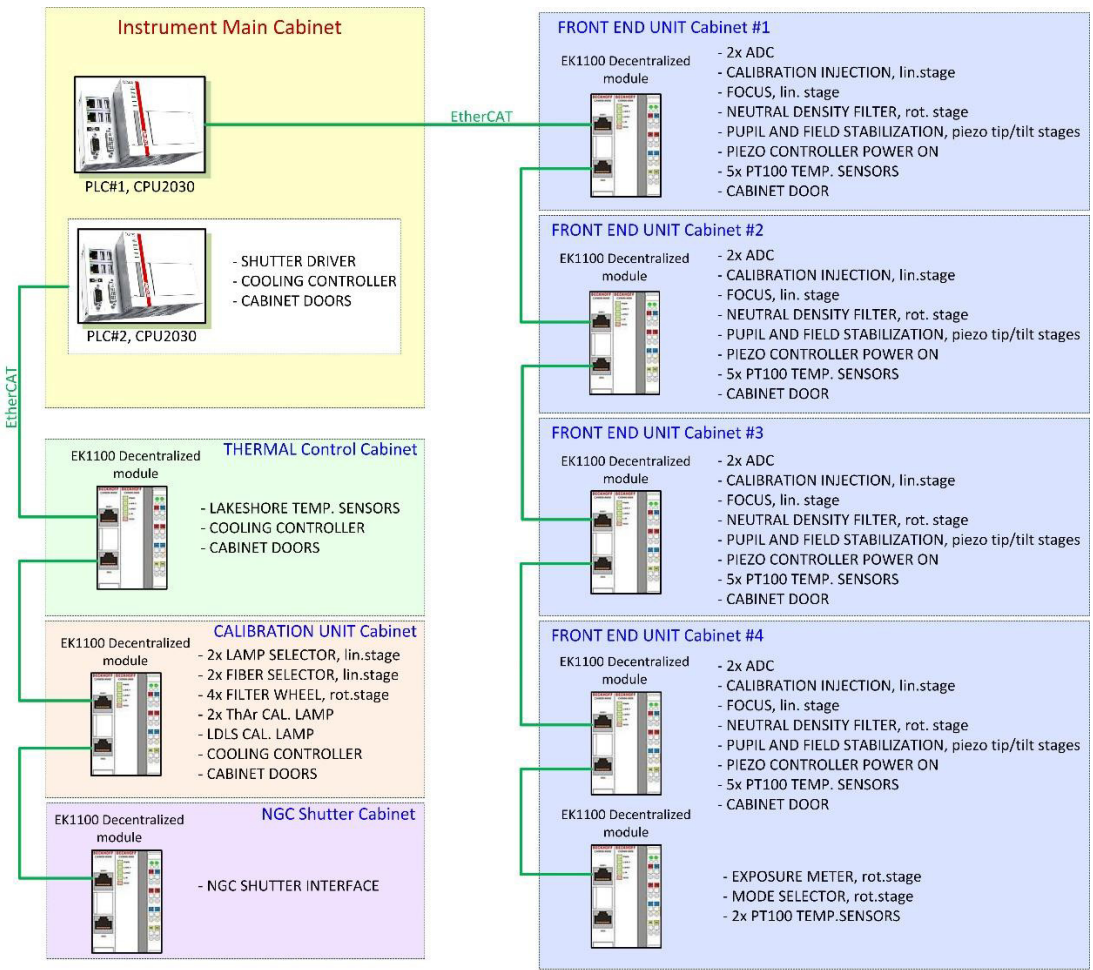


Figure 3: ESPRESSO functions overview.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The EL7342 is a full motor controller with included encoder interface but only its power output stage is being used because it is the one that better fits the requirements. The interface to the encoder is done with the ES5101, since it provides differential lines.

With this line-up of modules, both the velocity and position control loop must be implemented purely with software. The Beckhoff Motion Control (MC) subsystem provides all the necessary functions and it is already integrated in the programming environment. However there are some limitations. With this configuration only a P (Proportional) control loop for the position and a PI (with optional D (Derivative)) for the velocity control loop is possible. Tests have shown that if the velocity loop is well tuned, there are no problems with the position even if only the proportional term is used. Tests performed in the laboratory have also shown that some improvement can be gained by decreasing the cycle time of the motion control loop. It is set by default to a cycle time of 2 ms, but it can be easily changed to lower times. The counter-effect is an increase of the CPU load, but the proportional gain of the loops can then be increased providing a more responsive control without causing instability.

The tuning can be a tricky procedure, but the programing framework provides tools for monitoring and graphing the motor variables, including the current.

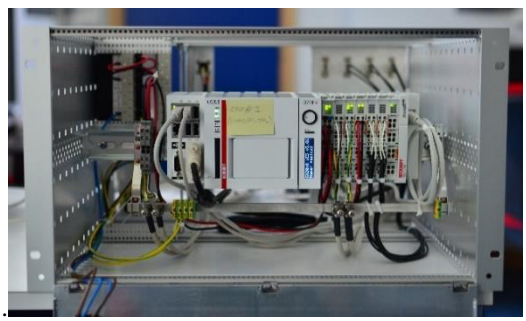


Figure 4: One of the main ESPRESSO PLC CPUs placed in a 19" sub-rack.

CONTROL ELECTRONICS HARDWARE LAYOUT

ESPRESSO CPUs will be hosted in the main ICE cabinet that will be placed in the CCL. In the same place the Thermal Control Cabinet, Calibration Unit Cabinet, Vacuum Cryo Control Cabinet and Fabry Pérot calibration light source will be hosted. These cabinets are 2000 mm high, 800 mm deep and 600 mm wide.

The cabinets are of Schroff Varistar LHX3 type. This variant includes a radial fan and heat exchanger and provides an optimum airflow for cooling of the equipment.

The four Front End Unit Cabinets are only 1200 mm high (see Figure 5) and will instead be placed near each FEU arm. They are equipped with a simpler cooling system. All the cabinets are mounted on a special custom made damper feet to reduce the effects of earthquakes.

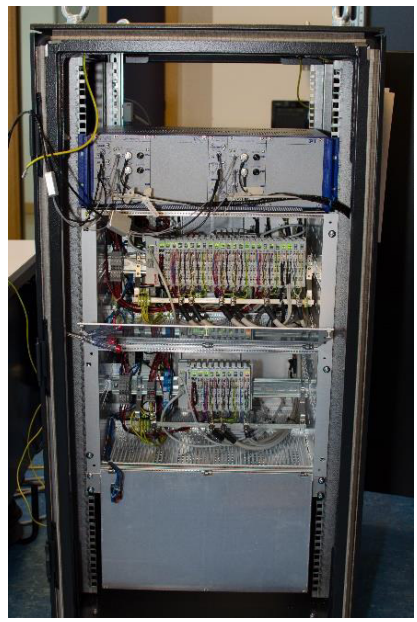


Figure 5: FEU #4 cabinet.

All the PLC CPUs and modules are mounted in 19" sub-racks, similar to those used for the VME crates, as shown in Figure 4. A custom fixture was made to provide support for two standard DIN rails, one facing the front and one facing the back, on which the components were mounted. This is an unusual configuration for PLC, but it was chosen as the best compromise to make them fit in the 19" cabinets. The use of 19" cabinets is due to the presence of other equipment available only in this format. This solution offers some benefits, like the possibility to remove the whole crate from the system for easier maintenance, but the space occupation is not optimal.

Due to the same mechanical composition of each of the four FEU arms, the FEU electronics cabinets will be identical (containing, in addition to the PLC modules, also the piezo tip-tilt controllers) except for the fourth one. This cabinet will host also an additional 19" sub-rack with the PLC modules for the control of the exposure meter stage and sensors and the mode selector rotary stage (see Figure 5).

In the main cabinet, besides the CPU sub-racks, the shutters drivers, a 12" touch panel, a SELCO control alarm annunciator and the network switch will also be hosted.

SOFTWARE SUPPORT

ESPRESSO uses OPC-UA for the communication between the software components running on the PLC and the instrument control workstation [3][4]. During the development and testing phase the use of the OPC-UA protocol [5] proved to be useful since the process variables could be examined and set-up by using commercial OPC-UA clients like Unified Automation UAExpert or Softing OPC client [6]. In this way the PLC software components could be tested without the intervention of the instrument workstation, allowing for a better debugging.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Moreover, the Beckhoff programming environment provides the possibility to build Graphical User Interfaces that can run either on the programming system or on the target CPU. The former possibility was exploited during the initial debugging and testing phase while the latter will be used in the production environment for maintenance and troubleshooting by providing the touch panel connected to the CPU residing in the main instrument control cabinet.

The Beckhoff PLC runtime can runs both on PLCs, and standard desktop PCs which can be used if equipped with a specific network card that acts as an EtherCAT master. While this might not be the best choice in a production environment, it proved to be very useful since the ESPRESSO assembling phase is performed in different sites. The single subsystems could be driven and tested without the need to ship the CPUs at the remote sites and were instead driven by the PCs that were also serving as the programming system.

ASSEMBLING AND TESTING PROCEDURE

The mounting of all the electronics cabinets and sub-racks of ESPRESSO is being done at various sites of the institutes participating to the instrument. Once all the hardware is ready it is fully tested with the instrument control software. The first testing and acceptance procedure occurs at the site where the cabinets are mounted. The FEU electronics is sent to the observatory of Merate (Milano) to be tested also with the fully mounted and aligned optomechanical parts of the FEU subsystem. All the electronic cabinets will be then sent to Genève for the European integration and acceptance. After a successful test of the electrical cabinets and all the devices, all the equipment will be delivered to the Paranal Observatory in Chile for the final integration.

CONCLUSION

The sensors, calibration lamps and moving parts of an astronomical instrument like ESPRESSO for the ESO@VLT needs a flexible, precise in positioning and easy to maintain in the long term control electronics.

For ESPRESSO spectrograph COTS components with a Beckhoff PLC-based control electronics are used.

The Beckhoff PLC decentralization features are fully exploited thanks to the distribution of the electronics sub-racks in different electronics cabinets placed each near the FEU functions to control.

The PLC modules are placed in 19” sub-racks, which may be a disadvantage in term of space occupation but it foresees very high maintainability features.

Schroff electrical cabinet supplier represented the best choice in term of wide selection of cabinet components, ensuring, in the water cooled Varistar LHX3 chosen for ESPRESSO, also a good air flow circulation inside the cabinet.

These cabinets are also earthquake resistant, an essential condition in a highly seismic territory like the Atacama desert.

REFERENCES

[1] D. Mégevand, et al., ESPRESSO: the radial velocity machine for the VLT - Montréal, Canada, SPIE2014, Ground-based and Airborne Instrumentation for Astronomy V, Volume 9147, 91471H.

[2] V. Baldini, et al., ESPRESSO Instrument Control Electronics: a PLC based distributed layout for a second generation instrument @ ESO VLT – Montréal, Canada, SPIE2014, Software and Cyberinfrastructure for Astronomy III, Volume 9152, 915228.

[3] R. Cirami, et al, An OPC-UA based architecture for the control of the ESPRESSO spectrograph @ VLT-San Francisco, USA, ICALEPCS2013.

[4] R. Cirami, et al., Adoption of new software and hardware solutions at the VLT: the ESPRESSO control architecture case, Amsterdam (ND), SPIE2012

[5] Unified Automation website: https://www.unified-automation.com/

[6] Softing website: industrial.softing.com

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A PROTOCOL FOR STREAMING LARGE MESSAGES WITH UDP

C. Briegel, Rich Neswold, Mike Slicznia
FNAL†, Batavia, IL 60510, U.S.A.

Abstract

We have developed a protocol concatenating UDP datagrams to stream large messages. The datagrams can be sized to the maximum of the receiver. The protocol provides acknowledged reception based on a sliding window concept. The implementation provides for up to 10 MByte messages and guarrantees complete delivery or a corresponding error. The protocol is implemented as a standalone messaging between two sockets and also within the context of Fermilab's ACNet protocol. The result of this implementation in vxWorks is analysed.

INTRODUCTION

The Fermilab control system [1, 2] required an alternative to providing large messages. Currently, the user must piece fragments together to support large messages. An atomic message needed to be transmitted guaranteeing delivery and provide the user with an atomic operation.

A working group was formed and discussed several alternatives. The resolution was to support existing protocols but with an extension protocol to concatenate messages with UDP datagrams. The user would not have to modify any code and realize a significantly larger message size. In comparison with other strategies, the implementation provided a solution with little impact to the control system, transparent to the user, minimized risk, and could be accomplished with minimal effort.

First, the implementation was accomplished as a standalone protocol. A simple interface was established to send large messages from one node to another with call-backs when the transmitter or receiver was complete. This minimal test solution would provide a base line for performance as well as a proof-of-principle for the protocol

Also, the protocol was used to increase the message size for ACNet [3], the Fermilab control system's messaging protocol. The added protocol for ACNet was implemented such that all existing protocols could be transparently supported by large messages. This included requests, replies to requests, and unsolicited messages.

PROTOCOL

The protocol in Figure 1 provides a mechanism to concatenate datagrams together. By acknowledging a sliding window of frames received, the message can be delivered reliably to the destination. The protocol abides by the following rules and recommendations:

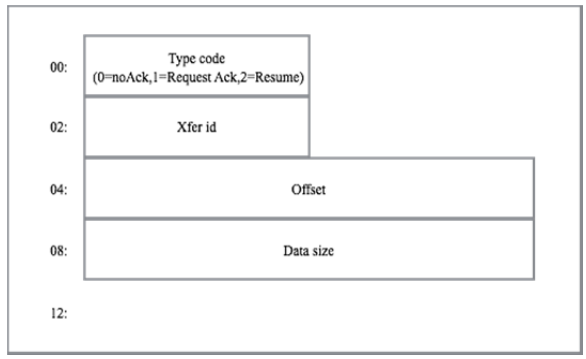


Figure 1: Large Message Protocol.

Rules

- If the offset is zero, then a new reply is arriving. The receiver can use the data size field to pre-allocate a buffer to hold the rest of the incoming data. After saving the data in the buffer, it sets the next expected offset to be equal to the size of data that was just received.
- If the offset is non-zero, it checks to see if the offset and transfer ID matches a reply that is in progress. If a match is found, the data is appended to the buffer and the next expected offset is updated.
- After appending the data, if the packet also asked for a response (type code 1 in the long message header), the task will send a resume message (Figure 2) with the current expected offset.
- If the offset is non-zero and a reply to a transfer ID is in progress but the offset is too high (a packet was dropped), the task waits for a packet that also wants a reply. When it arrives, a resume message is sent to the sender with the offset of the missing data.
- When the transfer is complete, the last packet will also require a response. The receiver returns the expected offset (which at this point will be the size of the data) or a previous offset, if a packet was dropped.

Recommendations

- The first segment **should** use type code 1, asking the receiver for a resume message. By doing this, part of the payload gets sent in addition to checking whether the receiver supports large messages (a timeout indicates no support.)
- The last packet of the message **should** use type code 1 to make sure the entire message was received.
- The sender **may** vary the interval between ACK requests to adapt to network conditions. For instance, the sender might begin the transfer with an interval of 4 packets before asking for an ACK. If there isn't an error, then 8 packets can be sent before the next

†Operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

ACK. If an error occurred, the sender reduces the interval of ACKs.

An alternative algorithm to the sliding window is also permitted. A "bit-map" algorithm allows the receiver to build its copy of the large message with the segments it receives and keeps track of the holes. When a packet arrives with the ACK request, the receiver can go back and specify the earliest hole. With each filler sent, the sender always asks for a reply. The receiver replies with each hole's offset until they are all filled [4].

STAND-ALONE IMPLEMENTATION

The protocol was implemented between two MVME5500s utilizing UDP socket communications. Both nodes ran vxWorks 6.4 operating systems. Repeated messages were sent for 10 Secs with selected large message sizes between 1,000 to 32,000,000 bytes. If a single datagram was needed for the large message, the protocol was still used to guarrentee messages arrived with acknowledgment. The maximum datagram sizes tested were 0x2100, 0x8100, oand 0xF100 bytes.

To establish a base-line, local messages were sent as well as messages over the a 1 Gbit Ethernet. This would provide a comparison of packet latency and associated problems with the network. The result of this local test is in Figure 2 for the various datagram sizes acknowledging every datagram. The maximum long message size displayed on the graph is 1,000,000 bytes since there was negligable change for larger message sizes.

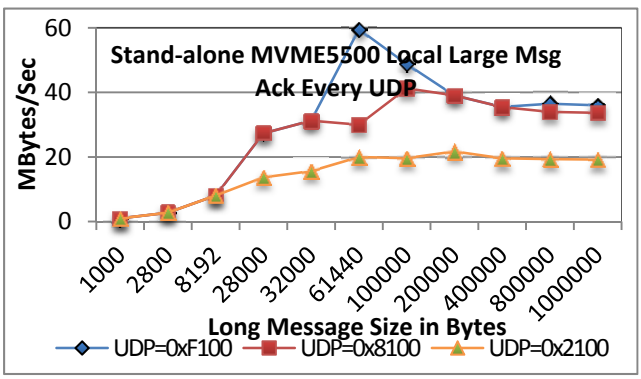


Figure 2: Stand-alone Local Msg Acking Every UDP.

Figure 3 displays the throughput sending messages of varying datagram sizes for a 10 second period. Messages was sent in only one direction over an operatonal 1 Gbit Ethernet acknowleging all datagrams.

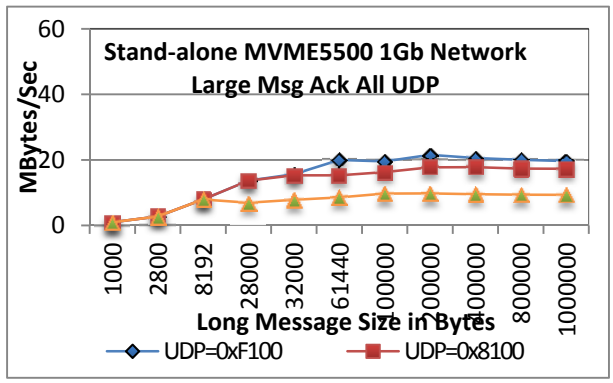


Figure 3: Stand-alone Network Msg Acking Every UDP.

Figure 4 displays the throughput from one node to another using a datagram size of 0xF100 while varing the acking rates.

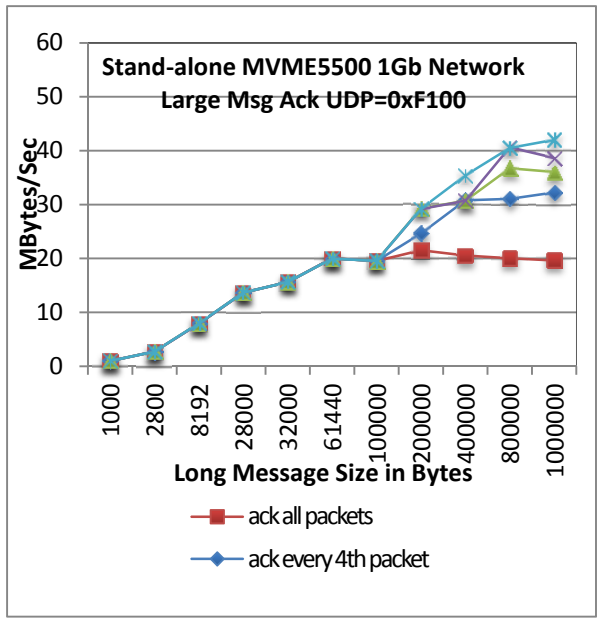


Figure 4: Stand-alone Network Msg, UDP=0xF100.

ACNET IMPLEMENTATION

ACNet is a peer-to-peer protocol which routes messages to connected tasks. ACNet has three distinct messages: unsolicited one-way messages, requests, and replies. Currently, ACNet uses UDP datagrams to implement this protocol and was limited to a single UDP datagram size for it's messaging. Historically, ACNet implemented packeting within the protocol but this was limited to a maximum of 16 packets. The goal was to provide at lease 10 mega-bytes of data in a single message.

The concept was to use a connected task called "LNGMSG" to implement the collection of UDP frames into a single message. If the task was not available on the destination node, then the node did not support the protocol. This would enable an adiabatic implementation across the complex.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Unsolicited messages were sent to the connected task "LNGMSG" followed by the large message protocol and the original ACNet header. The data portion of the message was specified in the large message protocol. The message received by "LNGMSG" would concatenate the data for the original message. After the protocol acknowledged receiving the entire message, the message would be delivered to the original task.

As before, an equivalent test was accomplished between two MVME5500 processors. The primary difference was the messages were sent as requests followed by a single reply echoing the original message. Comparing it with the original request validated the reply message.

Figure 5 is an ACNet local message sent as a request and single reply to itself. Since these messages result in memory moves acknowledging all datagrams was efficient.

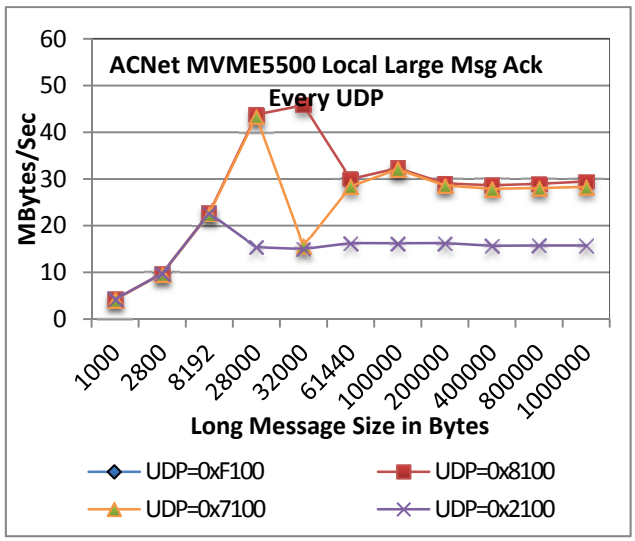


Figure 5: ACNet Local Msg Acking Every UDP.

Figure 6 displays the request/replies over the Ethernet between two nodes with various the datagram sizes acknowledging all datagrams.

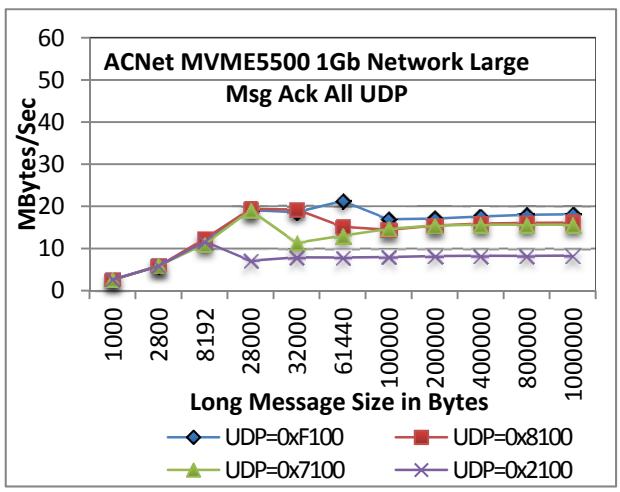


Figure 6: ACNet Network Msg Acking Every UDP.

Figure 7 displays the various acknowledgment intervals for a UDP size of 0xF100 bytes.

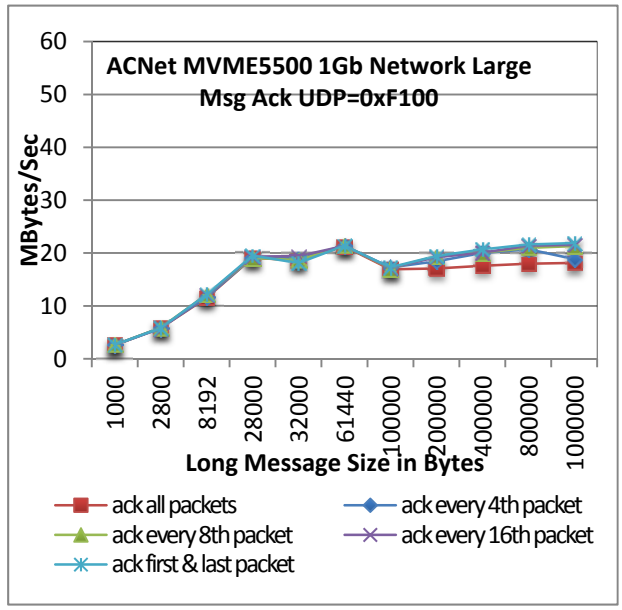


Figure 7: Stand-alone Network Msg, UDP=0xF100.

Figure 8 displays the various acknowledgment intervals for UDP size of 0x7100 bytes which appears to be optimal for the platforms tested.

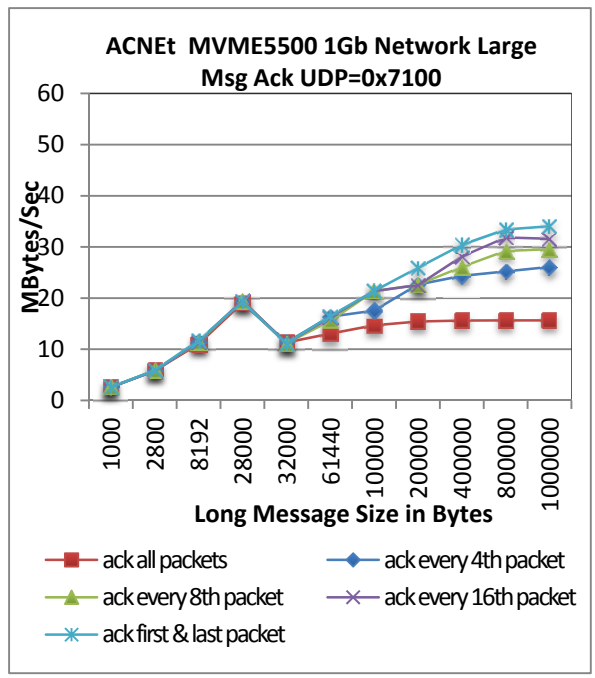


Figure 8: Stand-alone Network Msg, UDP=0xF100.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

PERFORMANCE ANALYSIS

The protocol without ACNet managed to provide minimal processing to combine the frames into a large message. Thus, the sender did not over-drive the receiver. Mismatched nodes or networks will eventually cause buffers to be consumed and packets dropped within the socket software implementation. In vxWorks, the error is observed in the full socket counter returned from a call to udpstatShow.

The ACNet protocol produced the above error more readily since the processing of the ACNet received message requires more processing than the sending of frames. The acknowledgment process provides flow control since the next portion will not be sent until the resume type code is received. Only when the acknowledgment rate decreased did the receiver run out of buffers. For these tests, the transmitter was tuned with small delays for non-acknowledged frames. While the protocol retransmits dropped packets and the large message will arrive, the throughput suffered drastically.

Delay tuning is not an appropriate solution. Providing flow control by acknowlegment is not optimal for throughput. The transmitter of the protocol needs to be adaptive and adjust the acknowledgement rate as needed.

The performance graphs have consistently shown reasonable throughput. The difference between the standalone protocol and ACNet is as expected. The ACNet overhead of concatenating the datagrams is slightly greater to process the ACNet protocols. The resulting imposed transmitter delays in ACNet became the most significant factor in performance differences between these two solutions.

CONCLUSION

The ACNet throughput appears to be adequate for our immediate needs. The protocol as specified is sufficient for guaranteed delivery of large frames via UDP datagrams. Further tests with all nodes in the control system will give a better understanding of system performance.

The implementation on vxWorks required about one person-month of effort. The implementation was transparent to the user. The infrastructure only needed to be rebuilt with changes to the include files. The implementation on Linux is in progress to provide full functionality for the control system.

REFERENCES

[1] J. Patrick, “ACNET Control System Overview,” Fermilab Beams-doc-1762-v1.

[2] K. Cahill, L. Carmichael, D. Finstrom, B. Hendricks, S. Lackey, R. Neswold, J. Patrick, A. Petrov, C. Schumann, J. Smedinghoff, “Fermilab Control System,” Fermilab Beams-doc-3260-v3, http://beamdocs.fnal.gov/AD-public/DocDB/ShowDocument?docid=326

[3] C. Briegel, G. Johnson and L. Winterowd 1990 The Fermilab ACNET upgrade, *Nucl. Instrum. Meth. A* 293 p.235-238.

[4] C. Briegel, C. King, R. Neswold, K. Nicklaus, and M. Sliczniak, ”Large ACNET Mesages, ” 2015.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE NEW MODULAR CONTROL SYSTEM FOR POWER CONVERTERS AT CERN

M. Di Cosmo, B. Todd
CERN, Geneva, Switzerland

Abstract

The CERN accelerator complex consists of several generations of particle accelerators, with around 5000 power converters supplying regulated current and voltage to normal and superconducting magnet circuits. Today around 12 generations of converter control platforms can be found in the accelerator complex, ranging in age and technology. The diversity of these platforms has a significant impact on operability, maintenance and support of power converters. Over the past few years a new generation of modular controls called *RegFGC3* has been developed by CERN's power conversion group, with a goal to provide a standardised control platform, supporting a wide variety of converter topologies. The aim of this project is to reduce maintenance costs by decreasing the variety and diversity of control systems whilst simultaneously improving the operability and reliability of power converters and their controls. This paper describes the state of the on-going design and realization of the RegFGC3 platform, focusing on functional requirements using Thyristor based converters as an example.

INTRODUCTION

The CERN accelerator complex consists of several different machines, each with particular powering characteristics. Power converters are a fundamental requirement of any such machine, being used to provide power to various loads such as magnets (normal, superconducting, dipole, quadrupole …), radio-frequency (RF) cavities and other powered equipment. Evolutions in both accelerator functional requirements and power technology require a regular updating of approaches for converters and their associated control electronics, in particular with enhanced diagnostics and more demanding regulation performance.

Table 1 gives a break-down of typical power converter control platforms, and their use in the CERN accelerator complex.

This diversity of platforms gives rise to significant challenges:

- The age of legacy platforms is expected to give rise to end-of-life effects, where some platforms are already >30 years old.
- Information about these platforms is not always coherently stored and available for experts.
- Some platforms cannot be re-produced, due to component obsolence.
- Each control platform generally has an associated software stack, which needs to be maintained.
- The engineers and system responsibles for these systems are not always available to help in diagnosing faults and issues.
- Operations increasingly want better control over some of these platforms, working on them is becoming increasingly difficult and risky.

This leads to higher failure rates, longer repair times, and generally lower availability of the CERN accelerator complex.

Table 1: Control Systems and Converters at CERN

Control Platform	Machine / Area	Number of converters
MIL1553	PS	1218
PLC	SPS, AD, LIER	510
RS422	PS	366
Junction Crate	North Area	322
MUGEF	SPS, TTs	652
FGC2	LHC	1782
RegFGC3	Various	497

A NEW CONTROL SYSTEM

In recent years, new projects and consolidation activities led to the development of a new control platform, as existing solutions were not able to meet requirements. This presented itself as an opportunity to develop a new approach for power converter controls, whereby a single platform could be developed to provide a standard, modular solution for any past, present or future converter at CERN. This new platform is called Regulation based on the 3rd Generation Function Generator Controller (RegFGC3).

The aim of this platform is to provide for the new projects (e.g. the LHC Injector Upgrade and LINAC4) but also to be applied to any requirements in the coming years [1, 2]. The medium term goal is to reduce the number of control systems in order to improve system maintainability and operability, reducing the burden on the controls sections in the power converter group. In this framework, the EPC group developed the RegFGC3.

THE REGFGC3 CONTROL SYSTEM

The RegFGC3 is a modular converter control platform developed at CERN with the main goal of providing a standardised solution and satisfying as many requirements

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

as possible for power converter controls, whilst using the minimum diversity of boards as possible. The platform uses a standard interface to the CERN control system, via a Function Generator Controller (FGC), in this case the third generation embedded computer [3,4]. The RegFGC3 platform extends the FGC3 capabilities by providing interface modules in order to control power elements of the power converter. The RegFGC3 board portfolio consists of several FPGA-based generic modules that can be used for numerous applications. This principle allows a flexible approach to configuring a control system, adding or removing boards as required. Modules are grouped into classes as shown in Table 2.

Table 2: List of Generic RegFGC3 Boards

Class	Board
Measurement	VS Measurement, VS V2V, VS I2V, VS HVV2V
Interlock	VS Digital Interlock, VS Analog Interlock, VS Beam Interlock
Regulation	VS Regulation DSP, SIRAMATRIX,
Control	VS State Control
Powering	VS PSU

Measurement Boards

The measurement class includes all the modules developed for acquiring analogue signals from High Precision Measurement devices such as Direct Current Current Transformers (DCCTs), Hall probes as well as high voltage dividers. The Current to Voltage (I2V) and Voltage to Voltage (V2V) boards interface the DCCT and deliver adapted voltage levels to the rest of the system. The VS Measurement board is a generic ADC-based acquisition board providing up to 11 analogue measurements. The board digitises and dispatches the data to the regulation system.

Interlock Boards

The RegFGC3 control system implements a redundant protection mechanism based on safety daisy chains shared by all the boards through the backplane. The chains can be opened by any of the boards in the system when a fault is detected. When a chain is opened, the power system is safely stopped. Three generic boards are developed for interlock functionality: the Analogue Interlock board, the Digital Interlock board and the Beam Interlock System (BIS) board. The Analogue Interlock board provides analogue fault detection for up to 16 analogue input channels. Every input analogue signal is compared against converter-dependent thresholds. The Digital Interlock board provides a protection mechanism against discrete fault events such as over-temperature, fast-abort and power rack operation. The Digital Interlock board allows up to 36 discrete signals to be monitored. While the Analogue and Digital interlock boards provide a local protection mechanism for the converter and the load, the BIS board is the interface with the Beam Interlock System, an important component in the accelerator complex which prevents from any accidental release of beam energy [5]. In order to prevent the system from erroneous operation, the BIS board compares the magnet current with four detection windows representing four possible different destinations. When the current value is out of a specific window, the corresponding beam permit is disabled.

Regulation and Converter Control

In the RegFGC3 system the digital regulation can be performed by three different modules: the FGC3, the VS RegulationDSP board or the SIRAMATRIX. Performance capabilities range from 10kHz to 1MHz in regulation frequency. The FGC3 provides all the functionalities required to control a simple converter. It consists of a main board, a communication board and a high-precision ADC-based acquisition board. The regulation takes place in the TI TMS320C6727 DSP while a Spartan3AN FPGA is used for low level peripherals handling. The DSP Regulation Board is based on the TMS320C2834x DSP and it is mostly used for high-frequency PWM control. The third regulation option is represented by the SIRAMATRIX, an FPGA-based regulation board developed for Fast Pulse Converters control. The board includes fast acquisition ADCs, DACs and a Spartan6 FPGA which runs the control algorithm.

The State Control is a generic board which is used to control the converter state machine and timing. It ensures that the low level sequencing of the power converter is executed correctly. For example first closing the main power contactor, then checking the voltage level is correctly achieved, and verifying all subsystems are operating correctly before permitting power to be applied to the load.

The RegFGC3 platform provides several communication interfaces to exchange data between the FGC3 and the different boards. The SPI is a real time interface used to send the reference value from the FGC3 to the regulation board (VS RegulationDSP or SIRAMATRIX). The SCI is a slow asynchronous serial interface used to exchange general data with any card. Finally, the QSPI is a serial communication link for diagnostic information exchange.

THYRISTOR CONVERTERS

Three-phase Thyristor-based power converters are a specific family of converters widely used at CERN especially in the injectors and experimental areas. The operating principle of a Thyristor power converter is to

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

supply from the electrical grid power to a transformer followed by a Thyristor bridge rectifier. An electronically damped output filter is used for filtering of the rectified noise. A 3-phase bridge can be connected in parallel or in series in order to increase the output current or voltage rating respectively. With the evolution of power electronics technologies, fast switching power converters (exploiting for example high power IGBTs), are increasingly used for low and medium power application (100kW up to 1MW). Thyristor power converters remain valid solution for high power applications (up to 150MW). Figure 1 shows the main Thyristor topologies deployed at CERN including single bridge, two bridges series and parallel converters.

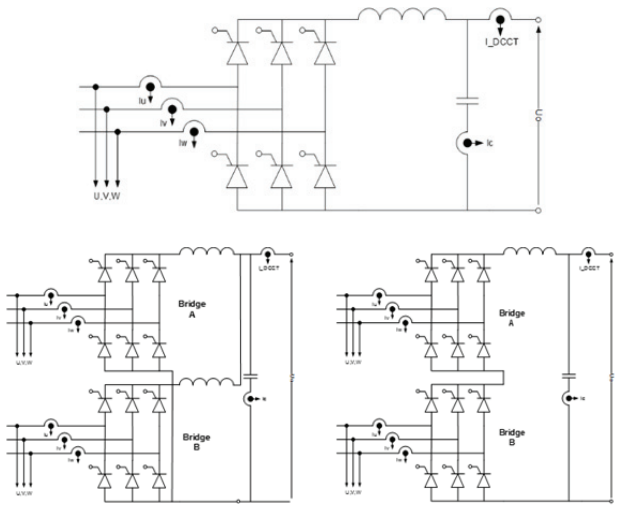


Figure 1 : Thyristor converter topologies.

As shown in Table 3, several power converters are planned to be consolidated in the next five years.

Table 3 : Estimated Number of Converters to Upgrade

Machine	Number of converters	Period
SPS + TTs	218	2016-2020
AD	6	2016-2020
ISOLDE	14	2016-2020
nToF	6	2016-2020
F61	59	2016-2020
LEIR	23	2016-2020
LINAC3	4	2016-2020

The RegFGC3 has been chosen as the candidate for the upcoming upgrade of the Thyristor converter control. Figure 2 shows a control block diagram for a Thyristor converter, using generic electronic boards developed for the platform, and a group of specific boards for the generation of the Thyristor firing pulses (VS Analog Firing, VS Analog Measurement, and VS Drive).

Measurements

The control of a Thyristor converter requires several analogue signals to be monitored for the regulation as well as for protection purposes. This includes input and output voltages and currents as well as the output capacitor current. In the described system, high-voltage measurements are performed using high-voltage dividers while Current Transformers (CT) and DCCTs are used to measure the currents. The I2V board interfaces the DCCT delivering a voltage proportional to the current being measured. The VS Analog Measurement board is a Thyristor converter specific module which receives the voltage measurements from the voltage dividers and makes them available to the rest of the control system.

Power Converter State Machine Control

The power converter operation can be represented by a state machine which defines the different actuations and statuses the converter handles during operation. Usually, the state machine depends on the converter topology being considered. Figure 3 shows the state machine for a Thyristor converter which is implemented in the VS State Control board. The converter starts in the OFF state where the MCB is open, the circuits are not powered and the regulation is disabled. Following a start command, the converter transits into the Starting state, the MCB is closed and the circuits are powered after some delay. Once the circuits are powered, the converter enters the Blocking state waiting for the ON command. When the converter transits into the ON state the regulation and the firing signals are enabled. In the Stopping state the MCB is opened following a control stop procedure. The FAULT state can be reached from any of the states when a fault condition occurs. In this state the MCB is opened and the regulation is disabled. The converter will stay in the fault state until the fault disappears and it is acknowledged by an operator. At this point the converter will transit to the OFF state.

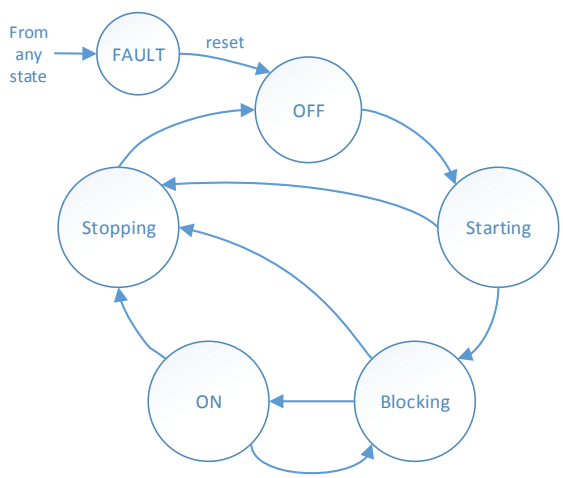


Figure 3 : Thyristor converter state machine.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

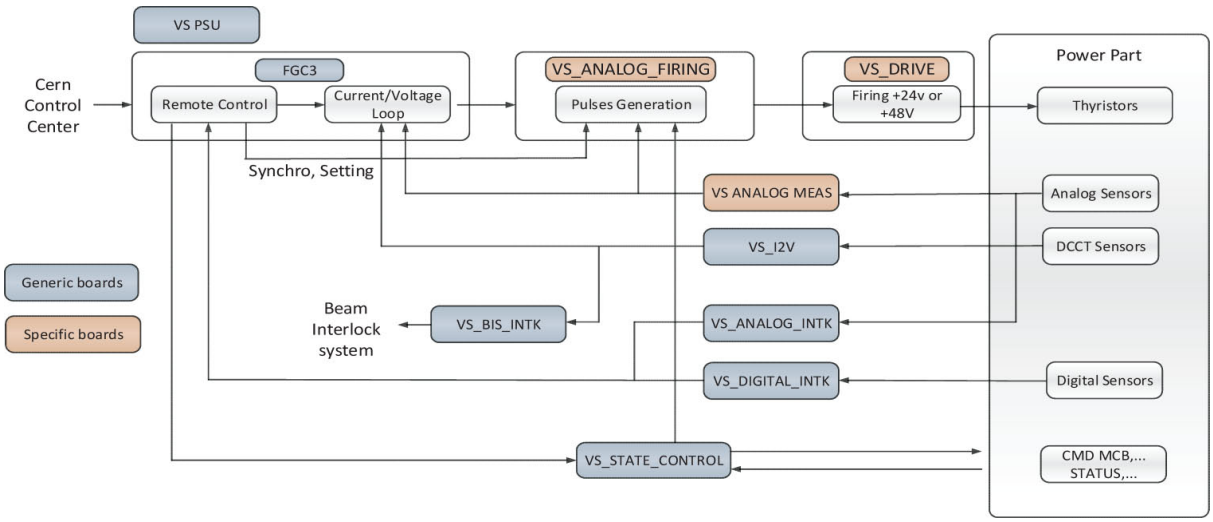


Figure 2 : Thyristor converter control diagram.

Regulation and Thyristor Firing

The core component of the regulation for Thyristor converters is the VS Firing board which is based on an evolution of the Cassel/van der Meer principle used at CERN since the SPS era [6]. The FGC3 implements the current and voltage loop receiving the user external reference through the network and a series of analogue signals required for the regulation. The analogue signals are digitised using four high precision ADCs (MAX5541). The regulation algorithm takes place in the DSP where the RST coefficients for the regulation are calculated using a collection of libraries written in C for function generation and control [7]. The regulation algorithm generates a voltage reference which is sent to the VS Analog Firing board which generates up to 24 low power firing pulses allowing the control of a two-bridge converter. Finally the firing pulses are adapted to high-power Thyristor-gate pulses by the VS Drive board which can drive up to 12 Thyristor gates.

CONCLUSIONS

One of the most challenging objectives for the Electronic Power Converter group at CERN is to reduce the number of converter control systems by using adaptable and scalable modular electronics. The RegFGC3 platform offers a standard solution that can be adapted to many different user requirements using a set of generic boards. Development time and costs are decreased by taking advantage of a common platform. Additionally, the remote diagnostic features provided by the RegFGC3 platform significantly improve troubleshooting and operational issues solving.

ACKNOWLEDGMENT

The authors of this paper are grateful to the MPC section at CERN (Medium Power Converters) in charge of the design, installation and commissioning of Thyristor power converters (https://section-mpc.web.cern.ch/).

REFERENCES

[1] LHC Injectors Upgrade Technical Design Report, edited by H. Damerau et al., CERN-ACC-2014-0337

[2] F. Gerigk (Ed.), M. Vretenar (Ed.), Linac4 technical design report, CERN-AB-2006-084 ABP/RF

[3] D. Calcoen, Q. King, P. Semanaz, “Evolution of the CERN power converter function generator/controller for operation in fast cycling accelerators”, ICALEPCS’11, WEPMN026

[4] R. Murillo-Garcia et all. “Control of fast-pulsed power converters at Cern using a function generator controller”, ICALEPS’15

[5] B. Puccio, A. Castañeda Serra, M. Kwiatkowski, I. Romera Ramirez, B. Todd, “The CERN beam interlock system: principle and operational experience”, IPAC’10, WEPEB073

[6] H. C. Appelo and S. van der Meer, “The SPS Auxiliary Magnet Power Supplies, 10.5170/CERN-1977-012

[7] Q. King et all. “Cclibs: The CERN power converter control libraries”, ICALEPS’15

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MAGNET SERVER AND CONTROL SYSTEM DATABASE INFRASTRUCTURE FOR THE EUROPEAN XFEL

Lars Fröhlich, Piotr Karol Bartkiewicz, Marcus Walla, DESY, Hamburg, Germany

Abstract

The linear accelerator of the European XFEL will use more than 1400 individually powered electromagnets for beam guidance and focusing. Front-end servers establish the low-level interface to several types of power supplies, and a middle layer server provides control over physical parameters like field or deflection angle in consideration of the hysteresis curve of the magnet. A relational database system with stringent consistency checks is used to store configuration data. The paper focuses on the functionality and architecture of the middle layer server and gives an overview of the database infrastructure.

INTRODUCTION

The European X-ray Free Electron Laser (XFEL) is a research facility currently under construction in close collaboration between the European XFEL Facility GmbH[1] and DESY[2] in Hamburg, Germany [1–3]. It consists of a superconducting linear accelerator delivering an electron beam with particle energies up to 17.5 GeV and a total beam power up to ~600 kW, several long undulator sections enabling the generation of extremely brilliant X-ray pulses at wavelengths down to 0.05 nm, beamlines and setups for photon science experiments, and the associated infrastructure.

The electron beamlines of the European XFEL will be equipped with more than 1400 individually powered electromagnets for beam guidance and focusing. In this paper, we give an overview of the various control system components that work together to provide a uniform interface for controlling and monitoring these magnets. We start at the interface between hard- and software with a short introduction to the server architecture of the magnet power supply system. Above this level, a new magnet middle layer server has been introduced that will be treated in more detail. It gives access to physical parameters like magnetic fields and deflection angles in addition to handling hysteresis effects. The paper closes with a few remarks on the database back-end used for the storage of configuration parameters for these and other systems.

POWER SUPPLY SERVERS

As shown in Fig. 1, distributed front-end servers establish a low-level interface to the magnet power supply controllers (PSCs). There are currently two independent PSC families using different communication protocols and interfacing logic. Most of the main magnets (bending dipoles, quadrupoles) along with some correctors are controlled by PSCs designed at DESY using the industrial CANopen communication protocol [4]. The PSCs of the second family originate from Budker Institute of Nuclear Physics (BINP), Novosibirsk, Russia, and use a proprietary protocol [5] which is also CAN-based.

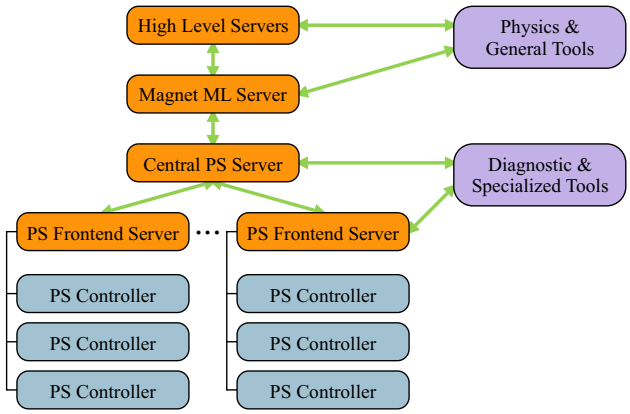


Figure 1: An overview of communication paths between user tools, various server layers, and power supply controller hardware.

An important task of the front-end servers is to hide differences in the interfacing logic and to provide a unified command set that is independent of hardware and communication protocols. The command set covers primitive actions such as switching a power supply on/off, ramping output current, setting current limits, delivering status and error information, and so on. The front-end servers also hide all addressing details, publishing descriptive names of the PSCs rather than their CAN bus identifiers and numbers. The front-end servers can be accessed by a central power supply server (CPSS) as well as by diagnostic applications using the TINE [6, 7] protocol.

The CPSS provides the only access to magnet power supplies for middle layer servers or other clients. It dispatches all requests referring to a single PSC to the corresponding front-end server where they are converted to PSC-specific commands. The CPSS is also capable of accepting composite commands that address PSC groups rather than single units. Such commands are split by the CPSS into several primitive commands and delivered to the front-end servers. The CPSS continuously monitors the front-end servers and collects status information, making it available for middle layer servers and other diagnostic or archiving tools.

MAGNET MIDDLE LAYER SERVER

For previous accelerators at DESY, the main magnet-related quantity provided at the control system level was

[1] European X-Ray Free-Electron Laser Facility GmbH, Albert-Einstein-Ring 19, 22761 Hamburg, Germany
[2] Deutsches Elektronen-Synchrotron DESY, Notkestraße 85, 22607 Hamburg, Germany

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

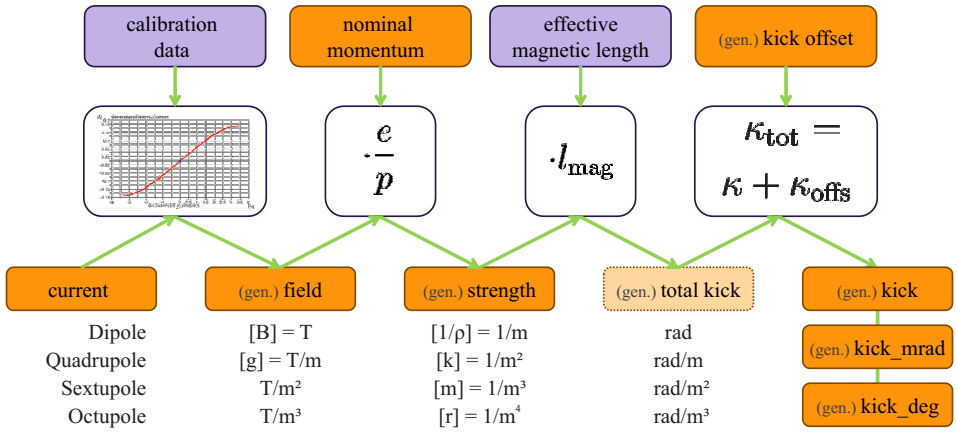


Figure 2: Overview of the various quantities calculated by the magnet middle layer server. User-controllable quantities are shown in orange and static configuration items in purple.

the output current of the power supply. Client software had to rely on additional toolboxes, libraries, or badly maintained ad-hoc code to convert these currents to more physical parameters like fields or deflection angles. At the XFEL, a new magnet middle layer (ML) server has been introduced to provide direct access to these parameters at the control system level. It also offers advanced functionality for the tracking of hysteresis effects, handling of magnet groups and various convenience functions. Where multiple magnets are connected in series to a single power supply, the magnet ML server represents them as individual locations that may or may not have different characteristics.

Implementation

The magnet middle layer server is a DOOCS [8, 9] server written in C++11; as such, it can natively communicate with DOOCS and TINE clients. For back end communication, it maintains a permanent asynchronous TINE link to the central power supply server. The following information is read at a repetition rate of 5 Hz for each power supply:

- current setpoint
- current readback value
- power supply on/off flag
- power supply fault flag
- power supply idle flag

The repetition rate is mainly determined by the maximum read-out rate of the power supply controllers.

Write access from the magnet ML server to the central PS server occurs through synchronous TINE calls and is limited to the setpoint of the current and few special commands.

Physical Quantities

The magnet middle layer server supports the main types of non-pulsed electromagnets used at the XFEL – general multipole magnets (dipoles, quadrupoles, sextupoles, octupoles) and solenoids. As illustrated in Fig. 2, many different physical quantities are calculated from the magnet current and from additional data that is either statically configured or can be changed by the user. To be able to refer to these quantities in a uniform manner across all magnet types, we use a custom vocabulary of generalized quantities:

Generalized field is calculated from the magnet current using a set of calibration curves for the up- and downward current slope (for details see below). For dipoles, this quantity signifies magnetic flux density ($[B]$ = T), for quadrupoles field gradient ($[\mathrm{d}B/\mathrm{d}x]$ = T/m), and so on.

Generalized strength is obtained by multiplying the generalized field by e/p where e is the elementary charge and p is the expected particle momentum at the magnet. This *nominal momentum* p is a setting of each individual magnet and can be changed by the operator maintaining alternatively the field or the strength; usually, it is adjusted to the measured energy profile of the accelerator by a dedicated tool. For quadrupoles, the generalized strength is identical to the k value used in many lattice design and tracking codes.

Generalized kick is defined as the product of generalized strength and effective magnetic length of the magnet. For dipoles, this *kick* actually corresponds to the deflection angle of the magnet, while for other magnet classes it is more commonly referred to as *integrated strength*.

Additionally, the server splits the setpoint for the generalized kick into an offset and a base part. The base part is used for the normal setup of the magnetic lattice whereas the offset is reserved for small corrections from the trajectory feedback. In this way, the main kick property is kept free of feedback noise.

Calibration Functions and Hysteresis

Practically all magnets at the XFEL, with the exception of only a handful of units, have massive iron yokes and therefore exhibit non-negligible hysteresis effects. In other words, their magnetic field B is not a unique function of the excitation current I, but depends on the magnetization history of the material. A multitude of analytical and numerical methods have been proposed for the modelling of hysteretic

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

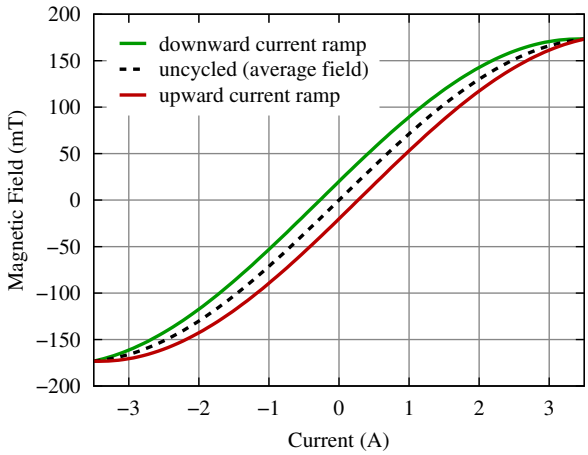


Figure 3: Exemplary calibration curves for a magnet with hysteresis. The opening of the curves has been exaggerated for clarity.

effects in the presence of arbitrary changes of the current or the corresponding magnetizing field H (see e.g. [10–12]), but there is little experience to suggest that these models could satisfy the demanding field precision requirements of the XFEL.

In the development of the magnet middle layer, we have chosen an intermediate approach: The server allows the standardization *(cycling)* of magnets to establish a well-known magnetization state. From this point on, it monitors the current and follows the position on an outer hysteresis curve in the $B(I)$ diagram as shown in Fig. 3. This curve ranges from the minimum of the current, I_{min}, to its maximum, I_{max}, and consists of a branch ramping upwards in current and another one ramping downwards. Both branches are described by independent calibration functions based on fits to measurement data. Currently, 5th-order polynomials are used for most curves. A tanh-based function from [12] finds application for a few magnets with scarce measurement data and bipolar power supplies:

$$B(I) = c_0 I + c_1 \tanh\left[c_2\,(I - c_3)\right] + c_1\left(\tanh\left[c_2(c_4 + c_3)\right] - \tanh\left[c_2(c_4 - c_3)\right]\right)/2$$

When the user sets the generalized field, strength, or kick, the corresponding current is found numerically using a stable solver based on Newton's method.

Obviously, the magnet only stays on its calibrated hysteresis curve as long as the current is changed *with* the ramp direction of the present branch. If it is changed *against* that direction, the server marks the magnet as *dirty*, i.e. in need of cycling. In this case, the exact field is considered unknown and is approximated by the mean value of the up- and downward calibration curves at the given current.

Cycling, Degaussing, and Other Sequences

A generic sequencing algorithm allows the execution of consecutive operations on a given power supply circuit. Command sequences are defined as comma-separated strings using the commands listed in Table1. There are currently three applications for this server-internal sequencer:

Table 1: Main Sequencer Commands

Command	Description
current *amps*	Ramp to the specified current.
field *val*	Ramp to specified generalized field.
kick *val*	Ramp to specified generalized kick.
max	Ramp to maximum current.
min	Ramp to minimum current.
strength *val*	Ramp to specified generalized strength.
wait *secs*	Wait specified number of seconds.

Cycling: A magnet that has left its measured hysteresis curve is in a state where its field is known only with an increased degree of uncertainty. To force the magnet back onto the known curve, it is usually driven multiple times through the cycle from I_{min} to I_{max} and back. Each magnet can be assigned an individual cycling sequence. Until better recipes are found experimentally, the default sequence is *"max, wait 1, min, wait 1, max, wait 1, min, wait 1"*, i.e. double cycling ending with a magnet on the upward ramp.

Degaussing: Magnets with iron yokes retain a remanent magnetization even without excitation current. If a bipolar power supply is attached, this magnetization can be eliminated by driving the current to a specific value or in alternating, decaying patterns. If such a procedure is defined, it can be launched through the server's *degauss* command.

Autocycling setpoints: Current, field, strength, and kick can be set through special *autocycle* properties. If these are used, the server makes sure that the requested target value is reached *and that the magnet ends up on its known hysteresis curve.* Depending on the initial state, one of the following procedures is executed:

- If the magnet is marked as *dirty*, the cycling procedure is executed before ramping to the target value.
- If the target can be reached by ramping in the present ramp direction, the value is set directly.
- If the target cannot be reached in the present ramp direction, the current is first ramped to the minimum or maximum and afterwards to the target value (semi-cycle).

Group Functionality

Magnets can not only be controlled individually, but also collectively. For this purpose, the magnet ML server offers an arbitrary number of group locations that can be used to access multiple magnets with a single command. Depending on the operation, these server properties work with arrays of values (e.g. for reading and setting magnet strengths) or with combined single values like collective status flags (e.g. "all group magnets are on"). Many convenience functions

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

are implemented on this level as well. For example, to cycle only the *dirty* (uncycled) magnets in section L1, a single call to CYCLE_DIRTY on the GROUP.L1 location is sufficient.

The server creates a number of groups on startup based on the magnets' associations with accelerator sections and user defined flags. These automatic groups can neither be removed nor can their list of magnets be modified. In addition, arbitrary user-defined groups of magnets can be created and removed through server commands at runtime.

DATABASE INFRASTRUCTURE

Traditionally, control system servers at DESY read their configuration from local files (using XML or a number of proprietary formats). These files are edited by hand, through specialized tools or scripts, or modified by the servers themselves. This approach necessarily leads to friction where different servers need access to the same information—component names, positions, and calibration constants are typical examples. In fact, given the scale of the XFEL project and the sheer number of individual servers and developers involved, inconsistencies in such *shared configuration items* are the rule rather than the exception.

In order to improve this situation, we have started to store selected configuration items in DESY's Oracle 12c database system. As far as practically possible, database constraints are used to keep the stored data consistent in case of modifications. A few tailored libraries/toolkits for C++, Matlab, and Python have been developed and put at the disposal of developers to allow easy access to the database. The usage of these instruments is slowly picking up throughout the codebase, and the magnet middle layer server is the first one to read almost its entire configuration from the database.

Of course, relying on a central instance like the Oracle system—even if geared towards high availability—introduces a single point of failure. Therefore, all servers store the information received from the database also in their traditional local configuration files. In case of connection problems, the locally cached configuration is used.

CONCLUSION AND OUTLOOK

The European XFEL is a facility of considerably higher complexity than all of the other currently operating accelerators at DESY. With respect to these older machines, a slightly different approach is used to control its more than 1400 solenoids, correctors, quadrupoles, and higher order multipole magnets. The servers controlling the magnet power supplies have been stripped of higher level functionality, and a new magnet middle layer server has been introduced to provide access to physical parameters such as field, kick, and nominal particle momentum. This new server also allows to operate magnets on both branches of their calibrated hysteresis curve, offering advanced functionality for cycling and degaussing of magnets when necessary. The server relies heavily on the configuration database infrastructure that has been recently introduced to minimize inconsistency between shared configuration items in control system applications.

Like most subsystems, the magnet middle layer server is going to be used more intensively starting with the commissioning of the XFEL injector at the end of this year. It is also available at the FLASH facility [13–15] with limited functionality. The operational experience from these accelerators is going to influence the features of the software continually. Among several possible future extensions, the tracking of magnetization inside of the calibrated outer hysteresis curve seems most promising, but requires intensive comparison with measurements and is therefore not viable in the short term.

ACKNOWLEDGMENTS

The authors wish to thank P. Duval (DESY) for his help with the preparation of the manuscript.

REFERENCES

[1] M. Altarelli et al. (eds.), "The European X-Ray Free-Electron Laser technical design report", DESY 2006-097, DESY, Hamburg, Germany (2007).

[2] W. Decking et al., "European XFEL post-TDR description", XFEL.EU TN-2013-004-01, European XFEL GmbH, Hamburg, Germany (2013).

[3] W. Decking et al., "Status of the European XFEL", WEA04, FEL'15, Daejeon, Republic of Korea (2015).

[4] The CANopen website, http://www.can-cia.org/can-knowledge/canopen/canopen/

[5] V. Kozak, "The CPS01 user manual", Budker Institute of Nuclear Physics, Novosibirsk, Russia (2015).

[6] P. K. Bartkiewicz and P. Duval, "TINE as an accelerator control system at DESY", Meas. Sci. Technol. 18 (2007), pp. 2379–2386.

[7] The TINE website, http://tine.desy.de

[8] A. Aghababyan et al., "The accelerator control system at DESY", in: ICFA Beam Dynamics Newsletter 47 (2008), pp. 139–167.

[9] The DOOCS website, http://doocs.desy.de

[10] I. D. Mayergoyz, *Mathematical models of hysteresis* (Heidelberg: Springer, 1990).

[11] J. Tellinen, "A simple scalar model for magnet hysteresis", IEEE Trans. Magn. 34(4) (1998), pp. 2200–2206.

[12] J. Takács, *Mathematics of hysteretic phenomena* (Weinheim: Wiley VCH, 2003).

[13] J. Roßbach et al., "A VUV free electron laser at the TESLA test facility at DESY", Nucl. Instr. and Meth. A 375 (1996), pp. 269–273.

[14] V. Ayvazyan et al, "First operation of a free-electron laser generating GW power radiation at 32 nm wavelength", Eur. Phys. J. D 37 (2006), pp. 297–303.

[15] K. Honkavaara et al., "FLASH: First soft X-ray FEL operating two undulator beamlines simultaneously", FEL'14, Basel, Switzerland (2014), pp. 635–639.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SECURING ACCESS TO CONTROLS APPLICATIONS WITH APACHE HTTPD PROXY

Piotr Golonka, CERN, Geneva, Switzerland,
Hannu Kamarainen, Jyväskylä University of Applied Sciences, Institute of Information Technology, Jyväskylä, Finland

Abstract

Many commercial systems used for controls nowadays contain embedded web servers. Secure access to these, often essential, facilities is of utmost importance, yet it remains complicated to manage for different reasons (e.g. obtaining and applying patches from vendors, ad-hoc oversimplified implementations of web-servers are prone to remote exploit). In this paper we describe a security-mediating proxy system, which is based on the well-known Apache httpd software. We describe how the use of the proxy made it possible to simplify the infrastructure necessary to start WinCC OA-based supervision applications on operator consoles, providing, at the same time, an improved level of security and traceability. Proper integration with the CERN central user account repository allows the operators to use their personal credentials to access applications, and also allows one to use standard user management tools. In addition, easy-to-memorize URL addresses for access to the applications are provided, and the use of a secure https transport protocol is possible for services that do not support it on their own.

INTRODUCTION

Controls applications for numerous systems at CERN's Large Hadron Collider, as well as for technical infrastructure are built with the WinCC Open Architecture SCADA software [1]. Being highly modular and inherently distributed in nature, it allows the separation of the operator consoles (typically in the Control Rooms, or remote access using Terminal Servers) executing the "User Interface" part of the controls application from other critical tasks performed on the control servers. This architecture allows for good scalability in terms of the number of users. A typical problem is however to deliver files, such as panels, libraries or binary plugins, needed at run time, to the operator consoles. These files are application-specific and therefore need to be stored in the control servers where the controls application is running. With over 500 consoles in the control rooms, 20 Terminal Servers and ~200 controla applications that should all be accessible from any console enforcing access control to protect from unauthorized use becomes complex to handle.

In the classical approach applied until now, a file-share is created on every control server, for every application, to make the necessary server-stored files accessible on every console. The fact that files need to be shared to clients running Linux (for consoles) and Windows (for Terminal Servers) operating systems makes it necessary to maintain two parallel file-sharing technologies (NFS and SAMBA), see Figure 1A. Even with an administrative toolbox, support from the operating system administrator, and automated tools to generate application-specific configuration files for the UIs, the management and maintenance of an ever-growing number of applications becomes a time-consuming task. The hardship of resolving numerous performance and stability problems have also prompted us to search for alternative ways of starting the application UIs on operator consoles.

Recent versions of WinCC OA allow for an alternative way of distributing the files to the User Interfaces using the *http* protocol. The files are served by the embedded http server component of WinCC OA, and may be requested as necessary by the clients, who also maintain a cache of the files on a local disk to improve performance. Optionally, the connection to process-data could also be tunnelled by this server, yet we do not use it. In this setup, any machine (with WinCC OA installed) could request the necessary files and hence run the controls application, by simply knowing the appropriate *URL* (address) at which the control server exposes the necessary files. Even though some basic security mechanisms are built into the WinCC OA web server, they are suitable for isolated plants and would not scale for large scale deployments. Moreover, configuring the TLS certificates to obtain a secure https connection is hard to maintain, as well as the fact that there is typically more than a single controls application running on a server. In addition, in the basic setup, we would not be able to assure proper functioning of redundant systems.

In the remaining part of the paper we will present an architecture with a http proxy server in between the server and the UI part of the application, and we will describe the benefits of such arrangement.

ARCHITECTURE WITH A REVERSE PROXY SERVER

A proxy server is a well-known pattern in computer networks, particularly for the World Wide Web: a dedicated server acts as an intermediary between clients and a server that hosts some services. Of our interest are the reverse proxies, that protect the server from uncontrolled access to its services coming from a large network, and perform tasks such as authentication/authorization, load-balancing or certificate handling (SSL/TLS offloading).

The popular Apache *Httpd* web server [2] readily available on Linux platforms, may be configured to act as

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

a reverse proxy for http traffic. This mature technology used in over 50% of World Wide Web installations [3] provides a scalable, secure and robust platform adequate for production systems. It can perform user-authentication tasks through the LDAP technology (which allows users to use their standard CERN credentials), and it may be deployed in clustered setups to provide a higher availability.

We installed a pair of proxy servers (in two separate physical machines) to improve the availability of the setup. We configured the servers as well as the DNS (network Domain Name Service [4]) such that the pair of proxies are visible through the same DNS name, and hence they could accept requests in a round-robin way. Having more than one IP address mapped to the same DNS name is a standard feature of the DNS [5], employed in proxy configurations, and in many scenarios where the maximum time to serve a request is not critical it provides sufficient amount of availability, without resorting to very complex setups.

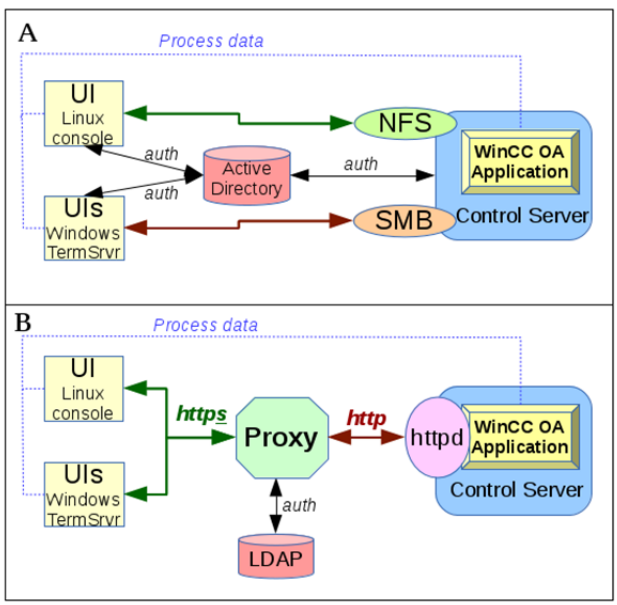


Figure 1: System architecture: (A) with file sharing; (B) with proxy.

NETWORK NAME SERVICE CONFIGURATION

To be able to serve access for many applications the proxy needs to listen for requests at many DNS names; then, by analysing the requested URL, it can determine the proper destination for the request to be forwarded (to one of the embedded WinCC OA web servers running in a controls application on one of the control servers).

Instead of manually registering a new name for every new application, a more advanced setup is used: a wildcard DNS record [4,6], `"*.scada.cern.ch"` pointing to the pair of proxy servers. In effect, a virtual on-demand "scada.cern.ch" subdomain appears in the DNS, allowing us to handle any number of host names.

We are then free to choose the "host name" for every application. The resulting URL that needs to be used to access the application would then have an easy-to-remember form such as `https://MyApplication.SCADA.CERN.CH`.

CERTIFICATES AND SSL/TLS OFFLOADING

A secured variant of the transfer protocol (*https*) should be used by the client and server to assure secure transfer of files and authentication information. This requires TLS certificates to be installed and maintained on the server side. With an already large and growing number of applications, maintenance tasks would become tedious: a signed-certificate would have to be generated and replaced whenever it expires for every single controls application (more than 200).

To sort out this issue we applied the "SSL/TLS offloading" mechanism, with a certificate installed only on the proxy servers. The communication between the client and the proxy (containing sensitive information) is transmitted over the encrypted *https* protocol, whereas the communication between the proxy and the control servers (which can be considered as happening over a trusted channel) is performed over the standard *http* protocol. This way the proxy server *offloads* the control servers not only from the management of their own certificates, but also from the actual task of encrypting/decrypting the communication.

At the technical level, this solution requires the use of a so-called *wildcard certificate* installed on the proxy. This special type of certificate is more complicated to arrange for (it could not be generated by the CERN Certificate Authority), yet it was purchased for us by the CERN Web Service support.

ACCESS CONTROL AND SECURITY

Access to the controls applications needs to be secured and limited to the people who are explicitly granted the necessary permissions. Even though an additional level of access control is provided by the applications themselves, we allow the start of the UI for a controls application only to authorized users.

In the previous approach, the authorization was implemented through file-system privileges of the NFS and SMB network shares, and integrated with operating system authentication and authorization mechanisms (Kerberos, Microsoft ActiveDirectory ™ , CERN egroups), leading to numerous problems with availability, debugging and configuration.

To provide a similar level of configurability and flexibility, we configured the proxy servers to enforce the authorization of use of an application by responding or denying the requests, according to username/password authentication. The standard *mod_ldap* module of Apache httpd has been deployed to integrate with CERN's identity

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

management services over the standard LDAP protocol. User credentials are checked against Active Directory servers, and the authorization decision is based on account membership of the so-called e-groups[7].

For each application, a separate hostname/URL is used for access, and a dedicated entry in the proxy configuration file is placed to define the permissions per application.

To ensure secure access, we configured the WinCC OA embedded http servers to only accept requests from the two proxy servers. We also enforced the use of the secure *https* protocol when connecting to the proxies.

In our solution we chose the *data* connection from the UI program to the control servers ***not*** to be mediated through the proxy. The principal reason is to conserve the high performance and robustness of the process-data communication. We consider this communication channel to be secure enough in the isolated CERN Technical Network, and it could also be strengthened by the use of WinCC OA's built-in Kerberos encryption and integrity mechanisms, if needed.

OPERATION WORKFLOW

The sequence of actions that happen when a user starts a new UI for a controls application of their choice, either on a console or in a Terminal Server (see also Figure 1B), is the following:

1. The user starts the WinCC OA User Interface program, pointing it at the application URL. This causes a *https* request for the application files to be sent to the proxy.
2. The proxy responds with an authentication request, which results in a window opening on the user's UI asking for a CERN user-name and password.
3. The credentials are sent to the proxy, which forwards them to the CERN LDAP server for verification.
4. The proxy analyses the URL to identify the actual application to which the user is requesting access. Using CERN LDAP it checks the user membership in the e-group that was configured for the application.
5. If the answer is positive, the proxy forwards the request over *http*, to the control server that sends the requested file(s).
6. The file(s) received by the proxy are forwarded to the UI client, which saves them in a cache on a local disk, and then proceeds with standard operation.

In the case where one of the proxy servers is not available, and there is a pending request, the UI automatically retries the request with the other proxy, after some short delay.

The special case of redundant WinCC OA systems (e.g. in [8]) is also treated with additional configuration on the proxy servers. Assume that the UI was connected to the first server (through a proxy) and hence received the files from it up to certain point. If there is a switch to the second server, the data connection is routed to the second server by the WinCC OA redundancy mechanism. However, the proxy will not be able to forward requests to the first server any longer; in such situation, it will find out that a timeout has occured, and then it will fall-back to requesting the same files from the second control server in the redundant pair, as expected. This allows for smooth interventions in redundant systems, where the complete server needs to be shut down. Such a scenario was not possible with the file-sharing architecture (when network share became unavailable, the complete UI session is blocked by a non-accessible file, even though data could flow from the second redundant system).

In case of failed authentication or authorization, or mistyped URL, the proxy server informs the client UI which displays the appropriate information. A fall-back web page will be served if the URL is opened in a web browser, displaying information about the application.

CONFIGURATION AND MANAGEMENT

The system is integrated with the central Configuration Database System Information and SCADA Application Service web portal for management and configuration [9]. They allow to enter/change all the necessary information about managed applications.

The configuration files needed by the proxy server software are generated by a set of Python scripts, which connect to the database and pull the information about all declared applications, namely the requested URL for the application, the information about the target application location (control server name, TCP port number of the embedded http server, or access control). The script, run every few minutes, re-generates the configuration files and signals the proxy to reload the configuration, if any change in the configuration is detected.

Certificate management and the start-up of the http proxy processes, as well as the overall security of the system is a standard operating system administration job.

On the control server side, the necessary configuration is automated. The task of starting and configuring (e.g. TCP port number extracted from the DB) the embedded WinCC OA http server is performed by a dedicated component centrally deployed and managed for all the applications.

On the clients, one needs to make sure that OpenSSL libraries are installed in addition to the standard WinCC OA client installation, to enable the secured *https* transport.

PERFORMANCE

Tests were conducted to compare the performance of serving files using http with that offile-shares (NFS, SAMBA). We measured the time necessary to start up one of our largest applications (the supervision of the CERN electrical network [8]) for which a large number of files need to be loaded by the UI. On Windows consoles where files are shared with SAMBA, it takes 34 seconds,

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

when the files are copied to a local disk, this is reduced to 21 seconds . The start-up time when using the http file server with the proxy takes 23 seconds without files in the cache, and 16 seconds when files are cached (Figure 2). This is a perceptible performance improvement for the operators. Simple scalability tests did not indicate any problem with a growing number of clients or applications.

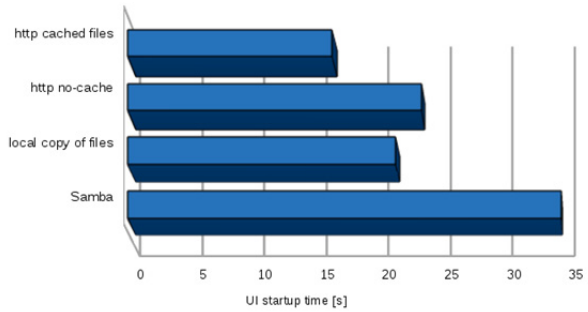


Figure 2: Time to start up the UI for a large SCADA application on a Windows Terminal Server, with Samba and http method.

CONCLUSIONS AND OUTLOOK

The architecture presented in the paper is currently being deployed for pilot use as an alternative way to start UIs for controls applications. We intend to watch the behaviour closely and identify possible limitations (until now the only one found is related to accessing a local cache based on sqlite database, that contains metadata for one of our applications.

The http file transfer provides a good alternative to standard solutions based on file shares, and offers very good performance. It also allows us to address the important case of WinCC OA redundant systems.

Standard, proven technologies and architectural patterns used typically for the World Wide Web prove to be effective also in non-standard applications for control systems. The solution proposed here is generic enough to possibly be implemented using other http proxy software packages (such as Nginx, which is gaining popularity and already takes 25% of web traffic [3]).

By delegating certain aspects of security to a dedicated service that can be maintained independently we will gain on flexibility and increased availability. Thanks to the redundant nature of the proposed architecture, numerous security-related interventions (patches) can now be applied without affecting the running applications.

If needed, the proxy can be used to secure fragile systems with embedded web servers, such as high voltage system mainframes, used in a large number in Detector Control Systems at CERN. It is of particular interest to assure their security for the long-term, as patching firmware to gain security may possibly induce undesired side-effects affecting their stable operation.

The proxy may also work with web sockets (using the `mod_proxy_wstunnel` module, although it requires a newer version of Apache httpd 2.4). This paves the ground for secure access to the future "Web UI" feature, presented also during this conference [10].

REFERENCES

[1] SIMATIC WinCC Open Architecture (previously PVSS) SCADA software from ETM (Siemens subsidiary), http://www.etm.at
[2] Apache HTTP Server Project, http://httpd.apache.org
[3] "Usage of web servers for websites" W3Techs Web Technology Surveys, retrieved on 15 September 2015 http://w3techs.com/technologies/overview/web_server/all
[4] P. Mockapetris,, "Domain Names - Concepts And Facilities", RFC-1034, http://www.rfc-base.org/txt/rfc-1034.txt
[5] T Brisco, "DNS Support for Load Balancing", RFC-1794, http://www.rfc-base.org/txt/rfc-1794.txt
[6] F. Lewis "The Role of Wildcards in the Domain Name System", RFC-4952, http://www.rfc-base.org/txt/rfc-4592.txt
[7] CERN e-groups, http://cern.ch/egroups
[8] J-C Tournier *et al*,"New Electrical Network Supervision for CERN: Simpler, Safer, Faster, and Including New Modern Features", ICALEPCS 2013, October 2013
[9] F.Varela *et al*, "High-level Functions for Modern Control Systems: A Practical Example", ICALEPCS 2013, October 2013
[10] A. Voitier *et al*, "Integrating Web-based User Interface Within CERN's Industrial Control System Infrastructure", ICALEPCS 2015 contribution WEPGF070

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PROGRESS OF THE CONTROL SYSTEMS FOR THE ADS INJECTOR II*

Yuhui Guo, Haitao Liu, Jing Wang, Jiangbo Luo, Yongpeng Wang, Zhiyong He,Ting Liu
Institute of Modern Physics, Chinese Academy of Sciences, Lanzhou, China

Abstract

This paper reports the progress of the control system for accelerator injector II used in China initiative accelerator driven sub-critical (ADS) facility. As a linear proton accelerator, injector II includes an ECR ion source, a low-energy beam transport line, a radio frequency quadrupole accelerator, a medium energy beam transport line, several crymodules, and a diagnostics plate. Several subsystems in the control system have been discussed, such as a machine protection system, a timing system, and a data storage system. A three-layer control system has been developed for injector II. In the equipment layer, the low-level control with various industrial control cards, such as programmable logic controller and peripheral component interconnect (PCI), have been reported. In the middle layer, a redundant Gigabit Ethernet based on the Ethernet ring protection protocol has been used in the control network for injector II. In the operation layer, high-level application software has been developed for the beam commissioning and the operation of the accelerator. Finally, by using this control system, the proton beam commissioning for injector II in the control room has been mentioned.

INTRODUCTION

The Chinese Academy of Sciences initiated an accelerator driven sub-critical (ADS) program in 2011 under the frame of "Strategic Priority Research Program" for the objective of the safe disposal of nuclear waste as well as the potentials for advanced power generation [1]. In an ADS system (e.g. [2]-[3]), the high intensity beams from a proton accelerator bombard a heavy-metal spallation target located at the centre of a sub-critical reactor core to produce an intense neutron source. Then, the sub-critical reactor is driven by this intense external neutron source from the spallation target to achieve criticality.

In the China initiative ADS system, the proton accelerator consists of two injectors, each for the energy of 25 MeV, and a main accelerator which is designed for the energy of 250 MeV and the current of 10 mA. A demo facility for injector II has been manufactured in the Institute of Modern Physics of Chinese Academy of Science. In this paper, we report the control system for the demo facility of injector II.

* This work is supported by Strategic Priority Research Program of Chinese Academy of Sciences (XDA03021503)
Corresponding author, Email: guoyuhui@impcas.ac.cn

THE OVERALL ARCHITECTURE OF CONTROL SYSTEM

Injector II in the China ADS system includes an ECR ion source, a low-energy beam transport line (LEBT), a radio frequency quadrupole accelerator (RFQ), a medium energy beam transport line (MEBT), 4 crymodules and a diagnostics plate (Dplate). Each crymodule consists of 6 superconducting Half Wave Resonators (HWR) cavities, 6 superconducting solenoids, and 5 beam position monitors (BPM). The main responsibility of the control system for injector II is to operate the accelerator injector II in the normal and harmonious fashion, and to monitor various parameters during the operation of the accelerator. The stability, reliability and ease of use are the main issues while designing the control system for injector II. Based on the Epics system architecture, a general three-layer control system is constructed for the ADS injector II [4].

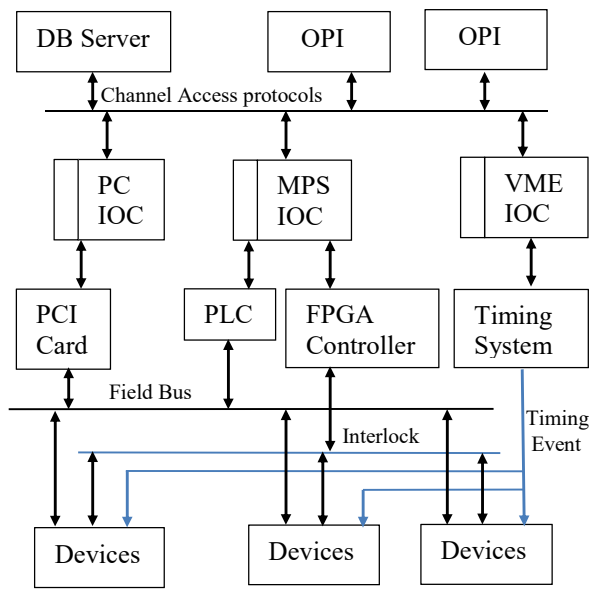


Figure 1: Overall structure of control system for injector II.

As shown in Fig. 1, at the top layer there is the OPI and DB servicer layer. It accesses two low-level machines for interfacing Epics IOC and the field devices by the intranet. At the same time, machine protection system and timing system will be built on the network of real-time control system with optical fiber. Based on the technology of FPGA and PCI bus, fast interlock controller can protect the critical equipment components from damage. The remote control and parameter setting for the field devices is implemented by the PCI cards and the PLC system.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

CONTROL NETWORK

A communication network is the backbone of the networked control systems. The industrial Ethernet switch is used to build the high reliability control network according to the different requests of reliability and data bandwidth. The commercial Ethernet switch is adopted to get the high bandwidth data network. Although industrial Ethernet and commercial Ethernet are designed based on the same standards, they have different working environment requests. The data is transferred by the Ethernet so that the staff can get the operational parameters and states from the central control house, laboratory and even the office. The network that building by industrial Ethernet switch is required to work steady under the extreme conditions, such as high temperature, electromagnetic interference. Thus the critical control equipment and interlocking protection devices is constituted by the industrial Ethernet switch.

Reliability, security, ease of use, and availability are the main issues while choosing the communication type. In order to improve the reliability of industrial Ethernet, various Ethernet redundancy methods based on Ethernet ring protection have been used. The Ethernet ring protection network builds a logical ring topology while maintaining a loop-free forwarding mechanism by logically blocking a link port in the ring, referred to as Ring Protection Link (RPL). Once a link fails, the vertices adjacent to the failure block the failed link, and the RPL is unblocked.

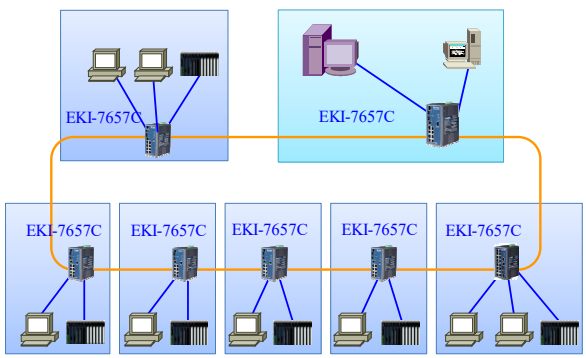


Figure 2: Redundant Gigabit Ethernet used for injector II.

Figure 2 shows the topological diagram of the Redundant Gigabit Ethernet used in the control system for injector II. In order to improve the reliability of the network, 7 EKI-7657C switches produced by Advantech company (http://www.advantech.com/) are used to build a X-Ring Pro that was developed by Advantech. The EKI-7657C supports seven Fast Ethernet ports and three Gigabit combo ports with 2x Digital Input and Digital Output ports. To create reliability in the network, the EKI-7657C comes equipped with a proprietary redundant network protocol, which provides users with an easy way to establish a redundant Ethernet network with ultra high-speed recovery time less than 20 ms. Furthermore, the EKI-7657C also supports many advanced network standards to optimize network performance, ease maintenance issues, and secure network safety.

The X-Ring protocol can help the network system recover from network connection failure within 10 ms or less and make the network system more reliable. The X-Ring algorithm is similar to Spanning Tree Protocol (STP) and Rapid STP (RSTP) algorithm but its recovery time is less than STP/RSTP. X-Ring protocol maintains a loop-free forwarding mechanism by logically blocking a link port in the ring, referred to as X-Ring backup path, e.g. the leftmost link in Fig. 2. Once a link fails, the vertices adjacent to the failure block the failed link, and the backup path is unblocked.

TIMING SYSTEM

The main function of timing system is to provide synchronizing function for the different accelerator modules and devices with synchronous requirement. When ADS injector II is running in pulse mode, the synchronous trigger signal is needed for the microwave power and chopper power-supply of the ion source, low level control system and beam diagnostics device [5]. In order to produce desirable beam bunch, the timing system should provide tow trigger signal to microwave power and chopper power-supply. The pulse width of the trigger signal can be adjusted from dozens milliseconds to several seconds. At the same time, the chopper power-supply's trigger signal, the RF power-supply's pulse giving signal and the beam diagnostics equipment's acquisition window signal should be synchronized by the timing system. The timing system in ADS injector II is provided by Shanghai institute of applied physics, Chinese Academy of Sciences.

MACHINE PROTECTION SYSTEM

A machine protection system (MPS) has been used to protect the machine's equipment from damage induced by beam losses and/or malfunctioning equipment. A set of diagnostic instrumentation installed throughout the injector, such as beam position monitors and wire scanners, will enable MPS to detect critical and non-nominal events. If a non-nominal event is detected, MPS will trigger an alarm, provide protection, or even initiate an emergency shutdown of beam.

Since the response time in the MPS is within several ms, the mainly control logic codes run on the dual redundant RFC460R 3TX controller. The acquisition and given for the control of the field facilities is achieved by the PLC IO modules which are distributed throughout the field that interconnected by PROFINET bus. When the main PLC goes down, the backup PLC will work immediately in several ms, so the time of system died is decreased and the system's reliability is improved [6]. The Machine Protection System manages the protection of vacuum chambers and magnets. In addition, a dedicated interlock system protects the RFQ, beam lines and half-wavelength resonator superconducting accelerator. It consists of one

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

slave PLC for each of the modulators and a master PLC that acquires cooling water and vacuum alarms from the accelerating sections.

Since the response time for the fast machine protection that is in the order of micro-seconds, the time of the fault detection and the time of the feedback control should be within 10 us [7]. As shown in Fig. 3, we have designed the fast controller for the fast machine protection system based on advanced FPGA chip and high-speed fiber optic network technology.

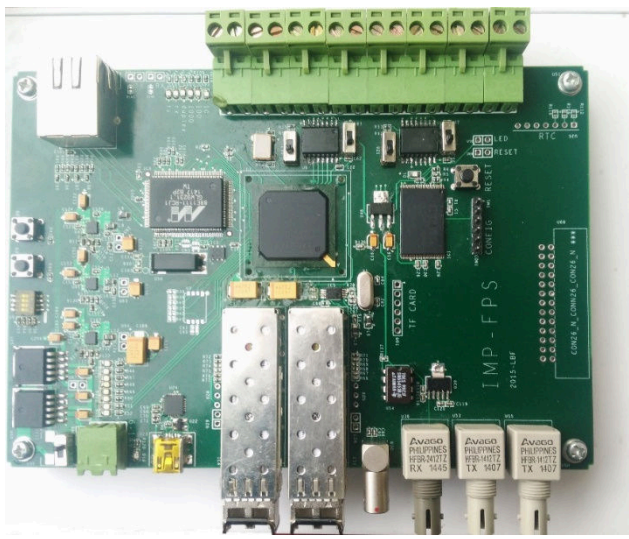


Figure 3: The fast machine protection controller based on FPGA and SFP.

The FPGA controller use Spartan 6 series products XC6SLX45T which was made by Xinlinx Company [8]. The SFP optical modules with the rate of 1.25 G/s are adopted. These modules, which are used to interconnect different modules, have the same encapsulation though there rate are different so that they can be exchanged [9]. The control interface between facilities has two forms which is TTL signal and 5 M optical fiber receiver HFBR2412. The Machine Protection System can protect the costly facilities such as RFQ and superconducting cavity from high energy beam damage. It relies on various detectors (beam loss monitors, Low level system, ion chambers, dosimeters, current monitors, etc.) and inhibits the beam by control the chopper power-supply.

LOW LEVEL CONTROL

In the above-mentioned three-layer control system, the bottom-layer is the equipment layer where various sensors and actuators are used to control the equipment in injector II. The low-level control in the equipment layer means the industrial standard control modules for these sensors and actuators. For example, in the water-cooled system, the low-level control includes the sensors and actuators for the measurement of temperatures, the measurement and control of water flow speed, and the switch-on or switch-off of various valves. The low-level control is achieved mainly by the following industrial control cards, programmable logic controller (PLC) and peripheral component interconnect (PCI).

The PLC used in the ion source system is Siemens S7 series 300 [10]. The inline controller series of Phoenix Contact has been adopted for the temperature detection of the beam line, the flow capacity of the cooling water and the chain protection system of the temperature. Profibus and Interbus are used to connect distributed peripherals to the CPUs, which communicate with the control system via Ethernet interfaces using the TCP/IP Send/Receive protocol and a dedicated Epics IOC server.

PCI card is mainly used on the motion control and the data acquiring systems. The PCI-1220 of Advantech Company is used to control the opening size of the air intake valve of the ion source, the position of the beam scraper, the motion of the Faraday cup. At the same time the PCI-9114 of Adlink Company is used to acquire data of the beam parameters for beam diagnostics devices, such as ACCT, DCCT. The PCI card can be debugged locally with an operator panel or remotely through a TCP/IP Ethernet interface and a dedicated Epics IOC server.

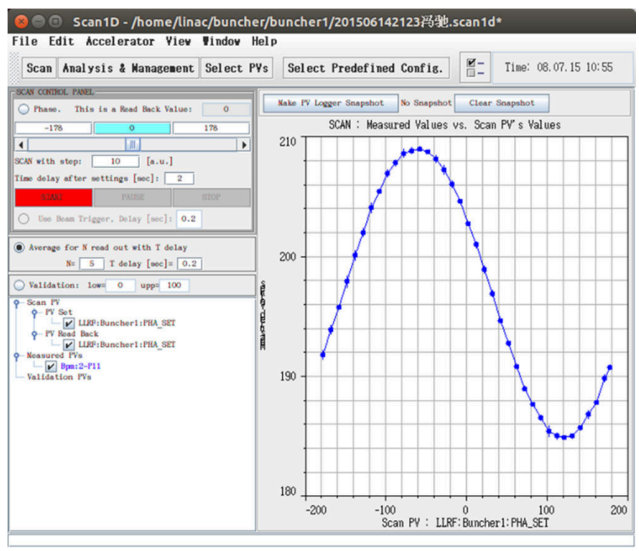


Figure 4: Phase scanning interface of buncher.

HIGH LEVEL SOFTWARE APPLICATION

In the top operation layer of the three-layer control system, various application software have been developed for the beam commissioning and the accelerator running, such as the application based on open XAL. As shown in Fig. 4, the phase scan program is used to determine the maximum acceleration phase of the accelerator. The cavity phase is scanned from 0 to 360 degree step by step. The BPM phase is recorded during scanning, whose maximum point means maximum acceleration. The x axis is cavity phase and the y axis is BPM phase in the curve. Since the curve is similar to sine wave, Fitting of the curve to sine wave will give out the phase and amplitude of the cavity for cavity setting.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The operation platforms in the central control room and subsystem's device control cabinet have been installed. The graphical user interface can display the parameters and the status during the operation of the injector. Figure 5 is the interface for the beam commissioning of injector II. The control system studio (CSS) is used to develop the operation interface which is based on JAVA. Compared with other tools, such as MEDM and EDM, the interface can be developed more practically by CSS. Thus, it has been used as a main OPI tool in the control system of injector II.

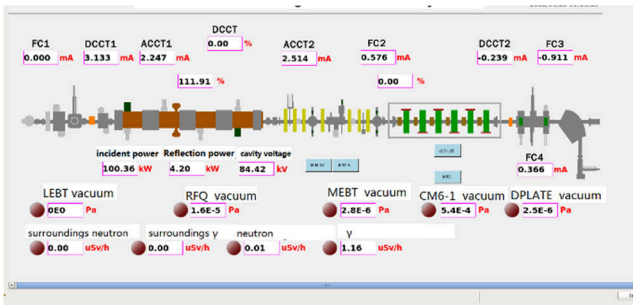


Figure 5: The interface for the beam commissioning of ADS injector II.

DATA STORAGE SYSTEM

With the development of the database and network technology, the information technology has been applied in the control system of accelerators. In the control system for injector II, the acquiring, managing and archiving of its data is an essential task. The main responsibility of the data storage system is to store multiple copies of all data and to be able to recall them on request. The stored data include the parameters of all devices, all information during the beam commissioning and the operation of the injector. The data storage system of ADS injector II can get data fast and provide friendly interface so that make it convenient to queries and displays the stored data.

The data storage system of ADS injector II is composed of two rack-mountable servers that are IBM System x3550 M3 services, and one storage system of IBM DS3500 storage system. The software for the data archived system has been developed with the Keep-alive and MySQL, in order to build a high-availability redundant database system. This system has dedicated to archive the beam related data and device status information [11].

CONCLUSION AND DISCUSSION

This paper has reported the progress of the control system for accelerator injector II used in the China ADS facility. Designed for the energy of 25 MeV, injector II consists of an ECR ion source, a LEBT line, a RFQ, a MEBT line, and 4 crymodules. By using this control system, the proton beam commissioning with the energy of 2.6 MeV has been finished in the control room in Nov. 2014, where the subsystems, ECR+LEBT+RFQ+ MEBT + a testing crymodule that consists of one superconducting cavity, are tested. Furthermore, the proton beam commissioning with the energy of 5.3 MeV has been finished in the control room in June 2015, where the subsystems, ECR+LEBT+RFQ+ MEBT+ a crymodule that consists of 6 superconducting cavities, are tested. The beam commissioning with the maximum energy of 25 MeV will be taken in the near future.

REFERENCES

[1] W.L.Zhan, H.S.Xu, et al., "Advanced fission enery program-ADS transmutation system", Bulletion of Chinese Academy of Sciences, p.375-381,Chinese (2012).

[2] Yan Fang,Li Zhihui,Tang Jingyu,et al., "Preliminary physics design of China Accelerator Driven Sub-critical System main linac", High Power Laser and Particle beams,p.1783-1787, Chinese (2013).

[3] Geng Huiping,Tang Jingyu ,Li Zhihui, et al., "Design of the second medium energy beam transport line for China Accelerator Driven System", High Power Laser and Particle beams,p. 1005-1008, Chinese (2013).

[4] Guo Yuhui,Yu Chunlei,Xu Weibin, et al., "Design of Software Platforms for Injector Ⅱ Control System in ADS",Atomic Energy Science and Technology,p.29-32, Chinese (2014).

[5] Xu Weibin,Guo Yuhui, Zheng Yawei, et al., "Design of synchronous controller for accelerator based on FPGA", High Power Laser and Particle beams,p.151011-151013, Chinese (2015).

[6] Yu Chunlei,Guo Yuhui,He Yuan, et al., "Design of Control System for High Intensity Proton RFQ Based on Redundancy Technology",Atomic Energy Science and Technology,p.740-745, Chinese (2014).

[7] Zheng Yawei,Xu Weibin,LuoBinfeng, et al., "Sub-control system based on PXIe Controller for high intensity proton accelerator", High Power Laser and Particle beams,p.951011-951015, Chinese (2014).

[8] Xilinx DS162(v1.11), "Spartan-6 FPGA Data Sheet:DC and Switching Characteristics",January 2011; http://www.xilinx.com

[9] Xilinx UG386(v2.2), "Spartan-6 FPGA GTP Transcelvers",April 2010; http://www.xilinx.com.

[10] Jiang Ziyun,Guo Yuhui,Liu Haitao, et al., "Control system of ion sources for injector of accelerator driven sub-critical system",High Power Laser and Particle beams,p.551021-551024, Chinese (2014).

[11] Bao Xun,LI Chuan,Xuan Ke, et al., "Accelerator data collection system based on Channel Archiver",High Power Laser and Particle beams,p.589-592, Chinese (2008).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INFORMATION SECURITY ASSESSMENT OF CERN ACCESS AND SAFETY SYSTEMS

T. Hakulinen, X.B. Costa Lopez, P. Ninin, P. Oser
CERN, Geneva, Switzerland

Abstract

Access and safety systems are traditionally considered critical in organizations and they are therefore usually well isolated from the rest of the network. However, recent years have seen a number of cases, where such systems have been compromised even when in principle well protected. The tendency has also been to increase information exchange between these systems and the rest of the world to facilitate operation and maintenance, which further serves to make these systems vulnerable. In order to gain insight on the overall level of information security of CERN access and safety systems, a security assessment was carried out. This process consisted not only of a logical evaluation of the architecture and implementation, but also of active probing for various types of vulnerabilities on test bench installations.

INTRODUCTION

Information Security of Control Systems

While network accessible server and desktop systems are nowadays out of necessity fairly well secured, control systems have until recent times been largely ignored when it comes to information security. This is due to the fact that most such systems are by their nature private and disconnected. Control system vendors also tend to strongly recommend that their systems be kept in isolation of publicly available networks, for which reason neither the vendors nor the clients may have seen the need for rigorous security. However, tendency nowadays is to increase information exchange between control systems and the rest of the organization to allow, e.g., supervision and remote control, and to facilitate operation and maintenance. Recent events, such as the infamous Stuxnet episode [1], have also served to demonstrate that even well isolated systems are not necessarily immune to security problems.

Access and Safety Systems

CERN GS/ASE group is responsible for the management of CERN personnel access and safety systems. This includes specification, design, implementation, contracting, operation, and maintenance of these systems. The current principal access and safety systems under our responsibility are LHC Access Control System (LACS), LHC Access Safety System (LASS), PS Access Control System (PACS), PS Access Safety System (PASS), Personnel Safety System of the SPS complex, Surveillance of Sites system (SUSI), CERN Safety Alarm Monitoring system (CSAM), Gas detection and alarm system (Sniffer), Site Information Panels / Simple Access Messages system (SIP/SAM), and Safety System Atlas (SSA).

CERN access and safety systems typically consist of many different kinds of devices, such as Windows and Linux servers, operator posts, panel-PCs, PLCs, video cameras, interphones, card readers, biometry scanners, etc. These devices come from many different vendors, and they are nowadays mostly directly network connected. Access systems reside mainly in the restricted CERN Technical Network (TN) but some devices reside also in the CERN-public General Purpose Network (GPN). The most important systems have their own private networks. Most systems also have their respective test platforms, which aim to replicate the production systems in sufficient detail, albeit in a smaller scale, in order for development and testing to be possible without endangering production.

We carried out an information security assessment of two of our most visible access systems, LACS and PACS. The work consisted first of creating an inventory of all the devices and services, which included establishing dependencies between systems, and then carrying out active penetration testing of the various services using available security tools or writing specific utilities, whenever necessary. The goal was not only to find any existing vulnerabilities, but also to gain a larger understanding of the principal risks involved and to develop procedures for managing such risks.

INVENTORY

The first task in a security assessment of a complex system is to create a comprehensive inventory of the entire system: types of devices, operating systems they run, services they use and provide, interdependencies they have. After all the devices were identified and categorized, representatives of the most interesting devices within a category were selected and focused on. Fifteen generic groups were identified within the test systems: servers, access point PCs, operator post, gateway PLCs, access control units, biometry scanners, webcams, network devices, interphone master stations, intercom servers, interphone client stations, UPS monitoring devices, virtual machine servers, generic windows PCs and generic PLCs. Every device was also reviewed for brand, model, operating system, installed software, and running services, as shown in Table 1.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Table 1: Part of the Inventory of LHC Test Bench Devices. Device Names Are Grayed Out

Type	Name	Brand	Model	OS	Software/ Services
Access point PC	LHC0	IEI	PPC-5150	WINDOWS 7	MS Terminal Service, MS Windows RPC
Windows devices	LHC0	HP	COMPAQ DC7100	WINDOWS 7	MS-DS Active Directory, MS Terminal Service
Windows devices	LHC0	HP	PAVILION	WINDOWS 7	MS-DS Active Directory, MS Terminal Service
Windows devices	LHC0	HP	PROLIANT	WINDOWS 2008	MS-DS Active Directory, MS Terminal Service, Oracle MTS Service
PLCs	LHC0	SIEMENS	S7-1200	STEP7	Industrial Port, HTTP

Interdependencies between devices were identified. These include the following:

- Access point PLCs depend on access point PCs.
- Biometry scanners depend on biometry servers.
- Webcams depend on video servers.
- Access control devices and operator posts depend on access control servers.
- Intercom devices depend on intercom servers.
- Everything depends on network devices.

Results from an interdependency analysis focus attention to the important testing targets: which devices are critical for the correct functioning of the entire system and what the consequences of a breach of a certain device could be to those that depend on it.

SECURITY ASSESSMENT

Methodologies

The different probing methodologies used in this assessment were deterministic (local or remote) and fuzz testing (fuzzing) attacks. Deterministic techniques, like local and remote attacks compromise a system precisely as specified. For example a local exploit may need a particular operating system version and a remote exploit may need a specific service or application running.

The idea of fuzzing is to try to proof the software against incorrectly implemented code, by testing it with non-deterministic (fuzzy) techniques. Fuzzing is used by software developers and auditors to find exceptions, which are not properly handled. Fuzzing techniques don't usually pose precise requirements, because they make use of different services or operating systems.

Preliminary Testing

After the initial inventory, a period of preliminary pilot testing of a small number of devices was carried out to determine the scope of the project, any special conditions, and the tools needed. The scope was determined based on typical attack vectors and types in information systems as presented in Table 2.

Table 2: Typical Attack Vectors and Types [2]

Attack Vectors	Attack Types
Code Injection	Buffer Overflow Buffer Underrun Viruses Malware
Web Based	Defacement Cross-Site Scripting (XSS) Cross-Site Request Forgery (CSRF) SQL Injection
Network Based	Denial of Service (DoS) Distributed Denial of Service (DDoS) Password and Sensitive Data -Interception Stealing or Counterfeiting Credentials
Social Engineering	Impersonation Phishing Spear Phishing Intelligence Gathering Tailgating

Tools

Several security-testing frameworks were evaluated, and the following tools were chosen for the project:

- **Metasploit Framework** is an open source penetration testing software project for security officers and administrators.
- **Armitage** is a GUI for the Metasploit Framework.
- **nMap** is a network mapping tool for discovering services, open ports, operating systems, and subnets.
- **Wireshark** is a versatile protocol analysis tool that displays network packets in a human readable form allowing sophisticated filtering and analysis.
- **Backfuzz** is a multi-protocol fuzzing toolkit supporting the most important network protocols.
- **W3af** (Web Application Attack and Audit Framework) is an open source tool for finding vulnerabilities in web-applications.
- **Nikto** web scanner detects outdated software, dangerous files and CGIs on web servers.
- **BeEF** is a penetration-testing tool that exploits web browsers. It creates a dedicated server on the attacker system to listen for connections.
- **THC Hydra** is a password-cracking tool that uses dictionary attacks against servers and databases using various protocols.
- **THC flood_router26** is a denial-of-service script that floods the network with router advertisements.
- **THC smurf6** is an IPv6 tool for DDoS (Distributed Denial-of-Service) attacks.

Penetration Testing

The tools were managed using a special Linux distribution, Kali Linux [3], which comes standard with a suite of security tools. This system was used as the platform for all penetration testing as well as network and system monitoring. As the network segment of the PS access system test platform is private to us, we were able to carry out tests, which normally might have been risky on publicly accessible segments.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Findings

The findings were classified using probability and criticality rating defined in the testing guide of the Open Web Application Security Project (OWASP) [4]. The classification contains 8 categories, of which two first are shown as an example in Table 3.

Table 3: Probability and Criticality Rating of Issues

	Probability Rating		Criticality Rating
1	**Skill level of hackers**	1	**How much data is affected that could be disclosed**
	(1) No technical skills (3) Some technical skills (4) Advanced computer user (6) Network & programming skills (9) Security penetration skills		(2) Minimal non-sensitive data disclosed (6) Minimal critical data disclosed (6) Extensive non-sensitive data disclosed (9)Extensive critical data disclosed or all data disclosed
2	**How motivated they are**	2	**How sensitive is the data that could be disclosed**
	(1) Low or no reward (4) Possible reward (9) High reward		(2) Minimal non-sensitive data disclosed (6) Minimal critical data disclosed (6) Extensive non-sensitive data disclosed (9)Extensive critical data disclosed or all data disclosed

Among the issues discovered were things like missing or vendor-default passwords in embedded devices, unpatched bugs in Windows machines causing them to crash or freeze, unsecured web-interfaces, open ports and unnecessary services on devices. Most of these issues can be fixed by configuration and patching. However, sometimes vendor patches for some embedded devices are not available despite requests, and the only options are to either isolate them or decide to ignore the risk (the latter option applicable to non-critical devices only).

A particular worry is the sensitivity of PLCs to intrusion and denial-of-service attacks. A simple network scan can crash a PLC and even sending commands to access PLC memory or to reboot it is possible using existing toolkits [5]. Network interfaces of old generation PLCs are often badly implemented and very intolerant of disturbances. Newer generation units can often be better secured. Of particular importance is protecting the Siemens 400 series PLCs acting as gateway devices between more restricted safety and more relaxed access systems. These units must be run in the password-protected mode.

Tools for Best Practices

Best practices of the computer security industry are available and tools exist for their implementation, auditing, and enforcement. We tried two: Lynis [6], for auditing Unix and Linux systems, and Open Vulnerability Assessment System (OpenVAS) [7], which is a combination of several services and tools. These are principally frameworks for gaining information and understanding of the systems and to be used in connection with other tools.

OTHER FINDINGS

While carrying out the security assessment, other security issues were also tested besides those directly connected with the target systems. Interesting observations were made on how a seemingly private network could be compromised, what risks are involved in the deployment of IPv6 in a network segment, and in particular, what the importance of physical security of installations is.

Tunnelling Out of Private Networks

One of the main reasons for isolating access and safety systems within their own private networks is security. Reducing or even completely blocking information exchange between networks greatly reduces the probability of successful remote attacks. This becomes particularly important considering that critical systems may not be able to follow the same update and patching cycles as less critical ones. Older systems may also be running obsolete hardware and software, which is known to be insecure, but which cannot be upgraded without a complete redesign of the system – a lengthy and expensive process possibly even requiring revalidation by national authorities.

While the critical safety systems are normally in strict isolation, we have also implemented some less critical access systems within their own private network segments but with controlled routing to other CERN networks. In this case only a few hosts can be accessed from outside of the private segment, but all devices enjoy basic central network services, DHCP, DNS, and NTP. It turns out that it may be possible to exploit DNS for unauthorized, difficult-to-detect tunnelling from the private network through firewalls all the way to the public Internet. This is based on the observation that DNS protocol allows inserting arbitrary data in the query and response packets, which can hence be used to transfer data. All that is needed is that DNS be allowed to resolve outside host and domain names from inside the private network. Iodine software [8] was used for this test:

1. A special DNS client is installed on a machine in the private network.
2. A special DNS server is set up in the Internet with its own top domain.
3. The client makes a DNS query to a subdomain of the top domain with a data payload attached.
4. Server answers with its own data-stuffed packet.
5. Client makes another DNS query to a different systematically named subdomain in order to avoid DNS caching by intermediate servers, etc.

While DNS service is useful for internal name resolution, blocking this exploit requires that hosts from private networks not be able to make DNS queries to the outside domains. Firewall could also be configured to inspect DNS packets for suspect payloads.

IPv6 Issues

IPv6 [9] has been right around the corner for the last 20 years. Only recently has it really started to be implemented, as the IPv4 address space is finally getting scarce. In addition to a much larger address space, IPv6 provides a number of new functionalities to facilitate network management, but which can also be used to subvert network security. As the protocol has only been in widespread use for a relatively short time, return of experience on it is still limited. As an example, we tested two particular is-

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

sues having to do with the new Stateless Address Auto Configuration (SLAAC) feature of IPv6:

1. A malicious host can freeze a Windows host by sending a large number of router advertisements with a different route prefix thus simulating different subnets. Until recently, Windows didn't limit the size of IPv6 routing table, which allowed this attack to fill up the system virtual memory and crash the system. After Microsoft update MS14-006, which limits the system routing table size, the system no longer crashes, but stays at 100% CPU while an attack is running and recovers after. The same behaviour is seen with Linux systems during an attack.
2. A man-in-the-middle attack is possible in mixed IPv4 and IPv6 networks. If the routers in the network are IPv4, a malicious host may use IPv6 router advertisements to hijack connections because an IPv6 router has a higher priority than an IPv4 one.

There are ways to mitigate these kinds of problems on the host level: firewall rules can be used to allow traffic only to known hosts and routers, router discovery can often be turned off on the host network stack, or IPv6 can be completely disabled on the host. The last option is particularly reasonable for control systems, where IPv6 support is in any case missing or often poorly implemented.

Importance of Physical Access

When considering any system, and in particular a critical isolated one, restricting physical access to it is of prime importance. An expert able to access a device can in principle do whatever he wants with it. It may not even be necessary to access the device in person, but to have someone else do it instead, e.g., by tricking the person to connect an infected USB key into a restricted system using social engineering. This is in fact very likely the way the Stuxnet worm was first introduced into the classified Iranian nuclear installations.

In supervised areas it may not be possible for an intruder to work with a device without being noticed even if he could access the facilities. There may also not be enough time to carry out compromising changes to a system even if a lapse in security could be exploited. However, devices exist that make this kind of an intrusion a lot faster and easier. We tested a USB keyboard injection device Rubber Ducky [10], which installs itself as a keyboard device to a system and runs a script that can do whatever a user could from the console. The device looks like any USB key as seen in Figure 1.

Figure 1: USB keyboard injection device Rubber Ducky.

Preparing a script for the injection device obviously requires good knowledge of the system in question. However, as most control system computers use commodity operating systems like Windows, this is often not a difficult thing to figure out. Once inserted into the system, the device is able to play a script very fast minimizing the exposure time of the intruder.

A Simulated Intrusion Scenario

It is instructive to conceive a scenario for an intrusion based on the above findings. Assume a semi-critical system running in its own private network but allowing basic network services from outside for convenience:

1. The intruder finds out basic information of the target: operating system, possibly any SCADA systems running. This kind of information is often surprisingly easily available.
2. The intruder accesses the facility by some pretext (cleaning crew, commercial representative, etc.).
3. At a suitable opportunity, using a keyboard injection device, the intruder installs a DNS tunnelling client to an operator console machine with an open session in the private network.
4. Now the intruder has full transparent access to the infected machine. There's a good chance that the account used to run the operator console has admin privileges and can be used to install low-lever network sniffers, password crackers, fake routers, etc.

CONCLUSIONS

We have presented results of an information security assessment carried out on some of CERN access and safety systems. A number of equipment were found vulnerable and subsequently patched or otherwise secured whenever possible following the best practices in the industry. We also had some surprises in learning about recent intrusion techniques and devices, which make the life of an intruder a lot easier than it needs to be. Lessons learned include ways to mitigate the risk of intrusions:

- Strict access controls to sensitive areas to know who enters and when.
- Devices in locked racks away from manipulation.
- Disabling of any unnecessary network protocols.
- Updated firewalls and monitoring of suspect traffic.
- Defense-in-depth: keep even isolated devices updated and patched as much as possible.

REFERENCES

[1] https://en.wikipedia.org/wiki/Stuxnet
[2] B.J. and B. Andrew, "Hacking with Kali", Elsevier, (2013).
[3] http://www.kali.org
[4] https://www.owasp.org
[5] http://libnodave.sourceforge.net
[6] http://sourceforge.net/projects/lynis
[7] http://www.openvas.org
[8] http://code.kryo.se/iodine/
[9] https://en.wikipedia.org/wiki/IPv6
[10] http://usbrubberducky.com

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INCREASING AVAILABILITY BY IMPLEMENTING SOFTWARE REDUNDANCY IN THE CMS DETECTOR CONTROL SYSTEM

L. Masetti, A. Andronidis, O. Chaze, C. Deldicque, M. Dobson, A. Dupont, D. Gigi, F. Glege, J. Hegeman, M. Janulis, R. Jiménez Estupiñán, F. Meijers, E. Meschi, S. Morovic, C. Nunez-Barranco-Fernandez, L. Orsini, A. Petrucci, A. Racz, P. Roberts, H. Sakulin, C. Schwick, B. Stieger, S. Zaza, CERN, Geneva, Switzerland
P. Zejdl, CERN, Geneva; Fermilab, Batavia, Illinois, USA
U. Behrens, DESY, Hamburg, Germany, O. Holme, ETH Zurich, Switzerland
J. Andre, R. K. Mommsen, V. O'Dell, Fermilab, Batavia, Illinois, USA, G. Darlea, G. Gomez-Ceballos, C. Paus, K. Sumorok, J. Veverka, MIT, Cambridge, Massachusetts, USA
S. Erhan, UCLA, Los Angeles, California, USA, J. Branson, S. Cittolin, A. Holzner, M. Pieri, UCSD, La Jolla, California, USA

Abstract

The Detector Control System (DCS) of the Compact Muon Solenoid (CMS) experiment ran with high availability throughout the first physics data-taking period of the Large Hadron Collider (LHC). This was achieved through the consistent improvement of the control software and the provision of a 24-hour expert on-call service. One remaining potential cause of significant downtime was the failure of the computers hosting the DCS software. To minimize the impact of these failures after the restart of the LHC in 2015, it was decided to implement a redundant software layer for the control system where two computers host each DCS application. By customizing and extending the redundancy concept offered by WinCC Open Architecture (WinCC OA), the CMS DCS can now run in a fully redundant software configuration. The implementation involves one host being active, handling all monitoring and control tasks, with the second host running in a minimally functional, passive configuration. Data from the active host is constantly copied to the passive host to enable a rapid switchover as needed. This paper describes details of the implementation and practical experience of redundancy in the CMS DCS.

ROLE OF THE CMS DETECTOR CONTROL SYSTEM AND EXPERIENCE DURING RUN-I

The DCS of the CMS experiment enables the coherent and safe operation of the detector. It provides the supervision of the experiment, allowing for the operation of the sub-detectors and sub-systems and works in synchronization with the LHC, automatically preparing CMS for data taking whenever LHC is ready to generate collisions. The availability of the DCS is crucial to ensure the operation of the experiment and to maximize data taking time.

The CMS DCS consists of an interconnected federation of independent applications, all of which are based on the WinCC OA control system toolkit from ETM professional. The central CMS DCS team takes responsibility for the infrastructure and platform on which the control applications run. This includes Windows operating system, WinCC OA software and other software dependencies such as the OPC DA servers used to communicate with front-end hardware.

WinCC OA is a modular and distributed software package where the functionality is implemented in independent processes, called *managers*, that communicate via the TCP/IP protocol. The central processing unit is the *event manager* which keeps the current process image in memory and ensures the distribution of data to other managers. The process image is organized in typed, structured process variables, called *data points*. A WinCC OA *project* is a configured set of managers running in a single server, typically connected to a local event manager. A WinCC OA *system* identifies a namespace corresponding to one event manager (or two redundant event managers in case of redundant systems). Many systems can be connected forming a distributed network which enables data exchange between systems.

The CMS DCS ran successfully throughout the first data taking run of the LHC with high availability. However, a potential cause of significant downtime was the failure of server hardware, which would immediately stop the hosted control application. In order to restore full functionality of the CMS control system, the central team would intervene urgently to prepare a new server and target the unavailable application to the new computer. This operation was time consuming (up to 1 hour) and could introduce unexpected problems due to the complexity of the installation process.

In order to mitigate the issue of server failure, it was decided to implement redundancy at the control software layer by running each CMS WinCC OA system on two computers, one located in the underground cavern and the other in the computer centre on the surface, in order to overcome a failure of the services or infrastructure at a single geographical location. In this way, if a single server would fail, all CMS DCS control applications would continue to run and a deferred intervention could then be planned to replace the failed hardware. Details on the selected server hardware can be found in [1]. The present paper presents the final software solutions adopted during the full migration of the CMS DCS to the redundant architecture.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

REDUNDANCY IN WINCC OA

The WinCC OA control system toolkit provides a built-in mechanism for achieving high availability through redundant systems. In this approach, two complete, identical instances of each project run on two separate computers, called *peers*. One project runs as the *active* peer while the other runs as a hot standby or *passive* peer. While the system runs in a fully redundant configuration, data changes in the active peer are constantly synchronized across to the passive peer so that it is always up-to-date. If the active peer fails or a manual switchover is requested, the passive peer immediately takes over the role of the active peer.

To avoid both peers sending and receiving value updates from the front-end hardware, the standard WinCC OA drivers running on the passive peer are programmed to discard commands and data point updates.

CONSTRAINTS IN CMS DCS

Although the CMS DCS is based on the WinCC OA toolkit, some implementation decisions and technologies that are used mean that the standard approach of running two identical peers is not feasible or possible. For instance, the CMS DCS makes use of significant quantity of CERN software that has been built upon WinCC OA, such as the JCOP Framework [2]. This software was not initially written with the requirement of supporting redundant systems. While many JCOP components could be quickly modified to become fully compatible with redundant contexts, there were some components where the fundamental design and implementation choices prevented any simple route to achieve redundancy readiness.

Moreover, many hardware devices (PLCs, power supplies) used in CMS do not support connections from two servers, or their performance drops significantly with multiple connections. Also, contrary to typical industrial applications where control systems operate with the same configuration for months or even years, the CMS DCS systems frequently evolve with time. Smooth handling of component installation is essential, so the use of redundancy must not complicate installation for the sub-detector developers. Interruption of the system during upgrades can be tolerated because upgrades are initiated by sub-detector experts who can schedule the intervention at an appropriate time.

An additional factor to be considered is that the behaviour of any non-trivial code running simultaneously on two peers can diverge because of timing issues or the limited functionality of the passive peer. For instance, even if the actions on the process variables (data points) are disabled on the passive peer, all the side effects of the managers, such as sending emails or opening a TCP socket must be disabled in the passive mode otherwise they will be executed twice (once per each peer). To inspect and modify all the potential issues in the code of every CMS sub-detector was not feasible, hence an alternative solution had to be found.

DUPLICATION OF ACTIVE/PASSIVE STATUS

The selected approach to address the aforementioned issues is to run the complete application including drivers and custom scripts only on the active peer and to run a minimal configuration on the passive peer (Fig. 1). This minimal configuration is similar for any application, consisting of the basic WinCC OA managers plus two CMS managers to handle redundancy. Mock drivers, called *simulation managers*, must also exist on the passive side for each corresponding driver on the active side to enable synchronization of the complete set of WinCC OA configuration data.

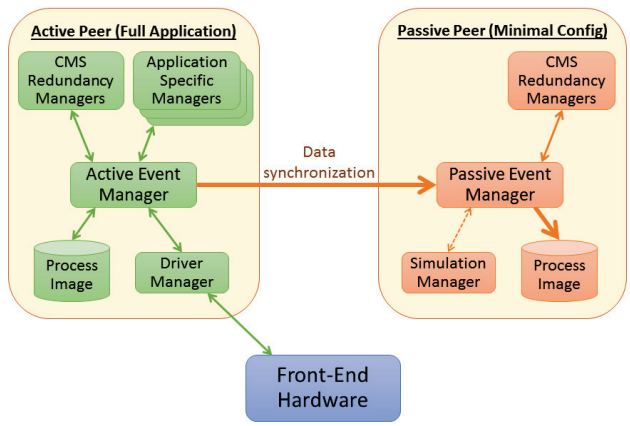


Figure 1: A typical configuration of active and passive peers.

The standard WinCC OA active/passive status of each peer is combined with a CMS-specific active/standby status. WinCC OA status is driven by WinCC OA, depending on the built-in redundancy mechanism. The CMS status reflects the configuration of the running managers: the passive configuration corresponds to the minimal configuration that should run in the passive peer, while any other configuration is considered active. All four combinations of states are possible but only two (WinCC OA Active/CMS Active and WinCC OA Passive/CMS Standby) are stable (Fig. 2).

	CMS Active	CMS Standby
WinCC OA Active	Normal (Active)	Restart as CMS Active or wait for the other to become active
WinCC OA Passive	Restart as CMS Standby or Force Active	Normal (Passive)

Figure 2: Combinations of WinCC OA/CMS Redundancy Status and possible actions to exit unstable configurations.

When an unstable combination is reached, custom logic will act in order to bring the peer to a stable configuration. The action taken depends on several conditions (e.g. checking if the other host is pingable). When a peer detects the configuration WinCC OA Passive/CMS Active, it can exit this situation by forcing itself to be active or by restarting itself with the minimal configuration. A peer that detects the situation WinCC OA Active/ CMS Standby will either

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

perform the transition to become CMS Active or wait for the other peer to become fully active.

INSTALLATION OF COMPONENTS IN A REDUNDANT SYSTEM

All the control software used in the CMS control system is provided in the form of packages, which can be deployed and updated via a dedicated web interface. As the CMS DCS installations are applied to running systems, frequently the start of the installation process involves switching to a safe, reduced functionality state (e.g. stopping the connection to hardware). The installation and upgrade of the components is handled in a dedicated tool developed centrally at CERN in the JCOP framework. This tool supports redundancy and keeps track of which version of the component is installed in each of the two peers. Following this approach the complete installation has to be repeated in each peer. The installation tool provides two operation modes for redundancy:

- *Installation in Active Mode*: The component is installed in the active peer and no redundancy switch is performed.

- *Installation in Split Mode*: When a component must be installed, the system is set to *split mode*, that is the two peers become independent, and the installation is performed on the passive peer. After installation, the passive peer becomes active and redundancy is re-enabled. The process is repeated in the newly passive host and finally the redundancy is enabled again.

However both approaches are problematic in the CMS environment. Using the first approach, any new component will not be installed in the passive peer until it switches to active. This is the wrong moment to install a component, because it will cause unscheduled downtime in the active peer during the critical moment of the switchover. Moreover, in this case, the installation will be unattended because there is no expert explicitly requesting it.

The second approach causes two redundancy switches during the installation of any component. This increases downtime when upgrading/installing a component, which would be tolerable for large, complex installations. For small, rapid upgrades and patches, the overhead of two redundancy switches is less acceptable.

The CMS solution is to install the components in the active peer (as it was done before redundancy was introduced) and then perform a light version of the installation in the passive peer. The possible elements of a typical installation were analysed to find the best way to deploy them to both peers.

- *Changes in the process image (data points, data point types etc.)*: the creation / modification of new data points or data point types is the basic purpose of an installation and does not pose any problem to redundancy because the process image is automatically synchronized by WinCC OA in the passive peer.

- *Configuration files*: Some project configuration files might be updated during installation. These files can be synchronized from the active to the passive peer by the CMS redundancy managers.

- *Addition of new WinCC OA managers*: During installation, new managers might be added to the project in order to perform specific tasks. These managers are not immediately activated in the passive peer in order to preserve the minimally active state, but their configuration is stored internally so that they can be started in case of a switch to CMS active mode.

- *Files inside the project directory*: Local files used in the project are synchronized from the active to the passive peer by the CMS redundancy managers.

- *Installation tool receipt to track which component is installed in each peer*: The receipt can be written on the active peer and automatically synchronized to the passive peer, so that the installation tool believes the component is already installed in the passive peer and will not trigger another installation in case of a switchover.

- *Special actions interacting with the operating system*: The developer can define special actions that interact with the operating system (e.g. modifying the Windows registry to configure an OPC Server). Such actions need to be repeated in both peers and must be placed in special scripts that are executed during the switchover to CMS active mode. The execution of these special actions is delayed until the switchover to ensure that they run with the most recent configuration data.

In case of a redundancy switch, the passive system is activated, already having the latest components installed, and becomes fully operational within a few seconds or several minutes, depending on system conditions and the specific application complexity.

This approach for redundant installation is almost transparent for the sub-detector developer, requiring only minor component changes related to special operating system actions. The central CMS DCS team have minimized these changes by preparing scripts for the most common cases.

REDUNDANCY SWITCHOVER CONDITIONS

The main purpose of redundancy in CMS is to handle the situation where the server hosting a peer fails completely. Further improvements can be made to trigger a switchover in other common failure modes.

Typical checks included in the list of switching conditions are related to the server resources. For example free memory and free disk space are evaluated in each server and cause an increase in the error level of a peer when falling under a certain threshold. This is a default feature of WinCC OA that has been re-used, but the thresholds have been redefined

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

because of the large amount of memory available on the servers where the DCS applications run.

Moreover, other application specific switching conditions are configurable in the CMS DCS implementation. The same condition (e.g. is the PLC pingable) is periodically evaluated in two peers. The state is propagated when the condition is stable, i.e. a waiting time (> polling time) has passed since both peers evaluated the condition. This avoids unnecessary switches in case the error is present on both peers, but was detected at different times due to desynchronized polling cycles (e.g. if an essential PLC is not pingable from both peers there is no benefit to switch).

This feature was implemented using *CMSfwClass* [3], a CMS library that implements object-oriented behaviour in WinCC OA, making it straightforward to extend a base class to define new custom conditions.

HANDLING EXTERNAL PROCESSES

The CMS DCS is mainly implemented using WinCC OA but it also needs to run other external processes. These processes typically need to be stopped on the passive peer and started in the active peer following the redundancy switch of WinCC OA. Examples of these processes include the name servers for the custom CERN Data Interchange Protocol (DIP) and Distributed Information Management System (DIM) protocols and OPC servers (that are automatically started with the WinCC OA OPC client but need to be stopped in the passive peer). A centrally-developed CMS tool allows component developers to configure the non-WinCC OA processes to be started or stopped in case of redundancy switch. This mechanism is connected to the CMS active/standby state rather than to the WinCC OA state. This defers the starting/stopping of the process until the complete activation of the project.

SPLIT BRAIN SCENARIO

When the network between two redundant peers goes down, both peers activate because they cannot contact the other server. When the network recovers, one peer must be killed and restarted in passive mode. By WinCC OA default behaviour, peer 2 will always be killed after a network outage. This is not the desired behaviour. Rather we want, if all other conditions are the same in the two peers, to retain the previously active peer in active mode. It was possible to obtain this behaviour by raising the error level of the newly active peer for a few seconds when switching from passive to active state because the other peer is not contactable. By enabling the relevant WinCC OA option, when the two peers reconnect, the one that had the highest error level while disconnected is restarted in passive mode.

AUTOMATIC RESOLUTION OF INTERNAL DATA POINTS

Data points that represent values specific to each individual peer (e.g. manager status, monitoring of the memory or of the disk) need to be duplicated (with the appending of a _2 postfix for the second peer) and configured to be writable also from the passive peer. The CMS DCS team developed an abstraction layer in form of a library that provides redundant compatible versions of the basic WinCC OA functions dpGet, dpSet and dpConnect (used for reading, writing or connecting to a value). These functions will automatically replace the data point names, adding the _2 postfix when needed. The default is to use the data point related to the active peer in a UI connected to both peers and the data point related to the connected peer in case of scripts connected to one peer only. The only modification for the sub-detector developers was to replace the standard WinCC OA functions with the CMS redundant-compatible versions.

CONCLUSIONS

The strategy for redundancy presented in this paper has been successfully applied to the CMS DCS during the first LHC long shutdown and is currently operating in 34 WinCC OA systems running on 29 pairs of servers.

The approach adopted in CMS DCS eliminated the need for adaptations in many third-party software components while enabling the full benefit of an automated switchover to a hot standby system. In this way the CMS DCS does not require urgent expert intervention as the result of issues with a single server.

The implementation fulfils the original requirement of providing robustness against total server hardware failure and, furthermore, triggers a switchover in other failure conditions such as exhaustion of memory and disk space.

By creating an abstraction layer over the redundancy tools of WinCC OA and CMS, the central CMS DCS team minimized the impact on the sub-detector developers in terms of the code changes needed to prepare their applications for running in the redundant environment.

All major issues related to the redundancy were successfully addressed, while keeping additional complexity to a minimum. The current configuration is expected to guarantee high availability and reliability of CMS DCS during the Run-2 phase of the LHC accelerator.

ACKNOWLEDGEMENTS

This work was supported in part by the Swiss National Science Foundation.

REFERENCES

[1] G. Polese et al. “High availability through full redundancy of the CMS detector controls system”, Journal of Physics: Conference Series Volume 396 Part 1 (2012).

[2] O. Holme et al. “The JCOP Framework”, ICALEPCS’05, Geneva, Switzerland, October 2005, WE2.1-6O.

[3] R. Jiménez Estupiñán et al. “Enhancing the Detector Control System of the CMS Experiment with Object Oriented Modelling”, MOPGF025, these proceedings, ICALEPCS’15, Melbourne, Australia (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A DATA ACQUISITION SYSTEM FOR ABNORMAL RF WAVEFORM AT SACLA

M. Ishii#, M. Kago, JASRI/SPring-8, Hyogo 679-5198, Japan
T. Fukui, H. Maesaka, T. Inagaki, T. Maruyama, T. Ohshima, Y. Otake, RIKEN/SPring-8, Hyogo 679-5148, Japan
T. Hasegawa, M. Yoshioka, SPring-8 Service Co., Ltd., Hyogo 679-5165, Japan

Abstract

At the X-ray Free Electron Laser (XFEL) facility, SACLA, we developed a data acquisition (DAQ) system to capture an abnormal radio frequency (RF) waveform for the diagnosis of suddenly occurring failures. The DAQ system consists of VME systems, a cache server, and a NoSQL database system, Apache Cassandra. When the VME system detects an abnormal RF waveform, it collects all the related waveforms of the same shot. The waveforms are stored in Cassandra through the cache server. Before the installation to SACLA, we ensured the performance with a prototype system. We installed the DAQ system into 34 LLRF VME systems at SACLA and six LLRF VME systems at a dedicated accelerator for the SACLA wide-band beam line. The DAQ system successfully acquired rare abnormal waveforms and contributed to the investigation of the failure analysis.

INTRODUCTION

At the X-ray Free Electron Laser (XFEL) facility, SACLA [1], driven with an 8 GeV linac, the stability of the phase and the amplitude of RF in each accelerating structure significantly affect the stability of the electron beam and the FEL intensity. It is quite helpful to collect the shot-by-shot RF data for a feedback control and the diagnosis of a failure source.

We have been using two types of data acquisition systems since the beginning of commissioning in 2011. One is a data logging system with a cycle of several seconds, which provides information for slow fluctuation, such as environmental temperatures and the flow of cooling water [2]. The other is an event-synchronized data acquisition (sync-DAQ) system that collects beam currents, beam positions, and the phase and the amplitude of the RF signals in synchronization with the beam operation cycle at the current maximum of 60 Hz [3]. Shot-by-shot data are tagged with a master trigger number to identify the beam shot to which the data belong. In addition, the sync-DAQ system collects RF waveform data every 10 min. If any fluctuations were observed in shot-by-shot data, it is important for the failure diagnosis to capture the waveform data of when the failure occurred. However, waveform collection every 10 min may not capture an abnormal RF waveform from a rare failure event that may occur only a few times a day.

ishii@spring8.or.jp

In order to collect the abnormal RF waveform data for a failure diagnosis, we developed a data acquisition system to acquire abnormal RF waveforms (abnormal WFM–DAQ system) in 2014. The DAQ system is useful for analyzing the phenomenon, because it has more information than point data, which are sampled from a part of waveform data. In this paper, we describe the installation of the abnormal WFM–DAQ system and an example of the acquired data.

ABNORMAL WFM–DAQ SYSTEM

Low-level RF Control System

The low-level RF (LLRF) control system consists of the 74 RF units of the SACLA linac. Figure 1 shows an overview of an RF unit in the LLRF control system. A trigger delay unit (TDU) board, a DAC board, three ADC boards and a CPU board are installed in a VME crate. The TDU board generates the trigger signals to the ADC and DAC boards, a klystron modulator, etc., whose timing is determined by counting with the 238-MHz master clock. The DAC board generates the pulse modulation pattern signals to the IQ modulator for controlling the RF phase and the RF amplitude. The ADC boards measure the phase and the amplitude of the RF cavity pickup signal from klystron (KLY), pulse compressor cavity, and two accelerating structures, and the high voltage (Vk) and the pulse current (Ik) of the klystron. The TDU board, the DAC board, and the ADC boards synchronize with the 238-MHz master clock.

The ADC board has four input channels. Each channel has four memory banks that can store 512 waveforms data in synchronization with the trigger signal. The data size of

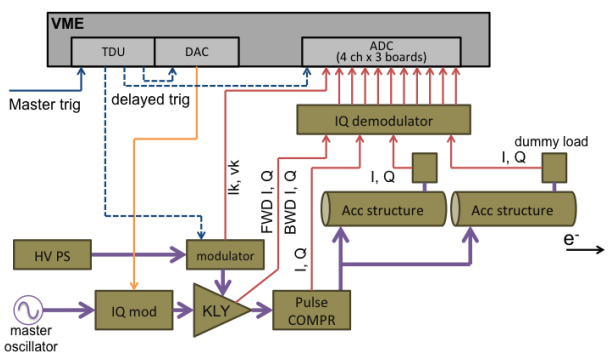


Figure 1: Overview of an RF unit in the LLRF control system.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

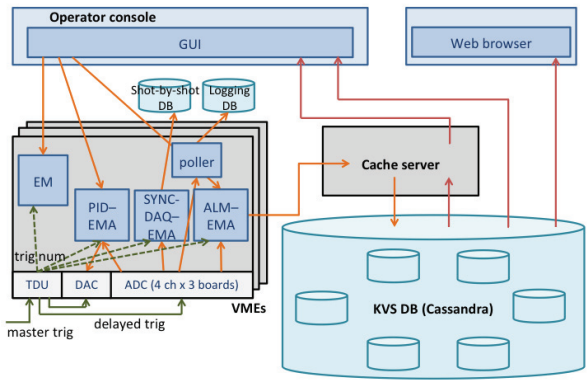


Figure 2: Diagram of abnormal waveform DAQ system.

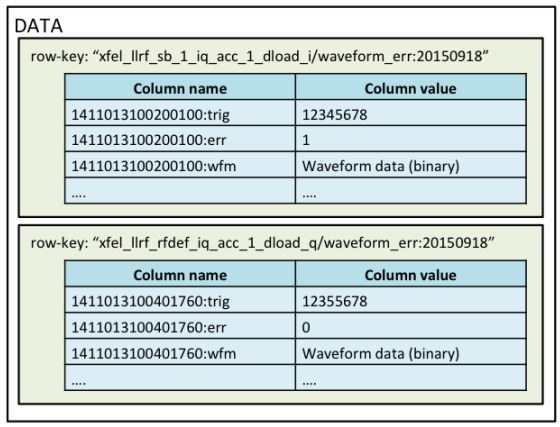


Figure 3: Cassandra's data structure.

a waveform is 4 KB for the 2048-point sampling or 16 KB for the 8192-point sampling.

Overview of the Abnormal WFM–DAQ System

The abnormal WFM–DAQ system consists of VME systems, a cache server, and Apache Cassandra, which is a key-value database system. Figure 2 shows a diagram of the abnormal WFM–DAQ system.

The ADC board generates an interrupt signal when it detects an abnormal waveform by comparing it with a reference waveform. When the difference between a sampled waveform and the reference waveform exceeds a defined allowance, the sampled waveform is categorized as an abnormal waveform. The width of the allowance can be changed from the application software [4, 5].

An abnormal WFM–DAQ process (ALM–EMA) receives the interrupt signal and acquires all the related waveforms. The waveform data are transferred to the cache server and stored in Cassandra. The stored data can be plotted in a graphical user interface (GUI) or a web browser.

Processes Running in the VME System

In the VME system, the following five processes are running:

- Equipment management process (EM) [2]
- Data logging process (poller) [2]
- Sync-DAQ process (SYNC-DAQ–EMA) in synchronization with the beam operation cycle [3]
- Feedback process (PID–EMA) for the stabilization of the phase and the amplitude of the RF cavity with 100-ms sampling intervals [6]
- ALM–EMA, which was newly developed for an abnormal WFM–DAQ system [7, 8].

From the results of the validation conducted using a prototype system, we found that it takes 1.1 ms to transfer 16 KB of waveform data from an ADC channel. To prevent blocking the VMEbus access from PID–EMA, we tuned ALM-EMA so that it takes each waveform data at a 500-ms interval. The CPU load of ALM–EMA is less than 1% on a multi-core CPU board. Therefore, ALM–EMA does not disturb other processes [9]. A multi-core VME CPU board is required for the LLRF VME system with ALM–EMA.

Database System

The requirements for a database system are high writing performance, fault tolerance, and treatment of variable-length data. Cassandra is suitable as a database for the abnormal WFM–DAQ. It is easy to increase the total throughput by adding more nodes to the system. We set up the Cassandra cluster system with six nodes and a replication factor of three. One node has two disks to acquire higher access performance by dividing the system space and the data storage space. One is a 460-GB system disk containing an operating system, Cassandra, transaction log system, and Java Virtual Machine. The other is a 3.6-TB data disk.

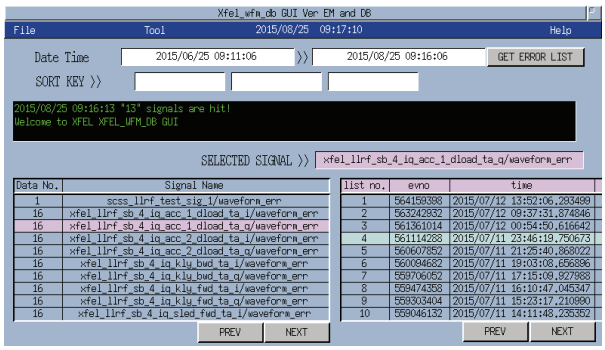
We designed a data structure to efficiently handle variable-length data such as the time series, trigger number, and error flag that indicate whether the waveform is normal or abnormal. The data structure is shown in Figure 3. One row key provides the information of one day's signal. The row-key name is formed from a signal name in addition to a date string, and the row contains collections of columns. One column consists of a name and a value. The name is formed from the timestamp in addition to a key word such as "trig," "err," or "wfm." Cassandra distributes the data to each cluster node under the hash value of the row-key. This data structure makes it possible to distribute data evenly into the Cassandra nodes [10].

The Cassandra cluster system has the concept of eventual consistency. From our measurement, the time required for guaranteeing data consistency between multiple nodes is as much as 1 s in the cluster system [11]. To prevent this inconsistency, we developed a cache server. The cache server keeps data for 2 s from the current time to complement Cassandra's eventual consistency. The cache server writes the waveforms received from ALM–EMAs to the in-memory and Cassandra in parallel. When an operational GUI requires a waveform with a signal name and a timestamp, the cache server takes the waveform from the in-memory or

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

a)



b)

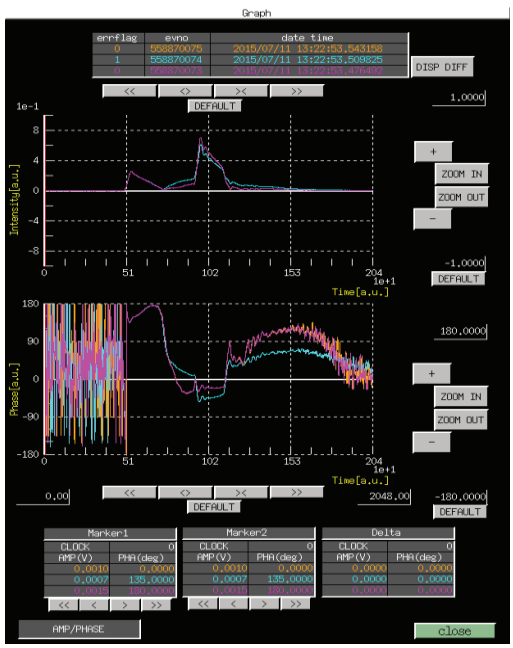


Figure 4: Management GUI for abnormal WFM–DAQ system. a) List panel of abnormal signal. b) GUI of an abnormal waveform viewer.

Cassandra in accordance with the timestamp and transfers the data to the GUI.

Graphical User Interface

We developed the GUIs for the WFM–DAQ system. Figure 4a) shows a list panel of abnormal signals. When a time period is set, the GUI acquires signals with an abnormal waveform for the set period from Cassandra and displays a list of these signals. When an abnormal signal is selected, the GUI displays the history of the abnormal signal. Upon the selecting of the event number that the operator wants to check, the GUI of an abnormal waveform viewer shown in Figure 4b) opens and plots the waveform. Further, a GUI was developed to set the parameters for ALM–EMA, such as the allowance value from the reference waveform and the repetition frequency. The GUI can issue a start/stop command. We also modified the integrated GUI for the LLRF control. The integrated GUI provides a one-click action of create/destroy/start/stop to all ALM–EMAs and monitors the status of all ALM–EMAs.

Processing Flows

The processing flows of the abnormal WFM–DAQ system are as follows:

- An operator issues a start command for abnormal detection from the operational GUI to ALM–EMA. When ALM–EMA receives the command, the process sets a normal waveform as a reference waveform with a stable RF condition to the ADC board and transfers the reference waveform to Cassandra through a cache server. ALM–EMA waits for an interrupt signal of abnormal detection from an ADC board.
- By receiving the interrupt signal, ALM–EMA disables the abnormal detection of the ADC board and takes the meta information, such as the timestamp, master trigger number, bank number in which the waveform data are written, and the address point in the bank of the ADC boards.
- ALM–EMA makes all ADC channels switch the bank to preserve the sampled waveform data. The process captures not only the abnormal waveform but also the previous and the following waveforms of the abnormal waveform on memory.
- ALM–EMA sends each waveform with the meta information to Cassandra through a cache server and receives a reply message from the cache server.
- ALM–EMA enables the abnormal detection and waits for a start command from the operation GUI.

AN EXAMPLE OF ACQUISITION

We installed the DAQ system into 34 LLRF VME systems at SACLA and six LLRF VME systems at accelerator for the SACLA wide-band beam line (SACLA–BL1 accelerator) [12].

Usually, the DAQ system monitors the RF signal from the upstream accelerating structure as the target of abnormal waveform detection. When ALM–EMA detects an abnormal waveform, it acquires 36 waveforms (3 waveforms/ch * 12 ch) at minimum.

Figure 5 shows an example of the acquired abnormal waveform at the S-band accelerator of SACLA. Figure 5a) shows the trend graph of the phase value of the RF signal in the S-band accelerating structure. More than 25 degree discrepancy was observed at 13:22. Figure 5b) shows the normal and abnormal waveforms of the cavity amplitude of the RF pickup signal in the accelerating structure, and Figure 5c) shows the normal and abnormal waveforms of the cavity phase of the RF pickup signal. Figure 5d) shows the normal and abnormal waveforms of the klystron cathode voltage. The WFM–DAQ system captured all the waveforms belonging to the specific S-band RF unit and investigated the cause. In this case, we speculated that a high-voltage discharge in the klystron modulator caused the high-voltage deficit and the considerable change in the amplitude and the phase of the

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

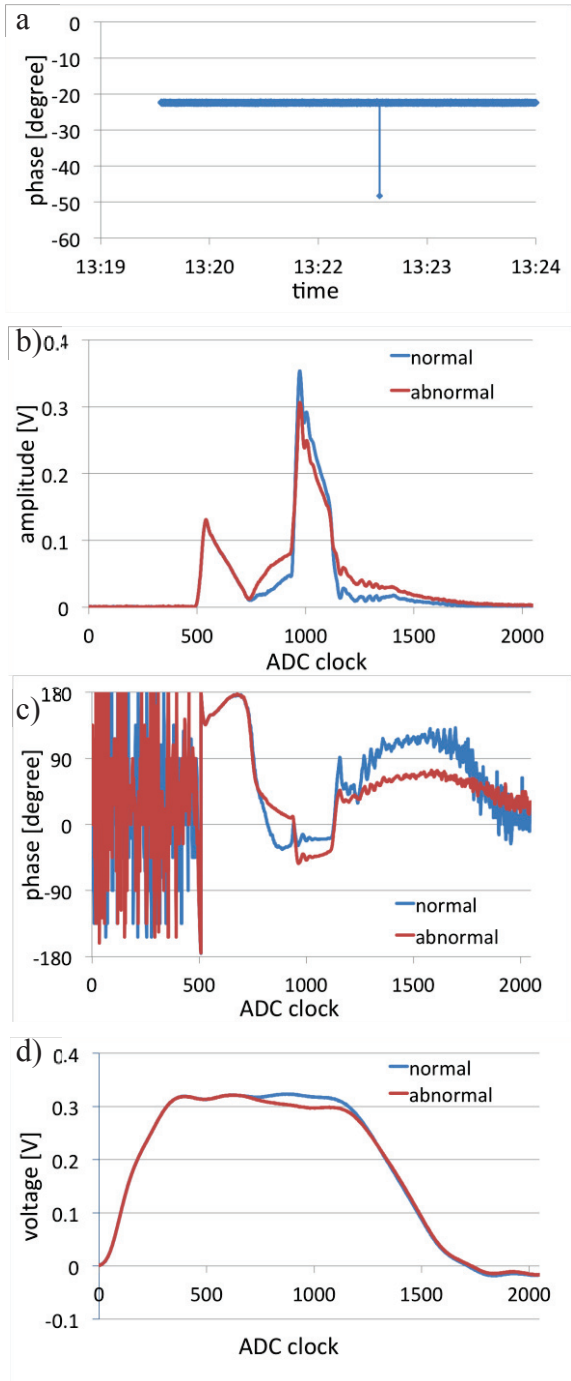


Figure 5: a) Trend graph of the cavity phase in the S-band accelerating structure. b) Normal and abnormal waveforms of the cavity amplitude of the RF pickup signal. c) Normal and abnormal waveforms of the cavity phase of the RF pickup signal. d) Normal and abnormal waveforms of the klystron cathode voltage.

RF power. Since this abnormal waveform frequently occurred in a specific RF unit, we replaced the modulator during the summer shutdown period. Currently, the failure has not occurred. As shown in this example, the WFM DAQ system can rapidly identify the cause of failure.

SUMMARY

We developed a data acquisition system for abnormal RF waveforms. This system captures a suddenly occurring abnormal RF waveform and stores all the related waveform data in Cassandra. We have been successfully operating this system in 34 LLRF VME systems at SACLA and six LLRF VME systems at the SACLA–BL1 accelerator. We could determine the failure source in a specific RF unit. In the future, we intend to acquire waveform data from 74 LLRF VME systems. The collected data are helpful in improving the reliability of the accelerators.

REFERENCES

[1] T. Ishikawa, et al., "A compact X-ray free-electron laser emitting in the sub-angstrom region", Nat. Photonics 6, 540-544 (2012)

[2] R. Tanaka, et al., "Inauguration of the XFEL facility, SACLA, in Spring-8", Proc. of ICALEPCS2011, p.585-588, Grenoble, France, (2011)

[3] M. Yamaga, et al., "Event-Synchronized data-acquisition system for SPring-8 XFEL", Proc. of ICALEPCS2009, p.69-71, Kobe, Japan, (2009)

[4] T. Fukui, et al., "A development of high-speed A/D and D/A VME boards for a low level RF system of SCSS", Proc. of ICALEPCS2005, Geneva, Switzerland, (2005)

[5] Y. Otake, et al., "SCSS RF control toward 5712 MHz phase accuracy of one degree", Proc. of APAC2007, p.634-636, Indore, India, (2007)

[6] H. Maesaka, et al., "Recent progress of the RF and timing system of XFE/SPring-8", Proc. of ICALEPCS2009, p.85-89, Kobe, Japan, (2009)

[7] T. Ohshima, et al., "Capture of abnormal RF waveform at SACLA", Proc. of the 11th Annual Meeting of the Particle Accelerator Society of Japan, Aomori, Japan, (2014)

[8] M. Yoshioka, et al., "A data acquisition framework for the abnormal RF waveform at SACLA", Proc. of the 11th Annual Meeting of the Particle Accelerator Society of Japan, Aomori, Japan, (2014)

[9] M. Ishii, et al., "A prototype data acquisition system of abnormal RF waveform at SACLA", Proc. of PCAPAC2014, Karlsruhe, Germany, (2014)

[10] M. Kago, et al., "Development of a scalable and flexible data logging system using NoSQL databases", ICALEPCS2013, p.533-535, San Francisco, USA, (2013)

[11] T. Maruyama, et al., "Development of web services framework for distributed database", Proc. of the 11th Annual Meeting of the Particle Accelerator Society of Japan, Aomori, Japan, (2014)

[12] N. Hosoda, et al., "A control system for a dedicated accelerator for SALCA wide-band beam line", ICALEPCS2015, Melbourne, Australia, (2015)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DRIVERS AND SOFTWARE FOR MicroTCA.4*

M. Killenberg†, M. Heuer, M. Hierholzer, L. Petrosyan, C. Schmidt, N. Shehzad,
G. Varghese, M. Viti, DESY, Hamburg, Germany
S. Marsching, aquenos GmbH, Baden-Baden, Germany
M. Mehle, T. Sušnik, K. Žagar, Cosylab d.d., Ljubljana, Slovenia
A. Piotrowski, FastLogic Sp. z o.o., Łódź, Poland
T. Kozak, P. Prędki, J. Wychowaniak, Łódź University of Technology, Łódź, Poland

Abstract

The MicroTCA.4 crate standard provides a powerful electronic platform for digital and analogue signal processing. Besides excellent hardware modularity, it is the software reliability and flexibility as well as the easy integration into existing software infrastructures that will drive the widespread adoption of the new standard. The DESY MicroTCA.4 User Tool Kit (MTCA4U) comprises three main components: A Linux device driver, a C++ API for accessing the MicroTCA.4 devices and a control system interface layer. The main focus of the tool kit is flexibility to enable fast development. The universal, expandable PCI Express driver and a register mapping library allow out of the box operation of all MicroTCA.4 devices which are running firmware developed with the DESY board support package. The tool kit has recently been extended with features like command line tools and language bindings to Python and Matlab.

INTRODUCTION

The MicroTCA.4 crate standard [1, 2] provides a platform for digital and analogue data processing in one crate. It is geared towards data acquisition and control applications, providing a backplane with high-speed point to point serial links, common high-speed data buses as well as clock and trigger lines. In typical control applications, large amounts of data have to be digitised and processed in real-time on the front end CPU of the MicroTCA.4 crate.

MTCA4U—The DESY MicroTCA.4 User Tool Kit

The main goal of the DESY MicroTCA.4 User Tool Kit (MTCA4U) [3] is to provide a library which allows efficient, yet easy to use access to the MicroTCA.4 hardware in C++. In addition, it features an adapter layer to facilitate interfacing to control system and middleware software. The design layout of the tool kit is depicted in Fig. 1.

LINUX KERNEL MODULE

The Linux kernel module (driver) provides access to the MicroTCA.4 devices via the PCI Express bus. As the basic access to the PCI Express address space is not device dependent, we follow the concept of a universal driver for all MicroTCA.4 boards. The kernel module uses the Linux Device Driver Model which allows module stacking, so that the driver can be split into two layers: A universal part provides all common structures and implements access to the PCI Express I/O address space. The device specific part implements only firmware-dependent features like Direct Memory Access (DMA), and uses all basic functionality of the universal part. For all devices developed at DESY the firmware will provide a standard register set and the same DMA mechanism, which permits to use a common driver for most boards. For devices from other vendors the universal part enables out-of-the-box access to the basic features, which can be complemented by writing a driver module based on the universal driver part. Like this, the interface in MTCA4U does not change and the new device is easy to integrate into existing software.

The MircoTCA.4 platform allows hot-plugging of all components, which means the PCI Express components can appear and disappear at run time. Usually this is not the case for PCI Express hardware, and not all drivers are prepared to handle this situation. For the drivers provided by MTCA4U special attention has been paid to make the driver hot-plug capable and allow safe operation at all times.

THE C++ DEVICE API

The core piece of MTCA4U is the C++ device access library, which provides a high level interface to the hardware. Its main component is the `Device` class (see Fig. 2), which provides a convenient interface to access registers in the hardware I/O address space. The `Device` class is not accessing the hardware directly but uses the abstract `DeviceBackend` interface, which has several implementations. The `PcieBackend` is accessing the PCI Express hardware through the Linux kernel module, abstracting implementation details like IOCTL sequences from the user. A `DummyBackend` can provide the same set of registers, simulated in the RAM of the CPU module. Deriving from this class, the user can simulate firmware functionality of a specific device, which simplifies unit and integration testing. The latest addition is the `RebotDevice`, which implements the "Register-based over TCP" protocol (ReboT). This allows access to network devices, either via the backplane of the MicroTCA crate or through an external network, broadening the scope of the MicroTCA.4 User Tool Kit also to applications outside the MicroTCA form factor.

* This work is supported by the Helmholtz Validation Fund HVF-0016 "MTCA.4 for Industry".
† martin.killenberg@desy.de

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

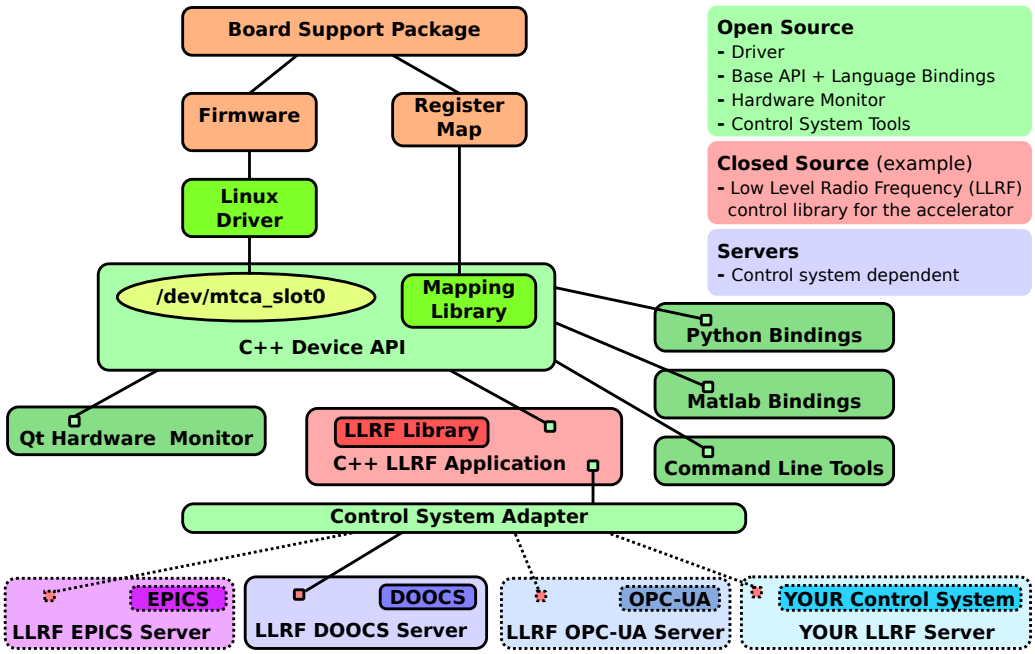


Figure 1: The design concept of the MicroTCA.4 User Tool Kit MTCA4U.

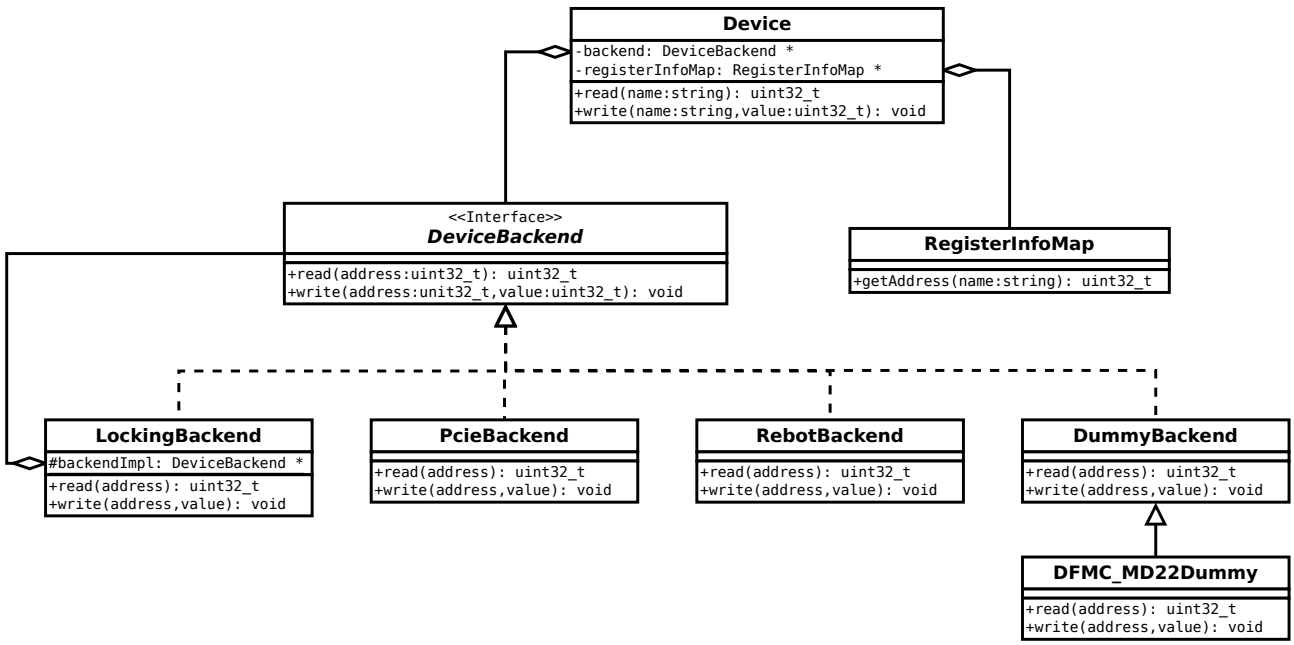


Figure 2: The MTCA4U device structure.

The Back-End Factory

Software using the MTCA4U device to access register based hardware does not have to know all back-ends at compile time. A plugin mechanism allows registering new back-end types to a factory at run time. It uses a configuration file to define shared objects which are loaded at the start of the programme. Like this, the user can easily extend the portfolio of back-ends, be it a new hardware protocol or a custom dummy device. As the loading of the back-end is only done at run time, a library does not have to be modified to run with a mock, which makes this a strong tool for software tests and for development if access to hardware is scarce for software developers.

Register Name Mapping

Another main feature of the C++ library is the register name mapping. With evolving firmware, the address of a register can change in the I/O address space. To make the user code robust against these changes, the registers can be accessed by their name instead of using the address directly, which also improves the code readability. The required mapping file is automatically generated by the Board Support

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

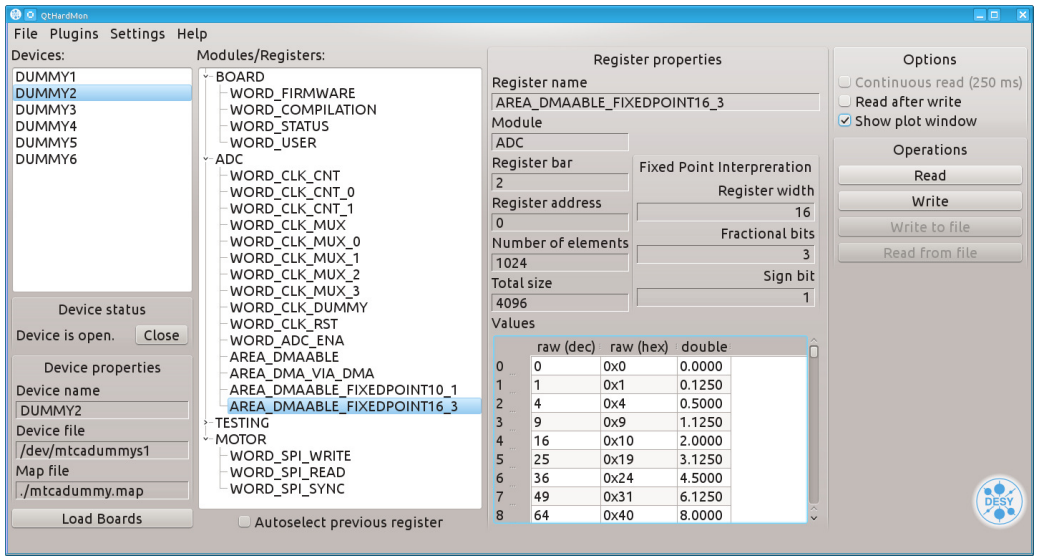


Figure 3: A screen shot of the Qt Hardware Monitor.

Package together with the firmware. Performance overhead due to repeated table look-up is avoided by the use of register accessor objects, which cache the address and provide fast access to the hardware.

Numeric Conversions

Unlike modern CPUs, hardware like FPGAs and micro controllers does not always have a floating point computing unit. Usually the calculation is done in a fixed point format and the interpretation of the raw data words depends on the firmware. However, the CPU which is talking to the hardware probably will use floating point arithmetic and has to convert to the right format before sending a data word. The MTCA4U mapping file does not only contain information about the register's size and address, but also about the fixed point interpretation in the firmware. The register accessors automatically convert the incoming floating point values to the right data format, which speeds up development in C++ and makes tools like the graphical user interface more versatile.

GRAPHICAL USER INTERFACE

The mapping file contains information of all the registers implemented in the firmware, which allows displaying this information in a graphical user interface (GUI). The Qt Hardware Monitor lists all registers and their properties, and permits the user to interactively display and modify their content, including automatic fixed point conversion. As the mapping file is automatically generated together with the firmware, this tool can be used for debugging and prototyping immediately after the firmware has been deployed. The hardware monitor is written using Qt [4], an open source, cross-platform user interface framework which is available on all Linux platforms. A screen shot of the Qt Hardware Monitor is shown in Fig. 3.

LANGUAGE BINDINGS

Scripting languages are ideal for prototyping and hardware testing because they provide direct interaction without having to compile code. For this reason, MTCA4U provides language bindings to Matlab and Python, as well as Linux command line tools. These allow writing configuration scripts for hardware with just a few lines of code. The output format of the command line tool is designed so that it can be easily parsed by a script. This allows accessing the PCI Express bus of a remote crate through an ssh tunnel, and evaluate the output in an automated way.

CONTROL SYSTEM ADAPTER

MicroTCA.4 allows developers to implement sophisticated control applications, which have to interface with the facility's SCADA system[1]. Usually this means a tight coupling of the application and the control system, which makes it difficult to port applications between different control systems. MTCA4U provides an interface layer to minimise these couplings and improve the re-usability of complex control applications, which are expensive to develop and maintain. This decoupling also removes the dependency on control system locks, which facilitates the implementation of real-time capable controls. A more detailed description of the MTCA4U control system adapter can be found in [5].

OUTLOOK

To provide highly reliable software for user facilities with minimal downtime, software testing is an essential part of quality assurance. To facilitate the development of functional mock-ups and software simulations, we are currently working on a "virtual lab" framework. Using the `DummyBackend` as a starting point, it will provide direct accessors to the simulated address space, a model of data sources and sinks

[1] Supervisory Control and Data Acquisition system

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

to connect different building blocks, and a virtual timer framework. The latter is particularly useful to study race conditions in multi-threaded applications or the interplay of hardware and software.

The virtual lab is currently in its early prototyping phase.

CONCLUSIONS

The DESY MicroTCA.4 User Tool Kit (MTCA4U) is a C++ library which allows convenient access to hardware with an extensible register based interface. Starting from PCI Express, which is used inside a MicroTCA.4 crate, the introduction of new, network based protocols extends its reach beyond a single crate and even MircoTCA itself. The tool kit comprises a C++ API with register name mapping and automatic type conversion, with bindings to widely used scripting tools like Matlab and Python. A graphical user interface is available for fast prototyping and firmware development, allowing direct access to the hardware without writing a single line of code. A control system adapter allows the application code to be independent from the actual control system in use. This makes the business logic portable between control systems with minimal effort and allows a wider field of application for software written using MTCA4U.

MTCA4U is published under the GNU General Public License and available on DESY's subversion server [3].

REFERENCES

[1] PICMG®, "Micro Telecommunications Computing Architecture, MicroTCA.0 R1.0" (2006).

[2] PICMG®, "MicroTCA® Enhancements for Rear I/O and Precision Timing, MicroTCA.4 R1.0" (2011/2012).

[3] MTCA4U—The DESY MicroTCA.4 User Tool Kit, Subversion Repository https://svnsrv.desy.de/public/mtca4u

[4] The Qt Project, http://qt-project.org/

[5] M. Killenberg et al., "Integrating control applications into different control systems", TUD3O05, *These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SERVICE ASSET AND CONFIGURATION MANAGEMENT IN ALICE DETECTOR CONTROL SYSTEM

M. Lechman[1, 2], A. Augustinus[1], P. M. Bond[1], P. Chochula[1], O. Pinazza[1, 2], A. Kurepin[1, 3]
[1]CERN, Geneva, Switzerland
[2]IP-SAS, Bratislava, Slovakia
[3]INFN-Bologna, Bologna, Italy
[4]RAS/INR, Moscow, Russia

Abstract

ALICE (A Large Ion Collider Experiment) is one of the big LHC (Large Hadron Collider) detectors at CERN. It is composed of 19 sub-detectors constructed by different institutes participating in the project. Each of these subsystems has a dedicated control system based on the commercial SCADA package "WinCC Open Architecture" and numerous other software and hardware components delivered by external vendors. The task of the central controls coordination team is to supervise integration, to provide shared services (e.g. database, gas monitoring, safety systems) and to manage the complex infrastructure (including over 1200 network devices and 270 VME and power supply crates) that is used by over 100 developers around the world. Due to the scale of the control system, it is essential to ensure that reliable and accurate information about all the components – required to deliver these services along with relationship between the assets – is properly stored and controlled. In this paper we will present the techniques and tools that were implemented to achieve this goal, together with experience gained from their use and plans for their improvement.

INTRODUCTION

A Large Ion Collider Experiment (ALICE) [1] is one of the big Large Hadron Collider (LHC) detectors at CERN that is optimized to study the physics of the quark–gluon plasma in nucleus–nucleus collisions. The experiment collaboration consists of 1550 physicists, engineers and technicians from 37 countries.

ALICE is composed of 19 sub-detectors constructed by different institutes participating in the project. Each of these subsystems has a dedicated control software based on the commercial SCADA package "WinCC Open Architecture" [2] and numerous other software and hardware components delivered by external vendors.

The ALICE Controls Coordination (ACC) team supervises the integration of all of these applications into one global Detector Control System (DCS) [3] that allows one to monitor and operate the whole experiment from a central operator console. The group is also responsible for providing shared services (like database, environment monitoring) and for managing complex infrastructure (including over 1200 network devices and 270 VME and power supply crates) that is used by over 100 developers around the world. The DCS includes as well interfaces to external systems maintained by other groups at CERN. Examples of such systems are beam interlock, safety mechanisms or software for acquisition and processing of physics data. The full context of the ALICE DCS is illustrated in Fig. 1.

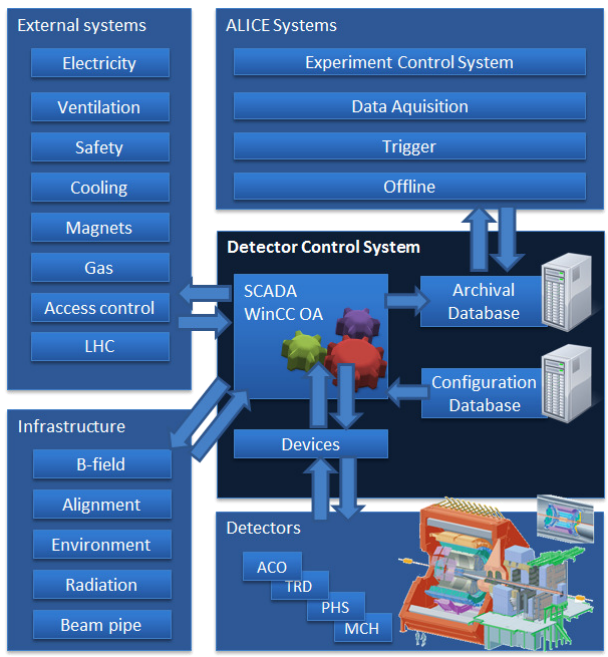


Figure 1: Context of ALICE Detector Control System.

The on-call expert service and training of DCS operators, who supervise the detector from the central console in Run Control Centre, are another tasks of the ACC group.

The team also participates in JCOP (Joint Controls Project) [4] – a collaboration of different experiments at CERN created to share the development effort needed for preparing software components used in their controls systems.

SERVICES AND THEIR ASSETS IN DCS

The aforementioned services require numerous software and hardware assets. Due to the scale of the control system, it is essential to ensure that reliable and accurate information about all these components along with the relationship between them is properly stored and controlled.

To meet these requirements, the central DCS team explores the possibilities to adapt best practises from ITIL (IT Infrastructure Library) [5] to manage its commitments, user groups and resources.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The ITIL has already been used at CERN to create the common Service Desk for the whole organization and to define incident and the request fulfilment processes [6].

The information about the components that are used in the DCS is recorded in 3 main Configuration Management Databases (CMDBs):

- LAN Database stores information about all the network devices at CERN.
- Detector Construction Database (DCDB) [7] is a universal repository for parts that exist in the ALICE detector. This storage allows creation of custom types of objects (together with definitions of attributes) and their hierarchies via a generic data model.
- JCOP System Information Database [8][9] stores data about WinCC OA control applications along with the information about the machines they are running on and their supporting software (like e.g. OPC servers).

Following the ITIL recommendations, the Configuration Management System (CMS) has been established on top of these CMDBs to provide a common interface to combined data about each of the component.

Another requirement for the tool has been that it copes with a continuously evolving environment. The main sources of changes in the ALICE DCS environment are:

- installation of additional detector modules,
- exchanges and upgrades of existing equipment,
- evolution of software (e.g. new operating systems),
- new dependencies between already existing components (e.g. additional software safety interlocks);

The services the ACC team is delivering have been identified and recorded in the CMS in a service portfolio together with their related users. The hierarchy of configuration items in the CMS is presented on Fig. 2.

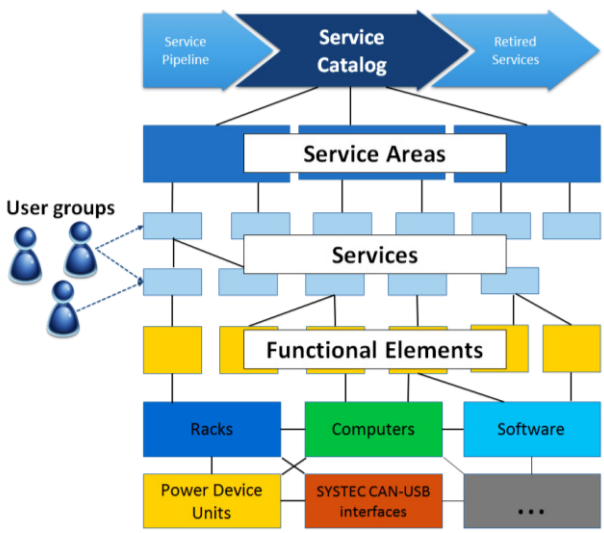


Figure 2: Types of supported configuration items and their relations.

The service portfolio includes a place to store services that are currently available (*service catalog*) as well as offering possibility to record retired services and new but not yet published services (*service pipeline*).

The concept of abstract "functional element" was introduced for the description of the logical layer between services and instance of hardware. Such objects corresponds for example to IP addresses that are assigned to network devices and are used in code and configuration files of applications. Thanks to that, these software components don't need to be modified in case of replacement of the hardware - only a reference *network device-alias* needs to be updated in the LAN Database. Functional elements also allows organizing different types of components into logical groups.

The CMS also permits the recording of dependencies between the aforementioned components and software and hardware configuration items as directed graph structure.

ARCHITECTURE

The CMS has been developed in the Oracle APEX environment and has been integrated with CERN SSO (Single Sign On), LDAP and E-Groups mechanisms to provide user authentication and authorization.

The tool has been also connected to the System Information database via the mechanism of Oracle database links and integrated via web services with the LAN Database. Mechanisms of Oracle Collections and XQuery are used for processing the data from this latest source.

The information about the Service Catalog is stored in a generic data model of DCDB, which allows for simple access to construction data from this storage.

The CMS can also be used to export data needed for proper labelling of hardware including unique part identifiers generated by the DCDB [10].

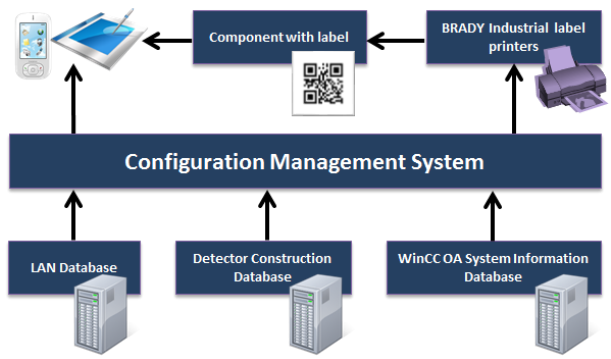


Figure 3: Configuration Management System – information flow.

RELATIONS BETWEEN COMPONENTS

Migration of experts is a factor that has to be taken into account in the case of long-term projects like ALICE. The expert knowledge about an individual sub-detector is maintained by the various institutes that developed these subsystems. It may happen that the transfer procedure does not explain the overall context for all the components. This is a motivation to centrally manage those dependencies. Similar initiatives have already been taken to monitor and authorize the connections between WinCC projects [11].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The CMS allows the tracking the relations of every component up to the level of services and user group both in tabular and graphical form. It is possible also to explore the dependencies down in the hierarchy to get the information about all the sub-components that are needed for the proper functioning of particular object.

The diagrams are generated in MS Visio so that a user can automatically lay out shapes and connectors neatly and apply further formatting easily. Every graphical object has a link to the details of the represented component from the CMS for quick access.

The diagrams can be used especially for change impact analysis and during resolving incidents and problems.

A sample diagram for the impact analysis for scenario of a broken power supply unit is presented on Fig. 4.

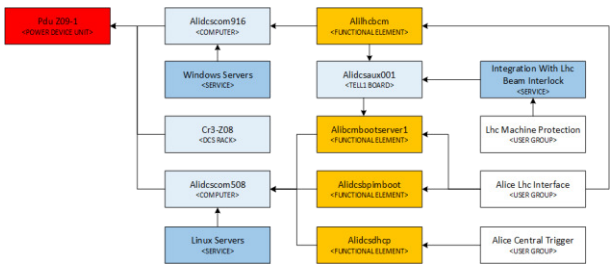


Figure 4: Automatic generation of diagrams showing dependencies between configuration items for impact analysis - sample result.

CONCLUSIONS

The complexity of the ALICE DCS required the implementation of dedicated software for the asset management. The tool evolved from a simple web application based on spreadsheets to a comprehensive solution integrated with all the sources of information about the infrastructure and its dependencies with services and particular user groups.

Using Oracle APEX technology, existing CERN IT services and a generic data model it was possible to deliver a wide range of functionalities to end users in a relatively short period of time and to minimize effort of the ACC group for maintenance and administration of the application.

The Configuration Management System has been so far successfully used in practice for managing control servers and their software. New modules, like service portfolio and racks management, are available and the data is mostly loaded.

The main challenge is to maintain the discipline among the users to keep the data up to date, especially in case of incidents requiring urgent on-call expert intervention. This effort, however, should pay for itself as the updated information will be supporting the experts in making right decisions in such situations.

It is envisaged in the future to extend the functionality of the Configuration Management System on more types of components and to invite other ALICE groups and members of JCOP project to assess its usefulness in their projects.

REFERENCES

[1] The ALICE Collaboration et al., “The ALICE experiment at the CERN LHC”, JINST 3, S08002, 2008.

[2] WinCC Open Architecture website http://www.etm.at

[3] P. Chochula, L. Jirden, A. Augustinus, G. de Cataldo, C. Torcato, P. Rosinsky, L. Wallet, M. Boccioli, L. Cardoso, “The ALICE Detector Control System”, IEEE Transactions on Nuclear Science, volume 57, 472 - 478, 2010.

[4] O. Holme et al., “The JCOP Framework”, ICALEPCS 2005, Geneva, Switzerland.

[5] ITIL website: www.itil-officialsite.com

[6] Z. Toteva et al., "Service management at CERN with Service-Now", CHEP 2012, New York, USA.

[7] W.S. Peryt et al., “Detector construction database for ALICE experiment”, CHEP 2003, La Jolla, California, 2003.

[8] M. Gonzalez, K. Joshi, F. Varela, “The System Overview Tool of the Joint Controls Project (JCOP) Framework”, ICALEPCS 2007, Knoxville, USA.

[9] F. Varela, “Software management of the LHC Detector Control Systems”, ICALEPCS, 2007, Knoxville, Tennessee, USA.

[10] P.Chochula, L.Betev, “Naming and Numbering Convention for ALICE detector part identification – Generic scheme”, ALICE-INT-2003-039, Available on http://edms.cern.ch

[11] O. Pinazza et al., “Managing Information Flow in ALICE”, ICALPECS 2011, Grenoble, France.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DATABASE APPLICATIONS DEVELOPMENT OF THE TPS CONTROL SYSTEM

Y. S. Cheng, Jenny Chen, C. Y. Liao, C. H. Huang, P. C. Chiu, Y. T. Chang, K. T. Hsu
National Synchrotron Radiation Research Center, Hsinchu 30076, Taiwan

Abstract

The control system had been established for the new 3 GeV synchrotron light source (Taiwan Photon Source, TPS) which was successful to commission at December 2014. Various control system platforms with the EPICS framework had been implemented and commissioned. The relational database (RDB) has been set up for some of the TPS control system applications used. The EPICS data archive systems are necessary to be built to record various machine parameters and status information into the RDB for long time logging. The specific applications have been developed to analyze the archived data which retrieved from the RDB. One EPICS alarm system is necessary to be set up to monitor sub-system status and record detail information into the RDB if the problem happened. Some Web-based applications with RDB have been gradually created to show the TPS machine status related information. The efforts are described at this paper.

INTRODUCTION

The TPS is a new high brightness synchrotron light source which constructed at the National Synchrotron Radiation Research Center (NSRRC) in Taiwan. It consists of a 150 MeV electron linac, a booster synchrotron, a 3 GeV storage ring, and experimental beam lines. Civil construction had started from February 2010. The construction works had been finished in half of 2013. Accelerator system installation and integration had been started in later 2013. The control system environment was ready in half of 2014 to support final subsystem integration test and commissioning without beam. Commissioning with beam was successful at December 2014.

The EPICS (Experimental Physics and Industrial Control System) [1] is a set of open source software, libraries and applications developed collaboratively and used to create distributed soft real-time control systems for scientific instruments such as the particle accelerators. Many facilities have good practical experiences for the EPICS and adopt it as the accelerator control systems. Many resources and supports are available as well as numerous applications have been developed.

The TPS Control system is also based on the EPICS framework [2]. The EPICS toolkit provides standard tools for display creation, archiving, alarm handling, etc. The big success of EPICS is based on the definition of a standard IOC (Input Output Controller) structure together with an extensive library of driver software for a wide range of I/O cards. The EPICS toolkits which have various functionalities are employed to monitor and to control accelerator system.

The Java-based EPICS data archive system was set up for the TPS project. The Relational Database (RDB) has been used as the data storage mechanism for recording historic EPICS data [3]. The various operation interfaces for browsing archived data have been developed. Taking the performance and redundancy into considerations, the storage servers and database table structures are tuned up relatively.

The Java-based alarm system has been developed as the alarm handler for the TPS control system. A distributed alarm system monitors the alarms in a control system and helps operators to make right decisions and actions in the shortest time. Moreover the historic alarm messages are logged into the RDB system, and the clients can use Java-based graphical operation interface to query and effectively check which system is failure.

Web-based machine status broadcasting is convenient to use with Web browser. The PHP webpage has been created for the TPS machine status broadcasting. The trend data are retrieved from the archive database, and also show variations on the Web-based machine status broadcasting.

The efforts for implementing are summarized as followings.

TPS EPICS DATA ARCHIVE SYSTEM

Software Environment

For the installation and commissioning phases, one EPICS data archive system is necessary to be developed to record various accelerator parameters and machine status information for long time observation. The archive system of CS-Studio [4-5] had been implemented completely to be used as the TPS EPICS data archive system in earlier 2014. The archive engines take PVs data from EPICS IOCs via channel access, and stores them into the data storage.

The CS-Studio based archive system can store data in a relational database (RDB). It applies JDBC libraries [6] in some databases, such as MySQL, Oracle, PostgreSQL, and the archive system includes example database definition (DBD) files to create the required tables for the database dialects. The EnterpriseDB [7] (PostgreSQL) RDB have been used for the EPICS data archive system of TPS project. The PostgreSQL RDB is a good compromise and bigger table sizes. Both the PVs historic data and the Archive Engine configuration are stored in the same relational database. The engine configurations are imported from an XML file format into the database.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The software architecture of TPS EPICS data archive system is shown as Fig. 1. The storage system is separated into two databases which named RDB1 and RDB2. The RDB1 is mainly used to store into the EPICS historic data from EPICS IOCs via PV channel access. The RDB2 is synchronized to be a duplicate of the RDB1. If the RDB1 causes service halt, the RDB2 will do replication mode for restoring to the RDB1. The RDB2 provides higher reading priority to enhance transmission bandwidth for extracting queried historic archived data.

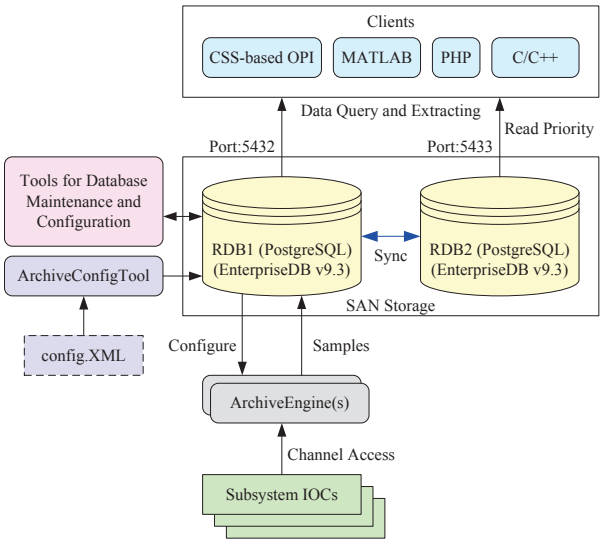


Figure 1: Software architecture of the TPS EPICS data archive system.

Database Tuning Up for the TPS Archive System

To easily maintain and manage the archived data in the database, the sample data table has been separated into a lot of sub-tables by days. The new EPICS sampled data have been inserted into the specific sub-table which is according to the day in the year. If the older samples need to be deleted, the day-named table can be indicated and deleted. Each created sub-tables have been added the table index to enhance performance for data retrieving. Actually the test is that one week historic data (one channel-sample per second) need about 10~20 seconds to be queried out, extracting and plotting. Otherwise, there is about 1~2 minutes at least for retrieving one week historic data without creating the table indexes. It is effective to create table indexes to enhance performance for retrieval.

CS-Studio Based Operation Interface

The clients can use the specific toolkits to retrieve the historic archived data from the PostgreSQL RDB. As a result, several operation interfaces have been created for difference purposes.

The Data Browser is a generic CS-Studio toolkit that combines Strip Tool and Archive Viewer functionality. It can display live samples as well as archived data in a plot, or export the data to files. Based on plugins for archive data sources, it can currently interface to the Channel Archiver and the EPICS Archive record. Each PV may have multiple data sources: for example, the Data Browser can merge samples from an archive record for recent history with those from a midterm and long-term archive. It is convenient that the CS-Studio based OPI can be used in difference operation systems, such as Windows and Linux-based. For the TPS commissioning and operation phases, the graphical interface of machine status is necessary to be created for observing the historic variations for long time, such as beam current of booster and storage ring. The CS-Studio based graphical interface is shown at Fig. 2.

Figure 2: GUI of CSS-based archive data browser.

To easily observe the temperature and vacuum variations during baking process of in-vacuum insertion device, the specific graphical monitor interface was necessary to be created by use of the CS-Studio as shown in Fig. 3. This interface was displayed the latest 30 minutes historic temperature and vacuum data trend which retrieved from the RDB of archive system.

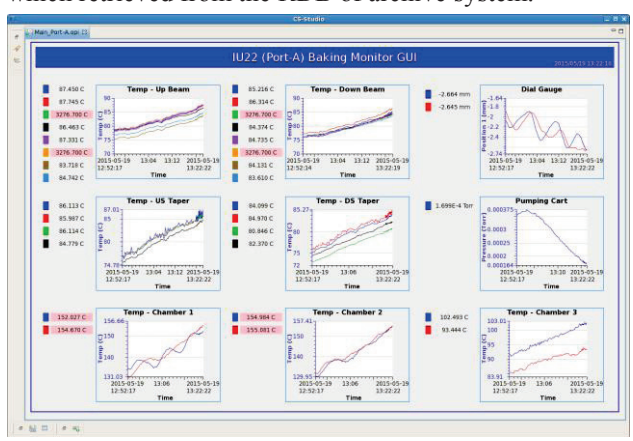


Figure 3: Archive data browsing GUI for ID baking.

Miscellaneous Operation Interfaces

To special purposes, the specific toolkit, such as MATLAB [8] (with JDBC connection), is used to create the interface to retrieve the EPICS archived data from the RDB.

To investigate the beam loss and its distribution, a RedFET (radiation-sensing field-effect transistor) reader was implemented with an EPICS IOC. All of the RedFET threshold voltage values based on the EPICS PVs channel access can be recorded into the PostgreSQL RDB archive server for further off-line data processing. The archived

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

data can be retrieved by use of a form of graphical representation of the CS-Studio based data browser to observe the trend. The MATLAB toolkit has been used to analyze the RadFET threshold voltage archived data which retrieved from the RDB archive system directly [9]. The MATLAB toolkit for analyzing the RadFET threshold voltage archived data is shown as Fig. 4.

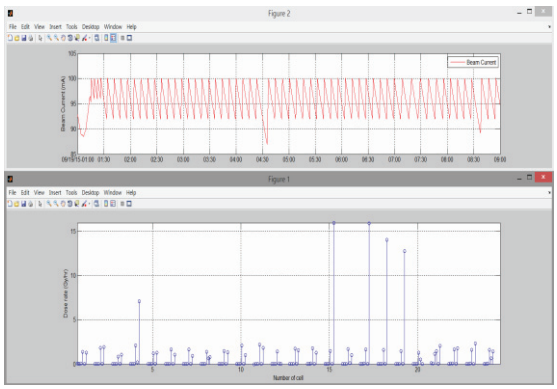


Figure 4: MATLAB interface for analyzing archived data.

TPS ALARM SYSTEM

The "BEAST" (Best Ever Alarm System Toolkit) of CS-Studio with the MySQL RDB has been adopted as the alarm handler for the TPS as shown in Fig. 5. A distributed alarm system monitors the alarms in a control system and helps operators to make right decisions and actions in the shortest time. In the CS-Studio based alarm system, each alarm is supposed to be meaningful, requiring an operator action. An alarm is no status display that operators may ignore. Each alarm requires an operator to react because the control system cannot automatically resolve an issue.

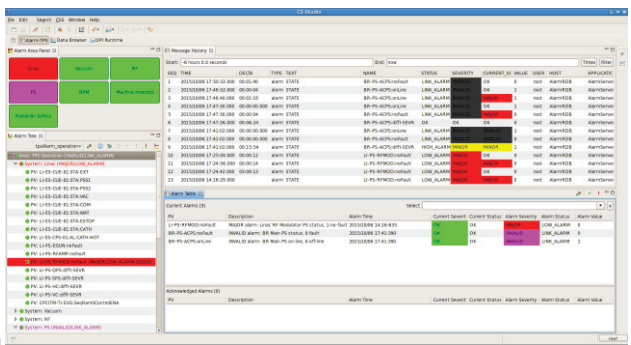


Figure 5: CS-Studio based graphical interface of TPS alarm system.

WEB-BASED APPLICATIONS

The webpage has been created by using the PHP and AJAX (Asynchronous JavaScript and XML) technique for the TPS machine status broadcasting. The client users can use Web browsers of PCs or smartphones to watch the machine status immediately. The trend graph is one of necessary components to show the historic variations of beam current and beam lifetime on the TPS machine status webpage. The trend data of several hours are retrieved from the PostgreSQL database of TPS archive system by the PHP program [10]. The TPS Web-based machine status broadcasting with trend graphs is shows as Fig. 6.

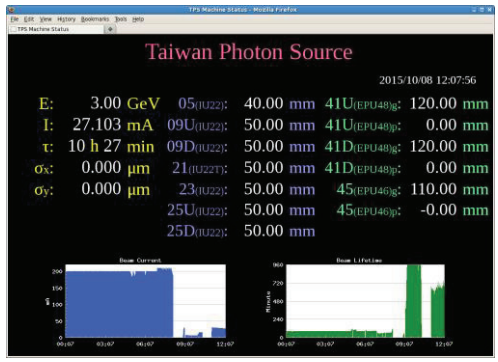


Figure 6: TPS Web-based machine status broadcasting.

The Olog [11] solution has been selected for the TPS electronic logbook. The progress of commissioning and operation information has been recorded into the MySQL RDB via the Olog by the commissioning team and operators. The Olog supports to use on the Web browser and CS-Studio, and also supports search function and print function to copy data into the logbook for logging.

SUMMARY

The RDB has been adopted as the data storage for the TPS archive system. The archived data are retrieved by use of the CS-Studio based interface. The archived data are also retrieved by specific toolkits for special purposes. The RDB alarm system has been built to help operators to make right decisions and actions in the shortest time. More database applications are in developing during the commissioning and operation phases.

REFERENCES

[1] EPICS website: http://www.aps.anl.gov/epics/
[2] C. Y. Liao, et al., "Commissioning of the TPS Control System", FRB3O01, these proceedings, ICALEPCS'15, Melbourne, Australia (2015).
[3] Y. S. Cheng, et al., "Implementation of the EPICS Data Archive System for the TPS Project", THPEA049, Proceedings of IPAC'13, Shanghai, China (2013).
[4] K. Kasemir, "Control System Studio Applications," MOPB03, Proceedings of ICALEPCS'07, Knoxville, Tennessee, USA (2007).
[5] K. Kasemir and G. Carcassi, Control System Studio Guide, 2012.
[6] JDBC website: http://docs.oracle.com/javase/6/docs/technotes/guides/jdbc/
[7] EnterpriseDB website: http://www.enterprisedb.com
[8] MATLAB with JDBC connection website: http://www.mathworks.com/help/database/ug/database.html

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

[9] C. H. Huang, et al., "Beam Loss Study of TLS Using RadFETs", MOPTY072, Proceedings of the IPAC'15, Richmond, USA (2015).
[10] PHP with PostgreSQL database website: http://php.net/manual/en/book.pgsql.php
[11] Olog website: https://github.com/Olog/logbook

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A REDUNDANT EPICS CONTROL SYSTEM BASED ON PROFINET*

Z.Huang, Y. Song, K. Wan, C. Li, G. Liu†, NSRL, USTC, Hefei, Anhui 230029, China

Abstract

This paper will demonstrate a redundant EPICS control system based on PROFIENT. The control system consists of 4 levels: the EPICS IOC, the PROFINET IO controller, the PROFINET media and the PROFINET IO device. Redundancy at each level is independent of redundancy at each other level in order to achieve highest flexibility. The implementation and performance of each level will be described in this paper.

INTRODUCTION

Availability is a key parameter for big scientific facilities, especially for user facilities. High availability drives the reliability demands for the control system. Redundancy is a common approach to improve the reliability and availability of a system. Some efforts are made to implement the system redundancy under EPICS environment, e.g. the redundant IOC is developed based on the redundancy monitor task (RMT) at DESY for the cryogenic system of the European XFEL project [1]; redundancy technology of the IOC is realized via Xen or Linux-HA base on ATCT at Institute of High Energy Physics(IHEP), Chinese Academy of Sciences [2]; the redundant control system is designed at Shanghai Institute of Applied Physics(SINAP), Chinese Academy of Sciences for the Thorium Molten Reactor System(TMSR) project [3].

PROFINET is a standard for Industrial Ethernet, it is defined by PROFIBUS and PROFINET International (PI) and, since 2004, is part of the IEC 61158 and IEC 61784 standards. PI released the Specification "PROFINET IO System Redundancy" on July 19, 2011, which describes the mechanism to build up a redundant PROFINET IO system [4]. There is no product which is consistent with this specification at present. However there are some existing commercial redundancy solutions with PROFINET, e.g. the redundancy system based on Phoenix Redundancy Layer from Phoenix contact [5].

By integrating the commercial solution into EPICS environment, we set up a prototype system. The prototype system consists of 4 levels: the EPICS IOC, the PROFINET IO controller, the PROFINET media and the PROFINET IO device. Redundancy at each level is independent of redundancy at each other level to achieve highest flexibility.

SYSTEM ARCHITECTURE

The system architecture of the redundant control system is shown in Figure 1. The system includes 4 levels: the EPICS IOC, the PROFINET IO controller, the PROFINET media and the PROFINET IO device.

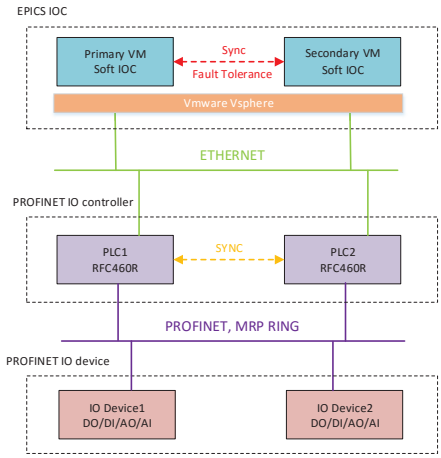


Figure 1: System architecture of the redundant control system.

EPICS IOC

VMware vSphere is the industry's leading and most reliable virtualization platform. It achieves always-available IT with live migration for virtual machines and high availability for applications in virtual machine (VM) clusters. VMware provides 4 features to improve availability: VMware vMotion, VMware Storage vMotion, VMware High Availability (HA), VMware Fault Tolerance (FT) [6]. In this redundant system, VMware FT is chosen to achieve hot standby for the EPICS IOC.

VMware FT provides continuous availability for VMs by creating and maintaining a Secondary VM that is identical to, and continuously available to replace, the Primary VM in the event of a failover situation. The Primary and Secondary VMs continuously exchange heartbeats. This allows the virtual machine pair to monitor the status of one another to ensure that Fault Tolerance is continually maintained [7].

The EPICS softIOC running on the VM communicates with the redundant pair of two RFC 460R PLCs via TCP/IP. An EPICS driver has been developed for the communication between the EPICS softIOC and the RFC 460R PLCs [8].

PROFINET IO Controller

The RFC 460R PLC from Phoenix Contact is a high performance controller that has been extended to offer redundancy functionality. The redundant pair consists of two synchronized RFC 460R PLCs connected via fiber optics. The built-in fiber optic interface is used for synchronization and adjustment between the connected devices [9].

* Work supported by National Natural Science Foundation of China (11375186)

† gfliu@ustc.edu.cn

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

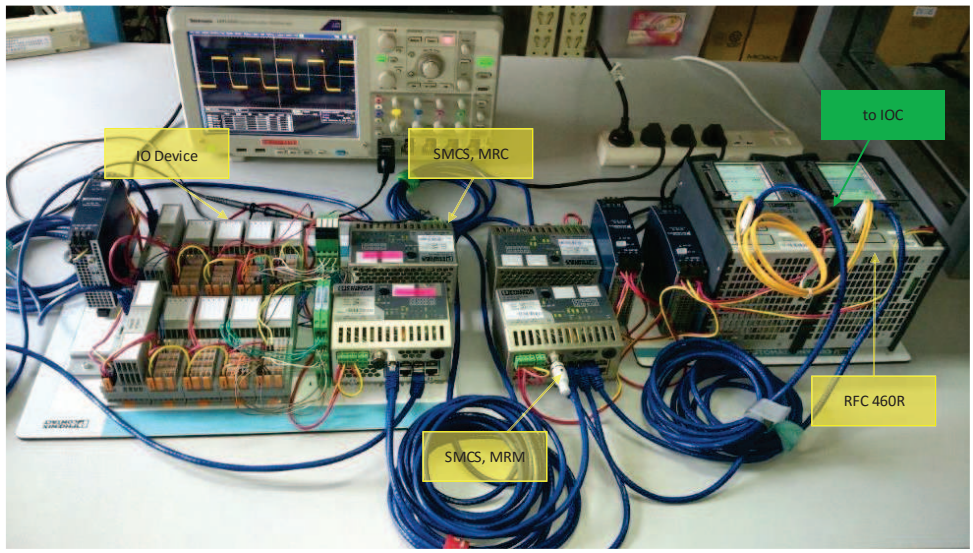


Figure 2: Photo of the prototype system.

During initial startup redundancy type FIRST must be assigned to one of RFC 460R PLCs and redundancy type SECOND to the other. The redundancy type does not change during operation. The SECOND PLC always has the IP address of the FIRST PLC increased by 1. The redundancy role of the RFC 460R PLC (i.e. PRIMARY or BACKUP) may change depending on the status of the redundant control system.

PROFINET Media

The Media Redundancy Protocol (MRP) is used to realize the redundancy of PROFINET. MRP is a self-recovery media redundancy protocols based on physical ring topological network architecture, designed for the fault of single switch or single switch link in ring network. One switch of the ring has the role of ring manager named Media Redundancy Manager (MRM), while the remaining switches have the role of ring clients named Media Redundancy Clients (MRCs). Each switch is connected to the ring through two ring ports. The MRM switch is responsible for handling failure/recovery events, and recovery time is a key parameter of MRP. [10].

PROFINET IO Device

The redundant pair of PROFINET IO devices consists of two PROFINET IO stations, one is primary station and the other is backup station. The wiring for the AO, AI, DO, DI modules installed on the PROFINET IO stations is redundant. Phoenix Redundancy Layer from Phoenix contact is used as the communication protocol between the redundant pair of RFC 460R PLCs and the redundant pair of PROFINET IO devices.

PROTOTYPE SYSTEM AND PERFORMANCE

A prototype system is set up based on the commercial solution, as shown in Figure 2. Two RFC 460R PLCs construct the redundant pair, four FL SMCS switches form the MRP ring, two AXL F BK PN bus coupler and the related IO devices establish the redundant pair. The EPICS softIOCs running on the VMs connect to the redundant PLC pair via Ethernet, however they are not visible in this photo.

EPICS IOC

In order to investigate the availability of the softIOCs running on the VMs with FT, an ao record is created, and it outputs a triangle waveform with 100 ms step. The EPICS extensions Striptool and the EPCIS command camonitor are used to monitor the value of the ao record.

Figure 3 shows the switch-over of the softIOC monitored by Striptool, and the switch-over time is about 800 ms. The switch-over time can be calculated precisely by camonitor, and the result is 809 ms, shown in Figure 4.

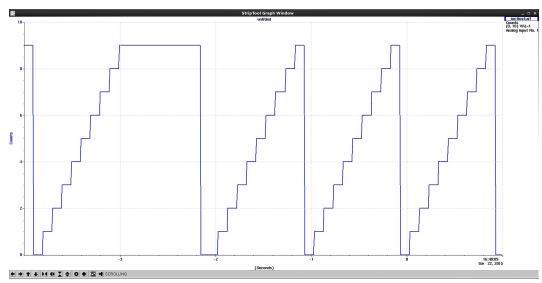


Figure 3: Switch-over of the softIOC monitored by Striptool.

```
iocHost:ai1                    2015-09-22 16:48:06.251241 7 HIGH MINOR
iocHost:ai1                    2015-09-22 16:48:06.351407 8 HIHI MAJOR
iocHost:ai1                    2015-09-22 16:48:06.451520 9 HIHI MAJOR
iocHost:ai1                    2015-09-22 16:48:07.260921 0 LOLO MAJOR
iocHost:ai1                    2015-09-22 16:48:07.447481 1 LOLO MAJOR
```

Figure 4: Switch-over of the softIOC monitored by camonitor.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

PROFINET IO Controller

Generally the control process is implemented in the PLC. A triangular wave is produced in order to simulate the control process. There are 9 switch-over ways for the redundant pair of the RFC 460R PLCs, they are switch-over by user, runtime error on Primary PLC, link down at the PROFINET controller interface, parameterization memory of the Primary PLC removed, switch-over by user (firmware service), the PROFINET controller of the Primary PLC has lost the connections to all devices, firmware update has started, firmware has shutdown, a timeout occurred on the SyncLink connection [9]. We test all 9 switch-over ways, and Figure 5 shows the switch-over with the 3rd way "link down at the PROFINET controller interface".

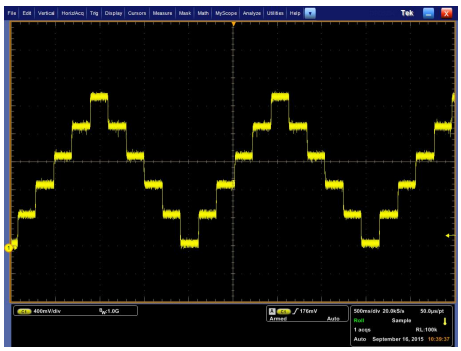


Figure 5: Switch-over of the PLC monitored by the oscilloscope.

The switch-over time is too short to observe from Figure 5. So we use the packet analyzer Wireshark to capture the communication frames between the PLCs and the PROFINET IO Devices. The switch-over time can be analyzed from the captured frames, Table 1 is the results. The average switch-over time is 6.229 ms, and the longest switch-over time is 7.078 ms.

Table 1: PLC Switch-Over Time Test

Number	1	2	3	4	5
Time(ms)	6.964	4.571	6.986	6.953	6.969
Number	6	7	8	9	10
Time(ms)	6.962	6.984	7.078	6.984	2.542

PROFINET Media

The recovery time is a very important parameter for a MRP ring. We also use Wireshark to capture the communication frames between the switches and analyze the recovery time, Table 2 is the results. The average recovery time is 69.338 ms, and the longest recovery time is 87.778 ms.

Table 2: MRP Ring Recovery Time Test

Number	1	2	3	4	5
Time(ms)	67.694	78.618	71.138	73.192	87.778
Number	6	7	8	9	10
Time(ms)	56.787	64.676	59.751	72.223	61.518

CONCLUSION

We set up the prototype system by integrating the commercial solution into EPICS environment. The prototype system consists of 4 levels: the EPICS IOC, the PROFINET IO controller, the PROFINET media and the PROFINET IO device. Redundancy at each level is independent of redundancy at each other level. This means each level can have different redundancy configurations. Beside its flexibility, the prototype system is also easy to implement, and the switch-over performance is good enough to adapt to the most control processes of the big scientific facility.

REFERENCES

[1] M. Clausen, G. Liu, B. Schoenbeurg, Redundancy for EPICS IOC, Proceedings of ICALEPCS2007, Knoxville, Tennessee, USA.

[2] Gang Li, Jijiu Zhao, Ge Lei, et al. Research of redundant EPICS/IOC based on the ATCA platform, Nuclear Electronics & Detection Technology, Vol. 20, No. 7, Jul. 2010.

[3] Zhang Ning, Yin Cong-cong, Han Li-feng, et al. Application Study of EPICS-based Redundant Method for Reactor Control System, Nuclear Electronics & Detection Technology, Vol. 33, No. 11, Nov. 2013.

[4] http://www.profibus.com/nc/downloads/downloads/PROFINET-io-system-redundancy-1

[5] Zhang Long, Du Pin-sheng, PROFINET Redundancy Fieldbus Ethernet PROFINET Technology, China Instrumentation, Vol. 5, 2011.

[6] https://pubs.vmware.com/vsphere-4-esx-vcenter.

[7] vSphere Availability Guide, http://www.vmware.com/products/vsphere/features/fault-tolerance

[8] Liu G, Li C, Li J, et al. EPICS Drive for PHOENIX CONTACT Redundant PLC, Proceedings of IPAC2013, Shanghai, China.

[9] Pheonix Contact, RFC460R User Manual, 2012.

[10] Giorgetti A, Cugini F, et al. Performance analysis of media redundancy protocol (MRP). IEEE Transactions on Industrial Informatics, Vol. 9, No. 1, Feb. 2013.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DESIGN OF CONTROL NETWORKS FOR CHINA INITIATIVE ACCELERATOR DRIVEN SYSTEM

Zhiyong He*, Yuxi Luo, Yuhui Guo, Qiang Zhao, Wenjuan Cui, Yichuan He
Institute of Modern Physics, Chinese Academy of Sciences, Lanzhou, China

Abstract

In this paper, we report the conceptual design of control networks used in the control system for China initiative accelerator driven sub-critical (ADS) facility which consists of two accelerator injectors, a main accelerator, a spallation target and a reactor. Because different applications have varied expectations on reliability, latency, jitter and bandwidth, the following networks have been designed for the control systems, i.e. a central operation network for the operation of accelerators, target, and reactor; a reactor protection network for preventing the release of radioactivity to the environment; a personnel protection network for protecting personnel against unnecessary exposure to hazards; a machine protection network for protecting the machines in the ADS system; a time communication network for providing timing and synchronization for three accelerators; and a data archiving network for recording important measurement results from accelerators, target and reactor. Finally, we discuss the application of high-performance Ethernet technologies, such as Ethernet ring protection protocol, in these control networks for CIADS.

INTRODUCTION

China is the world's most populous country with a fast-growing economy that has led it to become the largest energy consumer in the world in 2010. In China, nuclear power accounts for relatively small shares (nearly 1%) of the country's total energy consumption in 2011. China is actively promoting nuclear power as a clean, efficient, and reliable source of electricity generation. Driven by the national demand for safe disposal of nuclear waste as well as the potentials for advanced power generation, the Chinese Academy of Sciences initiated an accelerator driven sub-critical (ADS) program in 2011 under the frame of "Strategic Priority Research Program"[1]. The ultimate goal of the China ADS program is to build an industrial demo facility for ADS technology.

In an ADS system, a heavy metal spallation target located at the centre of a sub-critical core is bombarded by the high-energy protons from an accelerator. Figure 1 shows a drawing of China ADS facility which consists of an accelerator, a spallation target and a sub-critical reactor. As shown in Figure 1, a LINAC accelerator which includes two injectors, a medium energy beam transport line (MEBT), several spokes, and a high energy beam transport line (HEBT) is used in China ADS facility. In this paper, we report the conceptual design of control networks used in China ADS facility.

* corresponding author, Email: zyhe@impcas.ac.cn
This work is supported by Strategic Priority Research Program of Chinese Academy of Sciences (XDA03010000).

NETWORKS IN CHINA ADS FACILITY

Because different applications have varied expectations on reliability, latency, jitter and bandwidth, several networks have been used in ITER (International Thermonuclear Experimental Reactor) (e.g. [2]-[3]). The China ADS system includes two totally different facilities, accelerator and reactor. To successfully integrate the two different facilities into one system, several networks are required in the control system for China ADS system, while some networks are designed especially for accelerator and other networks are used especially for reactor.

Central Operation System and Network

Control systems in ADS may include a central control system and several local control systems. The central control system is used to control the overall ADS facility, in particular, to exactly couple the high-energy beam from the accelerator to the spallation target located at the centre of the reactor core. The central operation network (CON) used in the central control system provides interface between the operation system in the central control room and various controller and sensors in the system.

Reactor Protection System and Network

To guarantee the integrity of the reactor and to avoid an undue risk to the public health and safety, the plant design incorporates a reactor protection system (RPS). The overall purpose of the RPS is to prevent the release of radioactivity to the environment. To meet this objective, this system is capable of supplying reactor and component trip signals if safe operating limits are exceeded, and initiates the engineered safety features actuation(s) if an accident occurs.

The RPS contains two complete and independent trains of monitoring systems. If a monitoring system detects an unsafe event, signals are transmitted via the reactor protection network (RPN) to the RPS. The RPS determines whether the coincidence for a reactor trip function is satisfied. If so, the protection system opens the reactor trip breakers. Opening these breakers removes power from the control rod drive mechanisms, allowing the rods to fall into the reactor core. If an accident occurs and an engineered safety features actuation is required, the protection system actuates the appropriate safety equipment.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

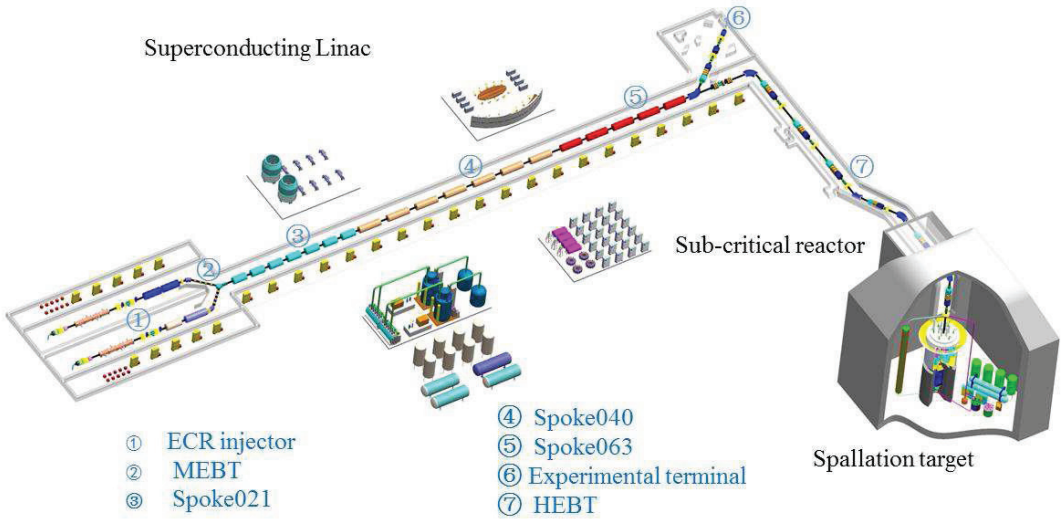


Figure 1: Layout of main systems in China ADS facility.

The RPS is designed to be independent of all process control systems. However, in certain applications, some control signals and other nonprotective functions are derived from individual protection channels through isolation amplifiers. These control signals are transmitted via the operation network to the main control room.

Personnel Protection System and Network

The personnel protection system (PPS) will protect personnel against unnecessary exposure to hazards from the machine, including radioactivity and electromagnetic radiation, electrical shock, and other dangerous phenomena, such as faulty ventilation, helium release, or oxygen deficiency hazards.

The PPS contains some sensors and controllers in the ADS facility, a network and the operation system in the central control system. If a sensor in the monitoring system detects an unsafe event, signals are transmitted via the personnel protection network (PPN) to operator's safety desk in the central control room. If an accident occurs, the personnel protection system actuates the appropriate safety equipment. The PPS safety functions will be defined and designed in advance, by performing a complete set of hazard and risk analysis for accelerator, spallation target and reactor.

Machine Protection System and Network

The machine protection system (MPS) will protect the machine's equipment in the accelerator from damage induced by beam losses and/or malfunctioning equipment. Protection will be achieved by initiating an emergency shutdown upon detection of critical and non-nominal conditions. The machine protection network provides communication between the machine protection system and the sensors or controllers. If a sensor in the monitoring system detects a non-nominal event, signals are transmitted via the machine protection network to the machine protection system. The machine protection system determines whether an emergency shutdown should be initiated.

In an accelerator driven sub-critical system, an emergency shutdown of the accelerator will result in the shutdown of the reactor. Therefore, the protection functions in MPS will be defined and designed very carefully in advance, by performing a complete set of analysis for accelerator, spallation target and reactor.

Time Communication Network

Time Communication Network (TCN) provides timing and synchronization with an accuracy of 10 ns RMS. The official project time used in China ADS system is UTC. Only plant system I&C requiring high accuracy time synchronization shall be connected to TCN. Plant system I&C may have multiple TCN network interfaces.

IEEE-1588, also called Precision Time Protocol (PTP), will be selected for the Time Communication Network (TCN). The N.I. PXI-6683 board may be used as the timing and synchronization board for fast controllers. The NI PXI-6683 and PXI-6683H timing and synchronization modules synchronize PXI and PXI Express systems using GPS, IEEE 1588, and IRIG-B to perform synchronous events. The PXI-6683 can generate events and clock signals at specified synchronized future times and timestamp input events with the synchronized system time. The PXI-6683 features an onboard TCXO that can be disciplined to GPS, IEEE 1588, and IRIG-B for long-term stability. These modules support the 2008 version of IEEE 1588, also known as IEC 61588:2009.

Data Archiving System and Network

As a research facility, acquiring, managing and archiving its data is an essential task for China ADS facility. All the data produced during the operation of ADS system needs to be stored and managed. The archive's main responsibility is to store multiple copies of

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

all digital objects and to be able to recall them on request. These digital objects grow in capacity over the years as the accumulated data grow. The stored data is expected to be investigated by researchers during the operation and beyond. Data archiving network is a scalable communication channel which allows the scientific data to be transferred from the sensors and controllers into the Data Archiving system. The data archiving network may be deployed using a dedicated high-throughput Ethernet communication network.

HIGH-PERFORMANCE ETHERNET TECHNOLOGIES

Ethernet IEEE 802.3 is the most widely accepted networking standard for many reasons. It is one of the cheapest solutions for shared media access among computers. Ethernet also provides enough flexibility to operate at varying data rates from 10 Mbit/s to 100 Gbit/s. Another distinct advantage of an Ethernet-based access network is that it can be easily connected to the customer network, due to the prevalent use of Ethernet in corporate and, more recently, residential networks. Ethernet's wide acceptance has also brought wide availability of hardware and support for every computer platform. A typical service provider's network is a collection of switches and routers connected through optical fiber.

The Ethernet topology could be a ring, a tree, a star (hub-and-spoke), or full or partial mesh. Reliability, security, delay or latency, throughput, ease of use, and availability are the main issues while choosing the communication type. In the time communication network or machine protection network, delay should be minimized. A tree or star topology can be considered. In the data archiving network, a dedicated high-throughput Ethernet should be used to maximize the throughput. On the other hand, in the central operation network, the reactor protection network, or the personnel protection network, network reliability is the critical issue while choosing the communication type. Various Ethernet redundancy methods, such as Ethernet ring protection, can be used to improve reliability. Furthermore, both redundant Ethernet and wired lines should be used in the reactor protection network to provide an extremely reliable communication link.

Ethernet Ring Protection Networks

The Ethernet Ring Protection (ERP) protocol, defined in ITU-T G.8032 [4], provides protection for Ethernet traffic in a ring topology, while ensuring that no loops are within the ring at the Ethernet layer. ERP networks have been used in the control systems for accelerator injector I and II in China initiative accelerator driven sub-critical (CIADS) facility. In this section, we discuss the potential application of ERP in the central operation network for CIADS.

ERP builds a logical ring topology while maintaining a loop-free forwarding mechanism by logically blocking a link port in the ring, referred to as Ring Protection Link (RPL) (e.g. [5]-[6]). Once a link fails, the vertices adjacent to the failure block the failed link, and the RPL is unblocked. With this mechanism, an Ethernet ring maintains a logical linear network. Under a single failure condition, a fast protection signal from an ERP protocol message can relocate block positions and provide protection switching times of less than 50 ms, maintaining normal service. In principle, when multiple links fail in a ring an ERP cannot provide protection.

We denote a network topology by a graph $G(V, L)$, where V is the set of vertices and L is the set of links, indexed by v and l, respectively. Figure 2 shows three examples of Ethernet ring network topologies, where there are 22, 37, and 37 vertices in Figure 2(a), (b) and (c), respectively, and 25, 43, and 40 links in Figure 2(a), (b) and (c), respectively.

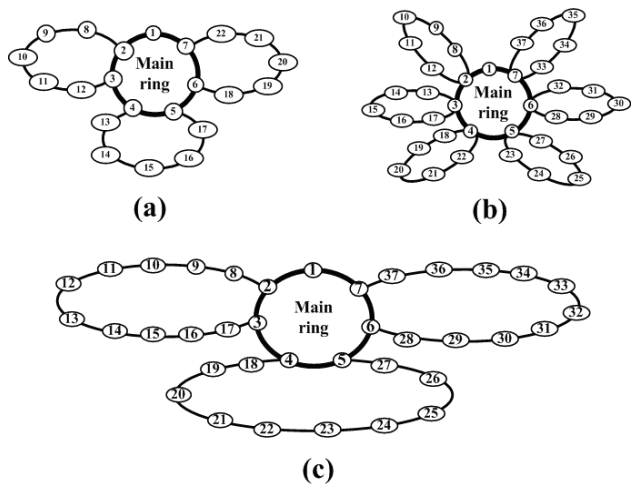


Figure 2: Three examples of Ethernet ring networks. (a) One main ring and three subrings. There are 5 vertices in each subring. (b) One main ring and six subrings. There are 5 vertices in each subring. (c) One main ring and three subrings. There are 10 vertices in each subring.

The set of the simple rings in G is denoted by R. A simple ring is composed of a set of links that form a physical ring/cycle with no straddling links. For example, vertices {1, 2, 3, 4, 5, 6 and 7} in Figure 2 form a simple ring which is used as the main ring.

Reliability Analysis of ERP Networks

In order to evaluate the reliability of an ERP network, we define the following variables. p_{link}^{1} and p_{link}^{0} are the possibilities of single link to be connected or disconnected, respectively. F_{ring}^{1} and F_{ring}^{0} are the possibilities of Ethernet ring protection to be successful or fail, respectively. In the simulation, with a given p_{link}^{0}, the random sampling is used to determine whether the i-th link in the set L is connected or not in each time cycle. Then, all links in each ring (main ring or subring) will be checked. If two or more links fail in a ring, Ethernet ring

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

protection is taken as failure in this time cycle. For example, in Figure 2(a), four rings (a main ring and 3 subrings) are checked. If two links in the same ring fail, e.g., the link from vertex 8 to vertex 9 and the link from vertex 10 to vertex 11, ERP in this time cycle is recorded as failure. Finally, the possibilities of ERP failure is calculated as follows, $F_{ring}^{0} = N_{fail} / N$, where N is the total sampling number and N_{fail} is the number of Ethernet ring protection failure.

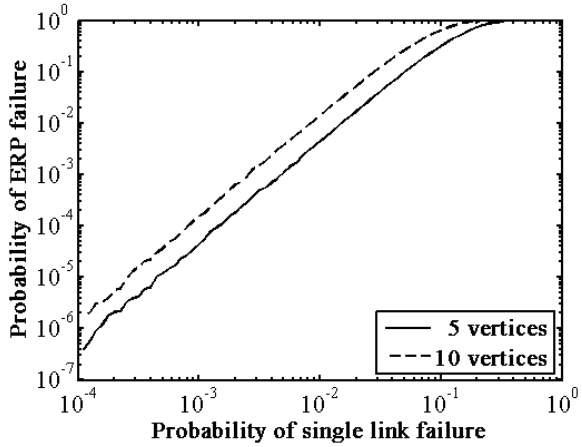


Figure 3: The possibilities of Ethernet ring protection (ERP) failure F_{ring}^{0} as a function of the possibilities p_{link}^{0} of single link failure. Two topologies in Figure 2(a) and Figure 2(c) are compared, where 5 or 10 vertices exist in each subring.

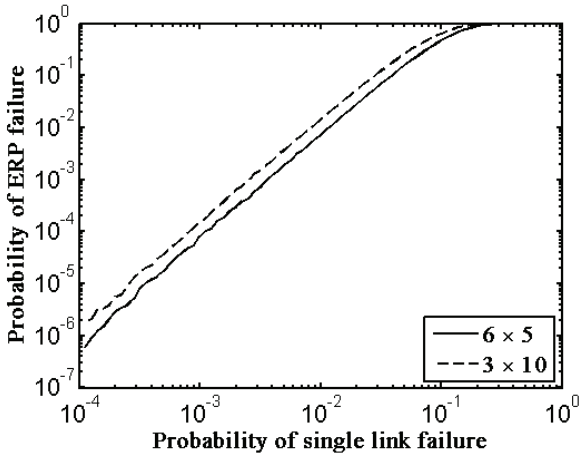


Figure 4: The possibilities of Ethernet ring protection (ERP) failure F_{ring}^{0} as a function of the possibilities p_{link}^{0} of single link failure. Two topologies in Figure 2(b) and Figure 2(c) are compared, where $m \times n$ represents that there are m subrings and there are n vertices in each subring.

Figure 3 shows the possibilities of ERP failure F_{ring}^{0} as a function of the possibilities p_{link}^{0} of single link failure for two topologies in Figure 2(a) and Figure 2(c). The difference between two topologies in Figure 2(a) and Figure 2(c) is that the vertex number in each subring is different. It is shown clearly in Figure 3 that increasing the number of vertices in each ring increases the possibility of ERP failure.

Figure 4 shows the possibilities of ERP failure F_{ring}^{0} as a function of the possibilities p_{link}^{0} of single link failure for two topologies in Figure 2(a) and Figure 2(b). The difference between two topologies in Figure 2(a) and Figure 2(b) is that the number of subrings is different. As shown in Figure 4, for a given total number of vertices, increasing the number of rings improves the reliability of ERP networks.

CONCLUSION

We have reported the conceptual design of control networks for China ADS facility. Six networks have been discussed, i.e. a central operation network, a reactor protection network, a personnel protection network, a machine protection network, a time communication network, and a data archiving network. Finally, we have studied the application of Ethernet ring protection network in CIADS, by evaluating the reliability of three Ethernet topologies with ERP protocol. The simulation results indicate that we should decrease the number of switches in each ring, in order to lower the possibility of ERP failure. If we have to implement many switches in the ERP network, we should increase the number of rings, in order to improve the reliability of ERP networks.

REFERENCES

[1] W.L.Zhan, H.S.Xu, et al., “Advanced fission enery program-ADS transmutation system”, Bulletion of Chinese Academy of Sciences, p.375-381, (2012).

[2] H. Gulati, H. Dave, F. Di Maio, A. Wallander, “Present status and future road map for ITER CODAC networks and infrastructure,” Fusion Engineering and Design 85 (2010) 549–552.

[3] S. Kitazawaa, K. Okayamab, Y. Neyatania, F. Sagotb, D. Houtteb, “RAMI analysis of the ITER Central Safety System,” Fusion Engineering and Design, 89 (2014) 800–805.

[4] Ethernet Ring Protection Switching ITU-T Rec. G.8032/Y.1344, 2010.

[5] J. Ryoo, H. Long and Y. Yang, M. Holness, Z. Ahmad and J. Rhee, “Ethernet Ring Protection for Carrier Ethernet Networks,” IEEE Communications Magazine, pp. 136-143, September 2008.

[6] M. Nurujjaman, S. Sebbah, and C. Assi, "Multi-Ring ERP Network Design: A Traffic Engineering Approach," IEEE Communications. Letters, vol. 17, no.1, pp. 162-165, 2013.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONTROLLING CAMERA AND PDU

O.J. Mokone, T. Gatsi, SKA South Africa NRF, South Africa

Abstract

The 64-dish MeerKAT radio telescope, currently under construction in South Africa, will become the largest and most sensitive radio telescope in the Southern Hemisphere until integrated with the Square Kilometre Array South Africa (SKA SA). This paper will present the software solutions that the MeerKAT Control and Monitoring (CAM) team implemented to achieve control (pan, tilt, zoom and focus) of the on-site video camera using the Pelco D protocol. Furthermore this paper will present how the outlets of the Power Distribution Unit (PDU) are switched on and off using Simple Network Management Protocol (SNMP) to facilitate emergency shutdown of equipment.

INTRODUCTION

The current running KAT-7 radio telescope (a MeerKAT prototype), which is running in the Karoo, is utilising three fixed cameras. Fixed cameras sometimes do offer appreciable performance in terms of resolution and image quality but do have amongst others a host of shortcomings for all their intents and purposes.

Despite them being more durable and needing less of mechanical service, the fixed cameras have the disadvantage of not covering a large area in the vicinity of the dishes and hence increases the cost of purchasing more fixed cameras in order to monitor the telescopes and the surrounding area.

To avert this challenge, the SKA SA MeerKAT team procured a Pan, Tilt and Zoom (PTZ) video camera. This Camera has been incorporated into the MeerKAT system because it can closely monitor the area under surveillance through the exploitation of the zoom-pan-tilt functionalities that are built within the camera.

The PTZ camera implements a state of the art technology, which is more appropriate for the MeerKAT project as it surveils a wide area around the dishes and allows for the pan, tilt and zoom capabilities. The PTZ camera is connected to the Power Distribution Unit (PDU). The PDU makes it possible to switch on/off the camera electronics that are connected to the specific PDU outlets remotely. The software that control and monitor camera was developed using Python programming language running on Ubuntu 14.04 LTS.

CAMERA SOFTWARE IMPLEMENTATION

Karoo Array Telescope Control Protocol (KATCP) [1] is a simple ASCII communication protocol layered on top of TCP/IP and is used for the monitoring and control of hardware devices. One of the most prevailing features of the KATCP protocol is its capability to interrogate KATCP servers for sensors. Interrogation of sensors [2] and requests, down to device level through KATCP, supports fluid run-time detection of system configuration, e.g. when a new sensor is added to any level of the system (including a hardware device), the rest of the CAM automatically discovers the change on-line on, and includes it in the monitor store and in its interfaces. Based on the functionality of the KATCP protocol, the video camera software had to implement similar technologies that respond to KATCP command messages.

Pelco D is a popular PTZ camera control protocol used in the CCTV industry. The camera control software uses Pelco D protocol over RS485, a TCP/IP physical interface. This is a client-server architecture, where the server is KATCP device server that connects directly to the camera device and the client is the CAM (Control And Monitoring) system software. One of the major roles of the server is to serve as a protocol translator; it translates Pelco D command messages to KATCP command messages and vice versa. Pelco D consists of 7 hexadecimal bytes. Table 1 (below) shows Pelco D message formats and their descriptions.

Table 1: Pelco D Message Format

Byte Number	Description
Byte 1	Synch Byte
Byte 2	Address
Byte 3	Command 1
Byte 4	Command 2
Byte 5	Data 1
Byte 6	Data 2
Byte 7	Checksum

Byte 1 is the synchronization byte, which is always fixed to FF. Byte 2 is logical address of the camera being controlled. Byte 7 is checksum, which is the 8 bit (modulo 256) sum of the payload bytes byte 2 through byte 6 in the message. The rest of the byte numbers depends on the commands that one is passing to the server.

PRESETS

The PTZ Camera is capable of storing and accessing up to 127 presets. These stored presets reside in a non-volatile memory within the camera. They can be accessed using an identity number from 1 through 127. The identity that is used to store a preset can subsequently be used to “go-to” to this preset, thus causing the camera to move to the previously stored configuration of pan, tilt,

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

zoom. Also, the previously stored presets positions can be cleared, which removes it from persistent memory.

To set the preset the CAM system (client) issues a KATCP message as "? preset-set 3" to the KATCP device translator. This KATCP message is then translated to Pelco D command as [FF, 0x01, 0x00, 0x03, 0x00, 0x3, 0x07]. The KATCP device translator relays the Pelco D message to the camera. The camera will then store the position that is currently in its persistent memory.

Query Position

The client has capability to query the position (PTZ) that the Camera is currently set to. To query zoom position the client will send a KATCP request message (? zoom-position) to the server and the server will then translate the KATCP message to Pelco D command message (FF, 0x01, 0x00, 0x55, 0x00, 0x00, 0x56), which will then send the Pelco D message to camera. The camera replies with the following message 0xFF, 0x01, 0x00, 0x5D, MSB, LSB, checksum. The translator then uses MSB (most significant bit) and LSB (least significant bit) to convert to the zoom position that the lense is currently at and send it to the client.

POWER DISTRIBUTION UNIT

PDU is used to switch on and off all camera electronics remotely. To achieve the controlling and monitoring of PDU the SNMP is used.This is a client-server architecture, where the server is KATCP device, which communicates directly with the PDU device and the client, is the CAM system software. One of the major roles of the server is to serve as a protocol translator; it translates SNMP command messages to KATCP command messages and from KATCP messages to SNMP messages.

HANDLING SNMP

When monitoring the PDU (whether outlets are on or off) the client polls or requests the status of all the outlets. This exercise of polling is impractical and expensive since the client has to poll or request information from every object. This process can be diagrammatically represented in Figure 1. The solution is to use traps. Traps are alerts/notifications generated by the PDU. Figure 2 illustrates the use of traps. Trap messages will be sent to a predefined trap host.

A predefined trap host has snmptrapd (Simple Network Management Protocol Daemon). Snmptrapd is trap daemon running that is waiting for the traps. It receives traps from the PDU at port 162. Snmptrapd has the capability of executing a particular script or command depending on the Object Identifier (OID). Snmptrapd is a linux command line program and is easy to use. After snmptrapd receives the trap it determines which script to run and directly polls the associated device to get a better understanding of the event/trap. Scripts are light and simple KATCP client scripts that receive (read in) traps from snmptrapd and update the CAM system. The script has a mechanism of knowing the destination (host and port) of the trap.

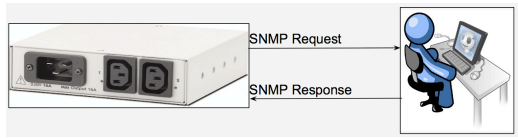


Figure 1: Polling.

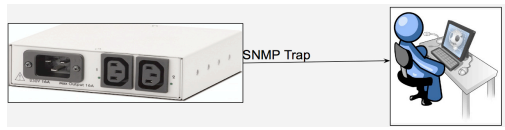


Figure 2: Traps.

START SNMPTRAPD

Starting snmptrapd is managed using the init script named snmpd. Edit the file /etc/default/snmpd to set the option TRAPDRUN to yes.

TRAP DAEMON CONFIGURATION

The daemon has a simple configuration file. One has to setup snmptd to log into syslog by editing the file /etc/snmp/snmptrapd.conf to set the option authCommunity log public. In the configuration file we defined two pieces of information that tells snmptrapd how to direct the incoming traps, the information is the OID of the incoming trap and the location of the traphandler. In our case the following line was added to the configuration file traphandle .1.3.6.1.4.1.318.0.269 python /home/snmp_apc_script.py, whenever an outlet is switched on/off the script snmp_apc_script.py will be executed.

While it is possible to switch on and off outlets from the web, we turn the outlets on and off via SNMP [3]. This makes it possible to switch on and off outlets automatically. Typically SNMP OID of setting ‘on’ outlet number one is (1.3.6.1.4.1.318.1.1.4.4.2.1.3.1.1). The OID of requesting/polling status of outlet number one is (1.3.6.1.4.1.318.1.1.4.2.2.0.1).

STREAMING

To capture video and snapshots from a Camera and stream them through a webserver we use motion. Motion has simple webserver built in. The video stream is in mjpeg format from cameras. It is able to detect if a significant part of the picture has changed, in other words it can detect motion. Motion comes with a configuration file, which needs to be changed to suit user specifications. To start motion process we run motion -n –c motion.conf. From a browser we point to the IP address of where motion is running at and the associated port number.

CONCLUSION

The text based KATCP protocol utilised in the implementation and development of the Camera software

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

solution has the advantage of ease of use and debugging as this protocol is widely known and used in the KAT-7 as well as the MeerKAT. Although the PTZ cameras are more applicable and more versatile than the fixed cameras, they do bring a lot of financial constraint on the project. The PTZ camera requires technical maintenance from qualified personnel to keep it running. The image quality on a PTZ camera can be poor in comparison to a fixed unit camera. This is because the CCD sensors on the PTZ cameras have to be smaller to accommodate the gears and motor mechanisms, so the image quality suffers slightly.

REFERENCES

[1] S. Cross et al, "Guidelines for Communication with Devices", SKA SA, July 2012, http://pythonhosted.org/katcp/_downloads/ NRF-KAT7-6.0-IFCE-002-Rev5.pdf

[2] L van den Heever et al, "MeerKAT CAM Design Description, Rev 1", SKA SA, August 2013, http://tinyurl.com/MeerKAT-CAM-Public-Docs

[3] L van den Heever et al, "MeerKAT CAM Requirement Specification, Rev 2", SKA SA, August 2013, http://tinyurl.com/MeerKAT-CAM-Public-Docs

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INTERFACING EPICS TO THE WIDESPREAD PLATFORM MANAGEMENT INTERFACE IPMI

M. Ritzert*, Heidelberg University, Germany

Abstract

The Intelligent Platform Management Interface (IPMI) is a standardized interface to management functionalities of computer systems. The data provided typically includes the readings of monitoring sensors, such as fan speeds, temperatures, power consumption, etc. It is provided not only by servers, but also by µTCA crates that are often used to host an experiment's control and readout system. Therefore, it is well suited to monitor the health of the hardware deployed in HEP experiments. In addition, the crates can be controlled via IPMI with functions such as triggering a reset, or configuring IP parameters. We present the design and functionality of an EPICS module to interface to IPMI that is based on ipmitool. It supports automatic scanning for IPMI sensors and filling the PV metadata (units, meaning of status words in mbbi records) from the IPMI sensor information. Most importantly, the IPMI-provided alarm thresholds are automatically placed in the PV for easy implementation of an alarm system to monitor IPMI hardware.

IPMI

The Intelligent Platform Management Interface (IPMI) is a common interface to monitoring and management data of servers, ATCA and µTCA crates and similar devices. Monitoring data is provided as analog or digital sensor readings. The sensor metadata provided includes several alarm thresholds that can be used in a monitoring system. Server management is performed via messages sent to the IPMI device from the control PC.

The structure of a typical IPMI system in an ATCA crate is shown in Fig. 1 The typical mode of access to IPMI systems in a distributed system landscape is via Ethernet LAN. The connection is received on the server side by the IPMI baseboard management controller (BMC) that operates independently of the host's CPU, often even when the host device is currently powered down. The BMC communicates via the intelligent platform management bus (IPMB) with other management controllers (MCs) using a modified I²C protocol. Each MC manages a set of sensors, and can be queried to enumerate and read these sensors in a standardized format. Automatic detection of the sensors available in a system is therefore possible, and all sensors are accessed in the same way; all details of the actual hardware access are hidden by the MC.

Since IPMI access is available also when the host device is powered off, it is also a useful tool to remotely manage compute farms. Servers or crates can be reset, started, or stopped without invoking the OS.

When custom hardware is built for IPMI-enabled platforms such as ATCA crates, the IPMI system can be extended by adding new MCs. The required connections to interface to the IPMB are provided on the backplane's connectors.

IPMITOOL-IOC

Ipmitool [1] has been chosen as the basis to develop an interface (input/output controller, IOC) from EPICS to IPMI systems. The main functionality of ipmitool is linked into a static library that is not installed into the target system. Therefore, the source code of ipmitool must be available at build time.

Since ipmitool is used for the communication, advanced features of IPMI such as authentication are easily implemented. Compatibility with a wide range of hardware is given; the systems examined during development include ATCA crates and 19" servers of several brands.

The IOC is written in C++, interfacing to the EPICS CA libraries and the ipmitool library written in plain C. Since data queries via IPMI are "slow" in the EPICS sense, the asynchronous PV support of EPICS is used. Typical scanning times for all sensors in a system could be in the order of a few seconds.

A custom logging library written in C++ is used to send textual log messages from the IOC to a STOMP server that can pass them on to JMS2RDB, so that they can eventually be displayed in the "Message History" view of CSS. This way, all deployed IOCs report their messages to a central place, and messages are less likely to be overlooked.

Usage of the Module

First, the connection to the IPMI host has to be established. From the IOC, this can be done by calling the new ipmiConnect function. It accepts parameters to define a numeric handle for the connection, the host name, and if required, the user name, password, and access level. Next, the device is scanned for sensors by calling ipmiScanDevice. The result of the scan is used to populate the meta data of the PVs that are loaded via the standard dbLoadRecords call.

Optionally, the result of the device scan can be dumped in EPICS database format to serve as a starting point when preparing the IOC to control the device.

The link addresses in the database use the AB_IO format "#Lw Ax Cy Sz @p", where w is the id of the link established with ipmiConnect, x is the IPMB target address, and y is the sensor id. z is reserved for future use (possibly for IPMB bridging addresses), as is p.

Automatic Sensor Detection

The IOC uses ipmitool's functions to enumerate the sensors in a system. It maintains a work list of IPMB addresses

* michael.ritzert@ziti.uni-heidelberg.de

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

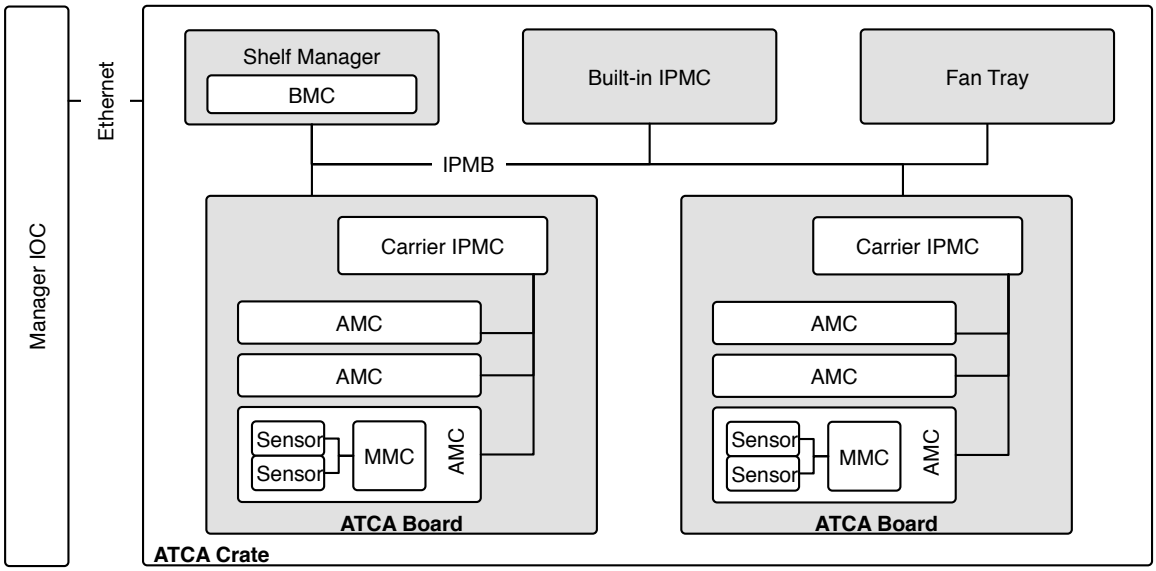


Figure 1: Typical IPMI system in an ATCA crate.

that are to be scanned. The list is amended, as new addresses are detected in the system via device locator records. In addition, the system is checked for the presence of the PICMG extension, that also provides access to a sensor bus.

The capability of IPMI to add hotplug devices at any time is not relevant for the intended application, and is not supported at the moment. It is planned to amend the code in the future.

EXAMPLE

The IOC is developed to be used with ATCA and µTCA crates housing readout and control hardware for the Belle II PXD subdetector. The health of the crates and custom add-on cards is monitored via EPICS. The information obtained via IPMI nicely complements health information read directly from the onboard FPGA. Figure 2 shows a CSS OPI

Figure 2: Control System Studio (CSS) displaying a GUI showing current IPMI readings from an ATCA board.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

screen that summarizes the state of one of the ATCA boards used in the PXD readout system. All data visible in this screen comes directly from the IPMI IOC. As can be seen, the analog readings are complemented with alarm thresholds automatically read from the crate. The alarms generated via these thresholds will be used to alert the operators to hardware malfunctions in the crates.

CONCLUSION AND OUTLOOK

An EPICS IOC to interface to IPMI-enabled devices has been written. A first stable version that supports automatic detection of sensors in a system and their readout is available. The next version of the IOC will support sending IPMI messages that are used to trigger actions in the system, e.g. a hard reset of a server. A future version will support accessing sensors that are dynamically added to the system. This involves processing IPMI events that are created when new hardware is added, and defining a new link naming scheme that takes into account that the sensor ids are no longer predictable in this case. Also planned for a later version is support for reading the IPMI event log.

ACKNOWLEDGEMENTS

This work has been supported by the German Federal Ministry of Education and Research (BMBF).

REFERENCES

[1] http://sourceforge.net/projects/ipmitool/.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DATA DRIVEN SIMULATION FRAMEWORK

Amar Banerjee, Subhrojyoti Roy Chaudhuri, Puneet Patwari, TRDDC, Pune, India
Lize Van Den Heever, SKA South Africa, Cape Town, South Africa

Abstract

Control systems for radio astronomy projects such as MeerKAT[1] require testing functionality of different parts of the Telescope even when the system is not fully developed. Usage of software simulators in such scenarios is customary. Projects build simulators for subsystems such as dishes, beam-formers and so on to ensure the correctness of a)their interface to the control system b)logic written to coordinate and configure them. However, such simulators are developed as one-offs, even when they implement similar functionality. This leads to duplicated effort impacting large projects such as Square Kilometer Array[2]. To mitigate this we leverage the idea of data driven software development and conceptualize a simulation framework that reduces the simulator development effort to: 1)capturing all the necessary information through instantiation of a well-defined simulation specification model 2)configuring a reusable engine that performs the required simulation functions based on the instantiated and populated model provided to it as input. We discuss the results of a PoC for such a simulation framework implemented in the context of Giant Meter-wave Radio Telescope[3] in this paper.

INTRODUCTION

Large projects that involve implementation of large hierarchy of control systems generate dependencies across the controllers to be developed and as a result on the teams developing them. This inter-dependency creates problems in the verification of controllers developed across teams since the individual teams follow their own time-lines which often are not well synchronized.

A general solutions to this problem is replacing the missing components with simulators to aid the verification of the dependent components. This however increases the effort to manually develop the simulators which also at times results in duplication of efforts.

Although the need for simulators in such scenarios seem to be essential, it is desired to reduce the cost of building such simulators since it might impact the overall cost of projects such as Square Kilometer Array(SKA) significantly, as such projects have large number of modules.

With the proposed simulation and testing framework it becomes possible to achieve the goal of verification of the module with simulation, following the model driven approach. The framework incorporates meta models of the controller node, the simulator and testing based on which it automatically generates the simulators. Much of this information is derived from the Self Description Data (SDD) which contains description of the Controller to be developed.

In this paper we discuss the simulator and test framework, the architecture and the working of the framework.

The paper starts with a discussion on standard practice involved in control system testing and verification using MeerKAT as the Case study. It is followed by the architecture section where we describe our understanding and design of the framework. In the final section we present a proof of concept showing the use of our simulation framework. This is followed by a conclusions section where we describe the conclusions of our work.

STANDARD PRACTICE

MeerKAT Case Study

One of the development practices of the MeerKAT CAM (Control and Monitoring) team was to use a fully simulated system at all times. The MeerKAT CAM team has been using simulators extensively and continuously for development, testing and qualification of the CAM subsystem functionality throughout the MeerKAT project life-cycle, since the early days of Fringe Finder for the very first two antennas in the Karoo, through KAT-7[4] to this day for MeerKAT with each array release. It is possible to run a MeerKAT CAM configuration including only simulated devices, or any combination of real and simulated devices combined. This allows full software development, unit testing, integration testing, and CAM subsystem qualification without any dependency on the hardware being available.

While the CAM team was responsible for developing most of the simulators, some of these device simulators were contractually delivered by the subsystem contractor to ensure that, given their knowledge of the device, the behaviour of the device is reflected with sufficient accuracy by the device simulator. In cases where the subsystem contractor did not deliver such a simulator, the CAM team developed a software simulator for the KATCP interface. Preparing the simulators gave the CAM team a valuable opportunity to gather information about the behaviour of the other subsystems even before those subsystems have been fully developed and are ready for integration.

Each simulator represents the specific messages on KATCP (KAT Control Protocol) interface for a subsystem (commands/requests and monitoring points/sensors), as

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

well as simulating the expected behaviour of the module when commanded through the control-and-monitoring interface, including maintaining state and mode and reflecting current state in monitoring points. Each simulator also provides a test interface that is used to stimulate the simulator to affect responses and outcomes, alarms and failure conditions.

This approach has been extremely beneficial for the MeerKAT CAM development, but the simulators can be improved by providing a simulation framework that can be personalised for each specific interface by using data-driven specification of the interface instead of writing each simulator manually and keeping it up to date separately. An additional improvement would be to extend the simulation framework with a standard behaviour extension module that can also be personalised through a data-driven behaviour specification. This could include specifying the timing taken by commands before responding, specifying interrelations between commands and reflecting the outcome in monitoring points, and also to relate commands to one another. To explore the benefits of these improvements a POC (Proof of Concept) has been developed by TRDDC on the GMRT with a possible approach to a data-driven simulation framework.

PROPOSED ARCHITECTURE

Simulation Specifications Model

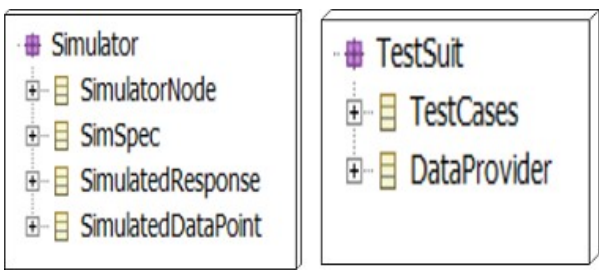


Figure 1: Simulator Model and Test Suite Model.

As Fig. 1 explains the abstract controller simulator model comprises of 2 parts :-

1)Software Emulator and

2)Hardware Simulator.

The analysis of this model leads to gathering requirements for the simulation and testing framework.

From the figure, the information that form the part of the controller simulator are grouped as

a) details such as commands, responses, events, alarms and data

b) communication protocols and protocol translation rules,

c) behavioral specifications such as state machines

d) specific input related to simulation such as simulated responses, data stream, simulation of the behavior of the underlying infrastructure such as unexpected responses, streams

e) skeleton structure of the implementation of the code.

The idea is to capture all these items as a part of the simulator model so that an environment can be provided to capture these information in a structured manner. This provides all the key requirements that the simulation and testing framework needed to support.

Architecture

Using the Model Driven Engineering philosophy, we provide as part of this framework aDomain Specific Language (DSL) MnC&ML[5] to capture all the above information. We term an instance of such information captured for a controller node as the Self-Description Data (SDD) for the controller node. Hence an SDD instance can capture all the key information pertaining to a controller that is not ready yet but is required to be simulated.

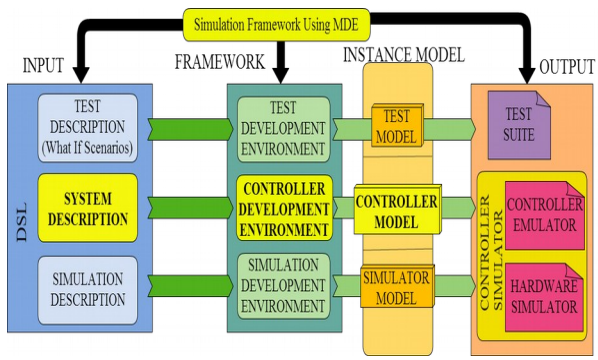


Figure 2: Role of Environments.

The high level architecture of the framework is provided in Fig. 3 and the information related to the implementation of the same such as the technology stack used and so on can be seen from Fig. 4.

From the instantiated model or SDD the framework then code generates the controller/software simulator, hardware simulator and the test suit to test the simulator itself.

The test suit acts as a driver and calls the functionality of the controller and the simulator provides with a suitable simulated reply. The controller node which needs to be tested can then be connected to the simulator to then carry on with the testing of the controller node.

As can be seen from the figure below, the environments help to create the different parts of the models:-

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

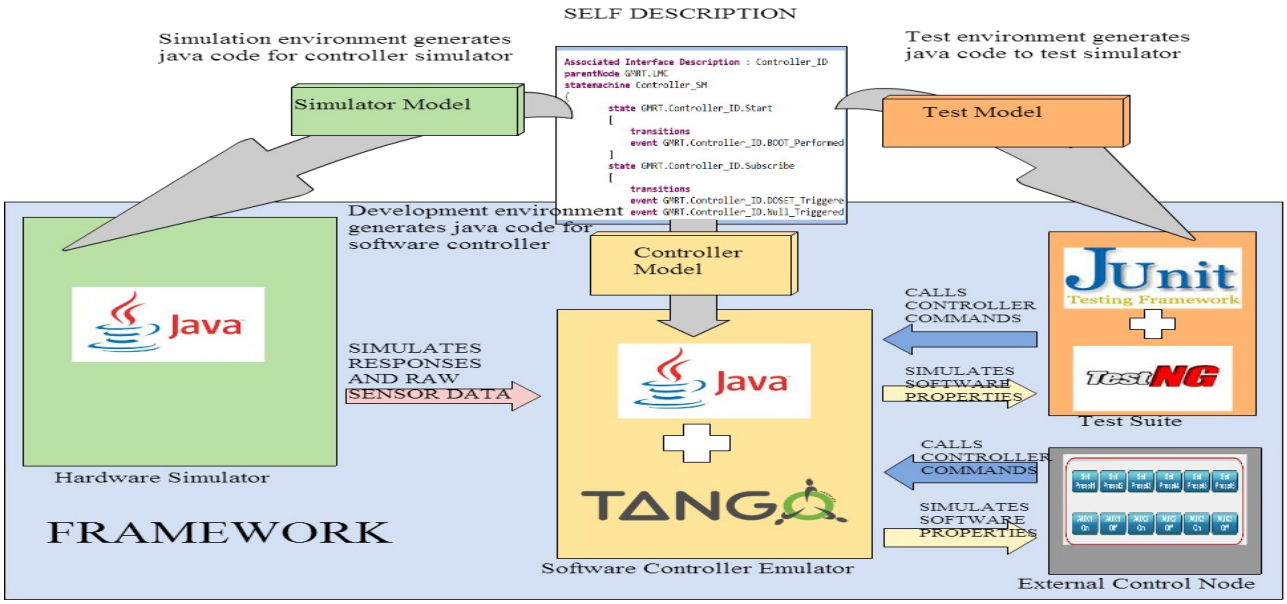


Figure 3 : Technology Stack and Internal Details.

- **Development Environment** – A controller model is instantiated by the development environment and captures the behavioral and skeletal properties of a controller node like commands, events, alarms, state machine logic etc. These properties are desired to create the software emulator for the controller node. This environment is developed as a plug-in for Eclipse using modeling tools such as EMF[6] and XText[7].

- **Simulation Specification Environment** – A simulator model is captured by the simulation specification environment and captures the specification required to simulate the behavior of the underlying hardware. The environment gathers specifications such as expected and unexpected responses to corresponding commands, selection of patterns to generate the sensor data such as sine wave and so on, the time lag to generate two consecutive sensor data values, the number of values to be skipped before generating the next sensor data value. These details inject realistic behavior into the generated simulator. Since these specification items do not form a part of the SDD of a controller they are captured separately through this environment This environment is implemented as a plugin for Eclipse using EMF framework and with building custom UI.

- **Test Specification Environment** – A test model gets instantiated by the test environment to capture the test scenarios involved to test the simulator itself. It reuses the information from the SDD such as expected response to commands, response validation rules and so on to automatically generate the test cases. It also injects some specific test conditions which force the simulator to act in a specified manner to obtain specific test results. The test environment is generic in the sense that it could also be used to test an actual controller and not necessarily only a controller simulator. This environment could incorporate algorithms to make sure that it generates all the exhaustive test cases to test a particular controller or hierarchy of controllers. This environment is implemented as a plug-in for Eclipse using EMF framework and with building custom UI.

After populating these models the environments automatically generates the code based on the instantiated models.

- **Controller/Software Emulator** – The controller emulator is generated by the development environment utilizing the details specified in the SDD. This controller simulator implements the functionality described in the SDD in form of an executable JAVA file utilizing the APIS' from the TANGO[8] framework.

- **Hardware Simulator** – The hardware simulator generated by the Simulator environment, incorporates the details captured by the Simulation model through the simulation specification environment. This too generates a JAVA file which contains mapping of the desired response to a command and the functionality to generate a raw sensor data as per the rules provided in the simulation model.

- **Test Suite** – The test suite is generated by the test environment implementing the test scenarios based on the populated test model. The generated Java code uses the Junit[9] framework API's and consists of multiple test cases for the simulator testing. Each test case is mapped to five different test conditions provided using the Data Providers. This makes the test cases run 5 times and tests the simulator for 5 different conditions. The test case tests the command of the controller by invoking it with 5 different input data and checks the corresponding response against an expected response collected as part of the test model. Hence for N commands of a controller the total tests performed will be 5N times.

The envetual goal for this test suite is to perform exhaustive testing of all features not just limited to commands. The test case also injects a specific test

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

condition where it forces the simulator to generate that specific response which is pre-populated by the test model, hence ensuring that every test case has at least one test condition which gets the desired response and passes the test case.

PROOF OF CONCEPTS

Giant Meter-wave Radio Telescope

As a PoC we have used the simulation and testing framework to simulate the controller nodes for one structural hierarchy of the GMRT as follows:-

In the GMRT control system, an Antenna Node controls an IF Controller Node, which in turn directly interacts with the underlying hardware device [provide reference to GMRT architecture]. In order to try out our framework, we assumed that the Antenna Node has been developed and needs testing of its functionality but the IF Controller is not yet developed With our framework, we could still create the self descriptions (SDD) of the IF Controller using our DSL. This file contained the behavioral and skeletal properties of the IF controller without having the implementation details and hence could be created in less than an hour.

Once the DSL is create we generated the controller software emulator for the IF Controller followed by generating the hardware simulator. The simulator model reuses some details in the controller SDD like Commands, Responses, States, Events etc. and also uses simulation specific input. We also generated the test cases for IF Controller Simulator which generates a report upon the execution of the test cases. . We needed to study the GMRT system a bit to come up with the required specifications. But once we had good understating of the features of the antenna node and the IF controller, the creation of the individual specs using our framework took less than an hour.

Using the simulation and test framework for creating a simulator for an IF Controller led us to the following observations:-

- Existing knowledge of the control system is required before creating the simulation specification.
- The use of the framework made it easy to generate the simulator as creating the specification required very less time (around 30 mins).
- Manually coding the simulator would have taken approximately 1-2 hrs, while with the framework it took only around 15-30 mins to do it, hence a 400 – 800 % efficiency in time consumption
- Test cases made it possible to test the simulation before actually plugging it in a real developed Antenna Controller.
- The execution of the test cases results into testing the dynamic behaviour of the simulator before it could be made to real use.

CONCLUSION

The MDE approach for implementing the simulation and test framework can definitely prove more efficient than the traditional way of manually developing individual simulators. However, the simulators generated will still need to be enhanced with appropriate domain logic Although the generated simulators currently are standalone and statically configured during compile time. we want to create an integrated approach which could make changes to the specification possible during its execution as well. We also look ahead to create better methodologies to generate test conditions which would target the acute cases for simulator testing. The simulation model now contains minimal necessary features, in future we wish to provide it with more features so that it would behave more realistically and show better performance.

ACKNOWLEDGEMENT

We would like to thank Dr. Swaminathan Natarajan, Dr. Alan Bridger and Dr. Yashwant Gupta for their continuous support in shaping our thoughts. Also thanks to the GMRT and SKA South Africa teams for providing their support.

REFERENCES

[1] MeerKAT CAM Design Description, DNo M1500-0000-006, Rev 2, 2013.
[2] A.R. Taylor, "The Square Kilometre Array", Proceedings IAU Symposium No. 291, 2012.
[3] Giant Metrewave Radio Telescope, www.gmrt.ncra.tifr.res.in
[4] KAT-7 (seven dish MeerKAT precursor array) http://www.ska.ac.za/meerkat/kat7.php
[5] Puneet Patwari, Subhrojyoti Roy Chaudhuri, Swaminathan Natarajan , G Muralikrishna : M&C ML: A Modeling Language for Monitoring and Control Systems.
[6] Eclipse Modeling Framework: http://www.eclipse.org/modeling/emf/
[7] Lorenzo Bettini, “Implementing Domain-Specific Language with XText and XTend”, (August 2013)
[8] The Tango Control System Manual Version 8.1 : www.esrf.eu/computing/cs/tango/tango_doc/kernel_doc/ds_prog
[9] JUnit : https://www.junit.org/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A SELF-CONFIGURABLE SERVER FOR CONTROLLING DEVICES OVER THE SIMPLE NETWORK MANAGEMENT PROTOCOL

V. Petrosyan, V. Rybnikov, DESY, Hamburg, Germany

Abstract

The Simple Network Management Protocol (SNMP) is an open-source protocol that allows manufacturers to utilize it for controlling and monitoring their hardware. More and more SNMP-manageable devices show up on the market that can be used by control systems for accelerators. SNMP devices are also being used at the free-electron laser (FLASH [1]) and planned to be used at the European X-ray Free Electron Laser (XFEL[2]) at DESY, Hamburg, Germany. To provide an easy and uniform way of controlling SNMP devices a server program has been developed. The server configuration, with respect to device parameters to control, is done during its start-up and driven by the manufacturer Management Information Base (MIB) files provided with SNMP-enabled hardware. The list of controlled parameters can also be tailored by user's configuration files.

The SNMP server integrates the various type devices into the Distributed Object-Oriented Control System (DOOCS [3]) used at FLASH and XFEL. Several SNMP devices of different types can be controlled by one SNMP server simultaneously.

SNMP AT A GLANCE

SNMP is an application-layer protocol defined by the Internet Architecture Board (IAB) in RFC1157 (May of 1990) [4]) for exchanging management information between network devices. It is a part of Transmission Control Protocol/Internet Protocol (TCP/IP [5]) suite. SNMP uses User Datagram Protocol (UDP) [6] for the transport layer. SNMP requests are transmitted as UDP datagrams over a connectionless transmission interface between SNMP manager and SNMP agent. Management Information Base (MIB) is a database containing information about elements to be managed. It represents Manageable Objects (MO) as resources. Every MO (device attribute) has a unique Object Identifier (OID) that can be used by queries. It also provides a map between numeric OIDs and a human-readable text.

SNMP agent is a daemon process running on a network device. It exposes device attributes to managers. SNMP managers control the attributes via Get/Set requests. Managers can have access to those attributes that are allowed by the agent. SNMP agents can also provide traps (asynchronous messages). Traps are mainly used for significant events in the device (e.g. power-off).

SNMP SERVER IMPLEMENTATION

The SNMP server is implemented as a DOOCS server.

It allows to integrate SNMP devices into FLASH and XFEL control systems in a transparent way. The DOOCS SNMP server as an SNMP manager provides access to attributes of the controlled device by means of DOOCS properties.

Server Design

The server uses the NET-SNMP library [7] that provides all functionality required to work with SNMP agents. The library allows:

- to retrieve the list of attributes controlled by an SNMP agent,
- read and write attributes values.

The list of attributes can be acquired starting from a certain node of the MIB tree. This node is defined by one DOOCS property of the SNMP server. Once the full list of attributes is available for the SNMP server it filters out the attributes of interest and creates for every selected SNMP attribute a set of DOOCS main (see Table 1) and optional (see Table 2) properties. Every property name in the set starts with the SNMP attribute name with upper case letters. The filtering of attributes is implemented via white or black lists provided by users.

Reading/writing an SNMP attribute value is implemented via a dedicated DOOCS property. There are two ways to read the attribute value:

- periodically via the server internal update timer,
- immediately on access to the DOOCS property.

Writing the attribute value is done on the corresponding DOOCS property (NAME) change.

Table 1: Main DOOCS Properties for SNMP Attribute

Name	Purpose
NAME.MIB	MIB attribute name
NAME.DESC	Attribute XML description
NAME.ACCESS	Access mode (e.g. read-only)
NAME.TYPE	Data type (e.g. ASN_INTEGER)
NAME	Value of the attribute
NAME.SL	List of valid values to write

According to “Structure of Management Information Version2” (SMIv2, RFC2578 [8]) the data types used by SNMP agents don't include floating point or double type.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Instead, scaled integers are used to show non-integer values (e.g. temperature 25.7 °C can be expressed as 2570). For such SNMP attributes a DOOCS polynomminal-parameter type can be used for converting the integer value to the corresponding float value. In addition, an archiving history can be provided for SNMP DOOCS property values. This allows to see the value changes over a long time period.

Table 2: Optional DOOCS Properties for SNMP Attributes

Name	Purpose
NAME.HIST	Value history
NAME.FILT	Filter parameters for history
NAME.FLOAT	Attribute value as float
NAME.POLYPARA	Conversion parameters to float (a0+a1*V+ a2*V^2)
NAME.ALARM	Alarm flag
NAME.LIMIT	Value for alarm limit
NAME.ALARM.OP	Operation for comparison
NAME.ALARM.OP.SL	List of available operations
NAME.ALARM.MASK	Flags to work with alarm

The DOOCS SNMP server provides an alarm mechanism to ease the device monitoring by operators. Every SNMP attribute can be configured for an alarm detection. An alarm will be triggered if a comparison of the current value of the attribute and some user predefined limit against one of logical operations gives a true value. The attribute alarm will propagate up to the special SNMP DOOCS server 'ALARM' location. This location sums up all incoming alarms and exposes the global server alarm via the value of its ALARM property.

IN OPERATION

On start-up the SNMP server reads out its DOOCS configuration file. The file contains all information required for the communication with an SNMP agent (see Table 3)

If during the start-up the SNMP agent is not reachable, the server will check the availability of the agent with some predefined time period. Once the agent is found, the sets of DOOCS properties for the required SNMP attributes will be created. After that the server monitors the attribute values either only periodically with its predefined period or additionally on a DOOCS request to the SNMP server from a DOOCS client (e.g. user panel).

One SNMP server can work with several SNMP devices of different types simultaneously. One DOOCS SNMP location is dedicated to a group of attributes within one SNMP agent. It means that several SNMP locations of one SNMP server can control several groups of attributes from one SNMP agent as well as from several independent SNMP agents. Due to the multi-threaded [9] implementation of the SNMP server the interactions with SNMP agents within the same server are completely independent.

Table 3: DOOCS Properties for SNMP Communication

Name	Purpose
SNMP.HOSTNAME	SNMP agent host
SNMP.VERSION	SNMP version
READ_COMMUNITY	SNMP read community name
WRITE_COMMUNITY	SNMP write community name
SNMP.TIMEOUT	SNMP maximum response delay time
MIBDIR	Directory with MIB files
MIBFILE	First MIB file name to use
NODE	SNMP object name to start the attributes search

Adding a new SNMP device into the SNMP server doesn't require a server re-start. A new SNMP DOOCS location is dynamically created by a DOOCS request to the server. To complete the new SNMP device configuration the user has only to set the DOOCS properties from Table 3 with the help of a panel (see Fig. 1). Once all properties are filled with values the SNMP server tries to reach the device.

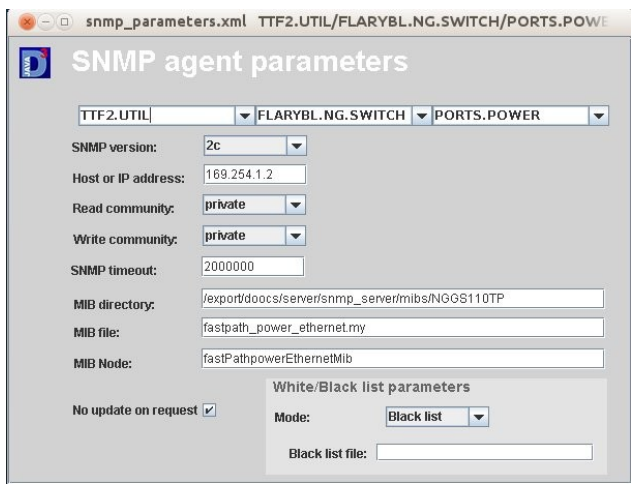


Figure 1: Panel for configuration of an SNMP agent

White/Black Attribute Lists

Usually an SNMP object tree contains many objects and attributes. To select only those that a user is interested in, a mechanism of white/black list is used. A dedicated DOOCS property defines whether the white or black list mode will be applied. A positive value means

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

the white list mode, negative – black list mode, zero – none of the lists will be used. The last case can be used for browsing the full SNMP tree of the device.

In case of the white list mode, the SNMP server creates DOOCS properties only for those SNMP attributes which names match the patterns from the white list. In case of the black list mode, the SNMP attributes with names, that match the patterns from the list, will be ignored.

Alarm Configuration

The white lists can also be used for the alarm configuration. In this case the line in the list contains three parameters:

- attribute pattern,
- operation for alarm detection (<, <=, =, !=, >=, >),
- alarm limit.

A few optional DOOCS properties are created for attributes with alarms (see Table 2). They provide full control over the attribute alarm configuration:

- changing operation for comparison,
- changing the limit value,
- masking or resetting the alarm.

The last operation allows to exclude malfunctioning devices from the ALARM monitoring without removing the device from the SNMP server.

Server Alarm

A dedicated DOOCS location is used for the alarm indication. Its ALARM DOOCS property is set to non zero value if there is at least one SNMP attribute with the non-masked alarm.

This tree-like propagation schema allows to build simple GUIs for monitoring alarms in many SNMP devices controlled by one SNMP server.

SERVER APPLICATIONS

NETGEAR GS110TP

Many network switches like the NETGEAR GS110TP [10] are exploited at DESY. They are used for integrating network cameras into different diagnostics systems. The switch (see Fig. 2) offers eight 1 Gigabit ports with Power-Over-Ethernet (PoE) [11] capability.

Figure 2: Front panel of GS110TP switch

The switch has an SNMP agent that provides a rich set of switch attributes. The most importand switch attributes with respect to camera operations have been chosen for the SNMP server. They are:

- PoE enable,
- PoE status,
- Power Consumption,
- Temperature etc.

An easy access to PoE-enabled attributes of the switches via the SNMP server allows to perform a remote power reset for the connected cameras.

Raritan's Smart Rack Controller

Raritan's environment rack controller [12] is an IP-based appliance that allows to use any of Raritan's sensors such as temperature, humidity, air pressure and airflow. EMX2-111 (see Fig. 3) is used for performing the climate control of the Schroff racks (VARISTAR LHX20/40 [13]) at the FLASH and XFEL facilities.

All monitored attributes are accessible via an SNMP agent. The device parameters are split into two groups. Every group is controlled by one location of the DOOCS SNMP server. The first location is responsible for the rack climate parameters (humidity and temperature sensors of the rack). The second one gets the information from the cooling equipment (air and water temperature, speed of fans). DOOCS archiving histories are provided for all attributes.

Figure 3: Raritan's smart rack controller

Rittal's LCP-Smart and CMC Rack Climate Control

Rittal's Liquid Cooling Package (LCP smart) [14] is a rack system that serves to dissipate high heat losses and for effective cooling of devices built into the enclosure. The package is equipped with a control unit (Basic CMC). Its task is to monitor all connected sensors and to control the cooling devices (fans, etc.) in order to reach the temperature set points. CMC can be controlled remotely via the embedded SNMP agent.

Along with LCP smart, CMC rack climate control devices (see Fig. 4) are being used at FLASH and XFEL . They are:

- CMC-TC processing unit [15],
- CMC-TC climate unit [16],
- CMC-TC I/O unit [17].

Equipped with the required sensors and cooling devices they perform the climate control in a rack. CMC-TC processing unit has an SNMP agent. Only one DOOCS location is used for managing one Rittal's control unit attributes available for the SNMP server.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Figure 4: CMC-TC processing , I/O and climate units

STATUS

The SNMP server design allows to deploy the server for controlling any device running an SNMP agent. One SNMP server can operate several SNMP agents of different devices simultaneously.

The white/black list approach for SNMP attributes selection is proven to be very powerful to reduce the total number of properties to operate on. It allows the operator to focus only on the paramerters required by users.

Having a set of DOOCS properties per one SNMP attribute provides a flexible way of controlling and monitoring of the SNMP attribute. It also allows to build the smart operator GUIs (e.g. based on jDDD [18]) that can provide all detailed information about the SNMP attribute (e.g. type, description, access, allowed values to write etc.).

The-tree like ALARM propagation schema offers a very simple way of monitoring alarms in all SNMP devices controlled by one SNMP server.

Twelve instances of SNMP servers are currently used in different diagnostic systems exploiting network cameras with the PoE capability. The DOOCS camera software [19] makes use of the SNMP server to perform automatic camera recovery in case of the camera disconnection is detected.

FLASH and XFEL exploit one SNMP server each for controlling currently used racks. Every server controls all available racks within the facility. At FLASH: Schroff – 3, CMC – 9, at XFEL: Schroff – 3, CMC – 5. The final number of CMC racks at XFEL is about 190 pieces. It is not yet decided how many SNMP servers will be used to control the parameters of all XFEL racks.

REFERENCES

[1] http://flash.desy.de
[2] http://xfel.desy.de
[3] K. Rehlich, “Status of the TTF VUV-FEL Control System”, PCaPA2005 proceedings, Hayama, Japan; http://conference.kek.jp/pcapac2005/paperindex.html
[4] https://datatracker.ietf.org/doc/rfc1157/
[5] W.Richard Stevens, “UNIX Network Programming” , (Prentice Hall, 1990), 199.
[6] W.Richard Stevens, “UNIX Network Programming” , (Prentice Hall, 1990), 200.
[7] http://www.net-snmp.org/
[8] https://datatracker.ietf.org/doc/rfc2578/
[9] Steve Kleiman, Devang Shah and Bart Smaalders, “Programming with Threads”,(Prentice Hall, 1996)
[10] http://support.netgear.com/product/GS110TP
[11] IEEE 802.3af, IEEE 802.3at
[12] http://www.raritan.com/products/environmental-monitoring/emx-smart-rack-controller
[13] http://www.varistar.de/src/varistar_lhx20_d.pdf
[14] http://www.rittal.com/
[15] http://www.rittal.com/imf/none/3_407/
[16] http://www.rittal.com/imf/none/3_413/
[17] http://www.rittal.com/imf/none/3_409/
[18] G. Grygiel, V. Rybnikov, “DOOCS Camera Server”, ICALEPCS07 proceedings, Knoxville, Tennessee, USA, October 2007, p.359 (2007)
[19] E. Sombrowski et al., “jddd: A Tool for Operators and Experts to Design Control System Panels”, ICALEPCS2013 proceedings, San Francisco, CA, USA, October 2013, p.544 (2013)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

HIGH LEVEL SOFTWARE STRUCTURE FOR THE EUROPEAN XFEL LLRF SYSTEM

Christian Schmidt*, Valeri Ayvazyan, Julien Branlard, Łukasz Butkowski, Olaf Hensler, Martin Killenberg, Mathieu Omet, Sven Pfeiffer, Konrad Przygoda, Holger Schlarb, DESY, Hamburg, Germany
Adam Piotrowski, FastLogic sp. z o.o., Łodz, Poland
Wojciech Cichalewski, Filip Makowski TUL-DMCS, Lodz, Poland

Abstract

The Low Level RF system for the European XFEL is controlling the accelerating RF fields in order to meet the specifications of the electron bunch parameters. A hardware platform based on the MicroTCA.4 standard has been chosen to realize a reliable, remotely maintainable and high performing integrated system. Fast data transfer and processing is done by field programmable gate arrays (FPGA) within the crate, controlled by a CPU via PCIe communication. In addition to the MicroTCA.4 system, the LLRF comprises external supporting modules also requiring control and monitoring software. In this paper the LLRF system high level software used in E-XFEL is presented. It is implemented as a semi-distributed architecture of front end server instances in combination with direct FPGA communication using fast optical links. Miscellaneous server tasks have to be executed, e.g. fast data acquisition and distribution, adaptation algorithms and updating controller parameters. Furthermore the inter-server data communication and integration within the control system environment as well as the interface to other subsystems is described.

INTRODUCTION

The Deutsches Elektronen-Synchrotron (DESY) in Hamburg is currently building the European X-ray Free Electron Laser (E-XFEL) [1]. This hard X-ray light source generates up to 27000 coherent laser pulses per second with a duration of less then 100 fs and a wavelength down to 0.05 nm. For this, electrons have to be accelerated to 17.5 GeV using a 2 km particle accelerator based on superconducting radio frequency technology. Precision regulation of the RF fields inside the accelerating cavities is essential to provide a highly reproducible and stable electron beam. RF field regulation is done by measuring the stored electromagnetic field inside the cavities. This information is further processed by the feedback controller to modulate the driving RF source, using a low level RF system. Detection and real-time processing are performed using most recent FPGA techniques. Performance increase demands a powerful and fast digital system, which was found with the Micro Telecommunications Computing Architecture (MicroTCA.4) [2]. Depending on the application either parallel processing for latency efficiency or pipelined processing of a large number of similar RF signals is aspired. In-tunnel installation with limited access requires a compact system with implied redundancy. Further remote access to status information, software upgrades and maintenance is ensured. Finally the modularity and scalability of this system guarantees to have later upgrades in order to meet demands within a lifetime of the machine and beyond. DESY currently is operating the Free Electron Laser (FLASH), which is a user facility of the same type as E-XFEL but at a significantly lower maximum electron energy of 1.2 GeV. The LLRF system for FLASH is equal to the one of E-XFEL, which allows for testing, developing and performance benchmarking in advance of the E-XFEL commissioning [3]. A picture of the FLASH installation can be found in Fig. 1.

Figure 1: Installation of MicroTCA crates inside the FLASH tunnel underneath the accelerator module. The radiation shielding (yellow cabinet) protects the electronic rack inside. The insert shows the MicroTCA crate installed in this rack.

LLRF SYSTEM LAYOUT

The present LLRF system for a single E-XFEL RF station controlling 32 cavities is given in Fig. 2. LLRF systems with a different quantity of cavities are build as a subset of this configuration. Machine operation is performed in a pulsed mode. The duty cycle, i.e. the repetition rate (10 Hz) of the RF pulse to the RF pulse length (about 1 ms) is about 1 %. The impact to the RF regulation is such that within in a regular mode of unchanged operating conditions, i.e. stable working point consecutive pulses are similar, however the initial conditions vary as well as long term changes occur. In

* christian.schmidt@desy.de

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

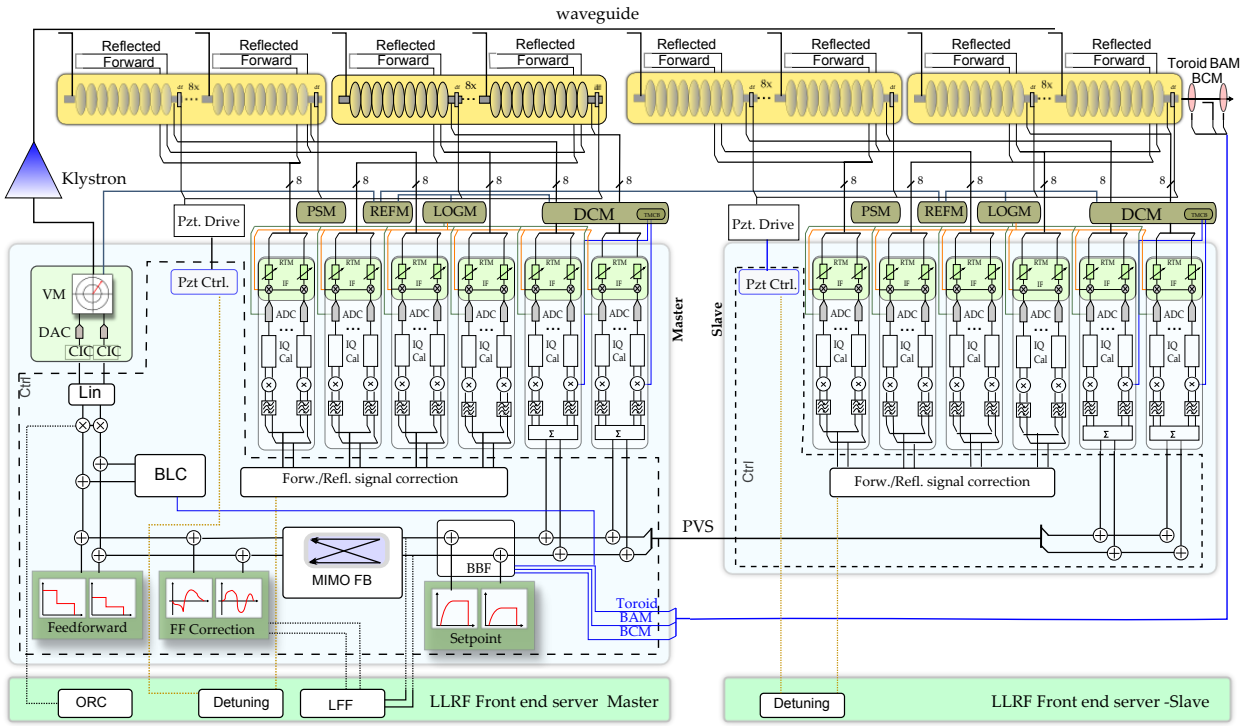


Figure 2: Basic layout of the LLRF station for the master and slave control used in E-XFEL. Signal processing path as well as communication links are presented. Communication between the master and slave system is done using optical links, as well as for connecting further subsystems to the LLRF (marked as BAM, BCM and Toroid).

order to optimize the field regulation a temporal separation of feedback is applied in the following way:

- *µs* scale: intra pulse feedback and digital signal processing on the FPGA
- *ms* scale: pulse to pulse adaptation, removing repetitive control errors, processed in the CPU
- *s* scale: long term drift compensation and slow feedback mechanism by external modules or feedbacks

Intra Pulse Data Processing

Fast regulation steps are processed within the FPGA. Furthermore the fast regulation loop is separated into a preprocessing step, running on a different FPGA than the main controller application, which collects data from several preprocessing boards using low latency communication links in the backplane of the MicroTCA.4 crate. The 16 cavity data acquisition and preprocessing section is identical for the master and the slave subsystem. The main controller sums up the two partial vector sums (PVS), resulting from the contribution of cryomodule (CM) 1 and 2 on the master LLRF system and CM 3 and 4 on the slave LLRF system. This total vector sum is processed within a multiple input, multiple output (MIMO) feedback controller [4] and added to the feed-forward signal in order to to generate the klystron drive. Further beam based information transferred through optical links are processed in order to improve the regulation performance (BBF) or compensate effects like beam loading (BLC).

Pulse to Pulse Adaption

Repetitive control errors such as generated by cavity detuning, are rather compensated by some additional feed-forward input instead of the feedback. An model based iterative learning feed-forward algorithm (LFF) processing data from pulse to pulse on the CPU is used to play some correction signals in order to relief the action required from the feedback. Changes in the actuator chain are visible in offsets of the correction signals and are compensated by automatic scaling and rotating the output vector (ORC).

Long Term Drift Suppression

Long term drifts, especially in the measurement chain are currently the most critical part in the regulation loop. There exist different approaches to compensate for this, e.g. reference signal tracking or passive methods like ensuring stable environmental conditions. Here a special drift compensation module (DCM) is used which measures a priori the RF pulse a reference pulse and corrects the RF signals for each channel accordingly. Further beam based measurements are used to quantify the control error with respect to the electron beam and slowly adapt the operating conditions by changing LLRF setpoints.

The combination of these feedback mechanisms is the basis to meet the given RF field regulation requirements of

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

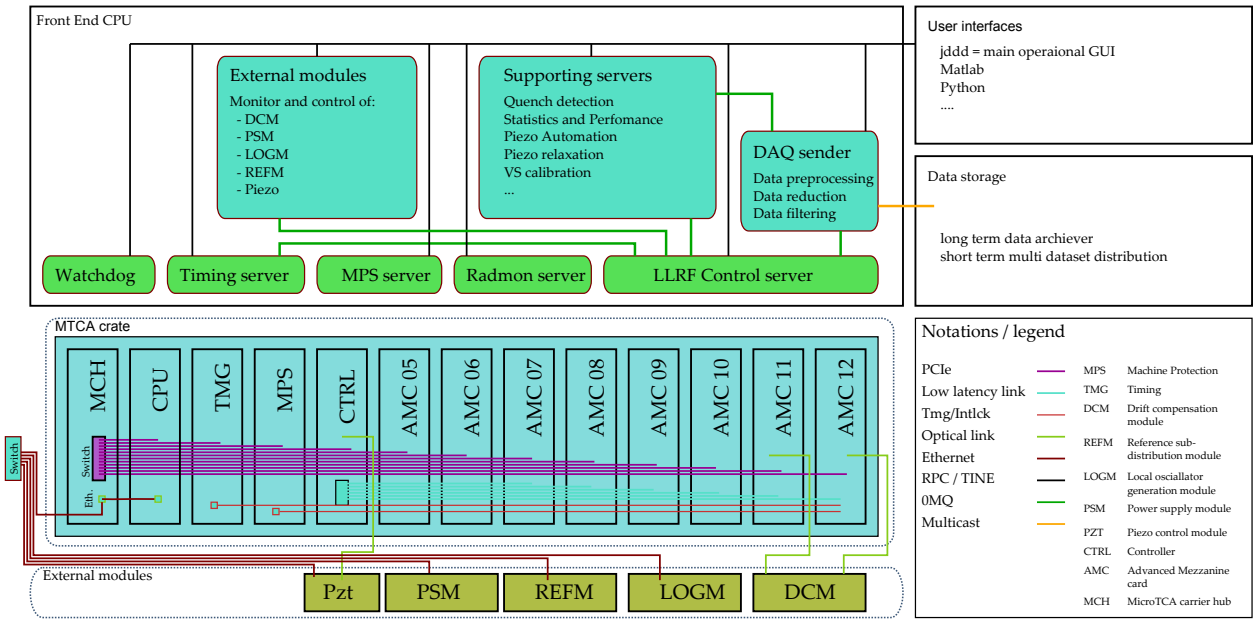


Figure 3: Front end server structure and communication links within the MicroTCA.4 LLRF system.

0.01 % and 0.01 deg in amplitude and phase. Control and communication with the firmware is based on the LLRF front end server.

ARCHITECTURE OF THE LLRF FRONT END SERVERS

Having a pulsed mode of operation requires to synchronize all subsystems to a frequent trigger. This is realized by a timing distribution system which triggers all subcomponents within the crate at a given time. Further the servers running on the front end CPU are triggered by the same mechanism. This synchronization puts the subtasks within one RF pulse in line which is: starting the RF pulse, processing data within the FPGA, transferring data to the CPU, processing data and further sub-distribute, update control variable withsin the FPGA. Communication between the FPGA and the CPU is done by direct memory access over a PCIe link [5]. Fixed point representation from the FPGA is converted to physical meaningful data before further distributing the data to graphical presentation, other processing tasks or data storage. Inter process communication on the same CPU is realized using the Zero*MQ* communication protocol [6]. Java based DOOCS is in charge of the graphical user interface [7]. Data communication to the GUI is done by RPC or TINE calls, which is further extended to use third party display and processing programs like Matlab. A sketch of the inter-server communication topology, inter-crate communication links and user interfaces can be found in Fig. 3.

The LLRF control server itself can be seen as the basic interface to the MicroTCA.4 hardware such that communication to front end registers is being controlled by this particular device. Data acquisition and pre-processing is a high priority task and should be treated as quasi real time capable. Therefore other tasks displayed as supporting servers will have a lower priority since the system is still operable even without having them running. The DAQ sending device is collecting data from the LLRF control server before transmitting them further to a data storage. Here the data from all subsystems is synchronized and stored for later analysis or directly used in higher level automation routines. This data storage is extremely helpful for post mortem analysis of system malfunctions.

Decoupling of the basic LLRF control server from other supporting servers turned out to be rather helpful in terms of maintenance and further developments. Having the front end server optimized at a given state it usually requires only minor adaptations during the running process. Further changes in the functionality of the server should be ideally transparent to other clients, which can be done except of interface changes.

Extensions and additional status and processing information is likely often adapted to new requirements. Maintaining such systems does not directly influence the main controller and also bugs usually induced by new developments are transparent to third party clients. Having defined clear interfaces allows to have a modular mechanism. For example the same LLRF front end server can be used for different subsystems even if additional supporting servers are not required or wanted. This reduces the overall load on the processing unit by reducing the number of subtasks to the required minimum. The goal of having a highly modular system for the MicroTCA.4 hardware and its corresponding firmware also reflects on the software architecture.

Basic Front End Servers

The basic idea for a front end server is to have one instance capable to communicate with the hardware device installed in the system. For example comparing Fig. 3, the devices

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

of timing control (TMG), machine protection (MPS) are single server - single device pairs. For the LLRF control server there are in total 7 devices controlled by one server which can be seen as one combined system. This comparison reflects the complexity and large amount of data processing taking place. Reducing the functionality of this server to basic functionalities is a consequence to distribute tasks in an effective way.

Supporting Servers

The subcategory of supporting servers contains several functionalities separated into frequently running instances and user triggered applications. As an example the routine of automatic piezo tuning should be described here. In order to minimize the cavity detuning, piezo actuators are used to play a mechanical stimulus to compensate to RF field induced deformations of the cavity. This is done by computation of the actual detuning within the statistics and performance server. Necessary data for this computation is send from the LLRF control server as described above. After the detuning parameters are computed, they are further send to the piezo automation server which computes the waveform to be played. This is transferred back to the LLRF front end server which sends this waveform to the firmware registers as a control command sequence for the next pulse. A different example can be found with the VS calibration instance, which is triggered unfrequent by user requests in order to recalibrate the measurement chain of the LLRF system. Permanent operation of this server is not possible since the calibration method is invasive to machine operation and usually requires a special setup.

External Device Servers

Within the LLRF system there exist next to the MicroTCA.4 crate supporting components which also require to be controlled and monitored. The setup varies with the different locations the system is implemented. Each of this system requires a server, however there are many functionalities within which can be transferred in between. All these servers share special classes which can be plugged together in a modular way. Communication is done over ethernet which would basically allow to run the instances also on a different CPU if high load on the front end CPU would require this.

CONCLUSION AND OUTLOOK

In this paper the MicroTCA.4 based LLRF control system in view of the LLRF software structure for the E-XFEL has been presented. Basic layout ideas and structuring as well as communication paths are discussed. This architecture is currently operating in the FLASH facility, meeting the required goals in terms of RF field stability. However the basic functionalities are successfully running, continues developments are ongoing which further improve the overall system performance, as well as the maintainability and reliability.

Currently all acceleration modules being installed in the E-XFEL are tested and characterized in advance. Part of this program is done by the LLRF system since gained information will be used for later automation routines. Storage of this data is done by a database which requires automatic acquisition and load to the systems being installed in the E-XFEL. Furthermore special automation routines developed for the tests are directly transferable to E-XFEL and also to FLASH.

Fast error tracking helps to minimize machine downtime. Therefore status and health information is monitored for all subsystems, however automatic processing this information is essential to support operators with the key information instead of a large amount of error notifications. Systematic failure analysis methods are currently under development.

Finally the LLRF system containing hardware, firmware and software are going to be used in other facilities which might use different control systems. To integrate this in a different environment, a control system independent server is developed, which decouples functionalities of the server from requirements of the control system [8]. All these modifications demand restructuring work, however the modularity of the systems as described in this paper helps to minimize the effort and reduces the risk of system malfunctions.

REFERENCES

[1] *The European X-Ray Free Electron Laser Technical Design Report*, `http://xfel.desy.de`

[2] MicroTCA® is a trademark of PICMG, MicroTCA.4 specifications: `http://www.picmg.org`

[3] J. Branlard *et al.*,*The European XFEL LLRF System*, Proceedings of the 3th International Particle Accelerator Conference 2012, New-Orleans, USA, 2012.

[4] C. Schmidt *et al.*, *Precision Regulation of RF Fields with MIMO Controllers and Cavity-based Notch Filters*, Proceedings of the XXVI International Linear Accelerator Conference 2012, Tel-Aviv, Isreal, 2012.

[5] M. Killenberg et al., "Drivers and Software for MicroTCA.4", WEPGF015, *These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

[6] http://zeromq.org

[7] E. Sombrowski, R. Kammering, K.R. Rehlich , *A HTML5 Web Interface for JAVA DOOCS Data Display*,*These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

[8] M. Killenberg et al., "Integrating control applications into different control systems", TUD3O05, *These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE EPICS ARCHIVER APPLIANCE

Murali Shankar, Luofeng Li, SLAC, Menlo Park, CA, USA*
Michael Davidsaver, BNL, Upton, New York, USA
Martin Konrad, National Superconducting Cyclotron Laboratory, Michigan State University, East Lansing, MI 48824, USA**

Abstract

The EPICS Archiver Appliance was developed by a collaboration of SLAC, BNL and MSU to allow for the archival of millions of PVs, mainly focusing on data retrieval performance. It offers the ability to cluster appliances and to scale by adding appliances to the cluster. Multiple stages and an inbuilt process to move data between stages facilitate the usage of faster storage and the ability to decimate data as it is moved. An HTML management interface and scriptable business logic significantly simplify administration. Well-defined customization hooks allow facilities to tailor the product to suit their requirements. Mechanisms to facilitate installation and migration have been developed. The system has been in production at SLAC for about 2 years now, at MSU for about 2 years and is heading towards a production deployment at BNL. At SLAC, the system has significantly reduced maintenance costs while enabling new functionality that was not possible before. This paper presents an overview of the system and shares some of our experience with deploying and managing it at our facilities.

REQUIREMENTS

At SLAC, BNL and MSU, we use EPICS as our control system for our various facilities; many of these have several million Process Variables (PVs) that are used for monitoring and control. A common requirement is to be able to archive some of these PV's to facilitate troubleshooting and analysis.

Archiving a million PVs has some non-technical requirements. In addition to scaling gradually by adding additional appliances/hardware, we need better support for managing all aspects of the system. This includes establishing policies for archiving and the flexible configuration of archiving on a per PV basis. The ability to manage the system using a web based UI and from within scripts is a necessity. These management functions must include the ability to make changes to the PV archiving configuration without having to restart the system. Finally, we needed a simple migration path from the ChannelArchiver [1, 2].

In terms of storage, the data rates are high enough to require faster storage but the data volumes are also large. In addition, the archivers require the storage to be always available. A solution using multiple stages of storage is most economical. The first stage is expensive fast storage local to the appliance and thus always available. The final stages of storage can be much slower and cheaper with relaxed requirements for availability including support for tape storage. The final design should be extensible enough to support storage from external vendors like S3.

Our customers also wanted a solution that focused on data retrieval performance. The most common use of archive data is to help trouble shoot machine operations. Thus, the bulk of the data retrieval requests are for recent data. Data can then be moved off to cheaper, slower storage once it has aged. Our goal was to be able to retrieve a day's worth of 1Hz double data in less than 0.5 seconds; the actual retrieval numbers are much faster. In addition, support for various forms of data reduction during data retrieval is critical.

ARCHITECTURE

The EPICS Archiver Appliance uses an appliance model for deployment. An installation is a cluster of appliances. Each appliance (see Fig. 1) has multiple storage stages and multiple processes.

A wide variety of storage configurations are possible (on a per PV basis). The out-of-the-box configuration has these storage stages

- STS (Short term store) - The most recent couple of hours worth of data is typically stored here. This is typically a RAM disk.
- MTS (Medium term store) - The most recent couple of days worth of data is stored here. At SLAC, we use RAIDed 15k SAS drives for the MTS. At MSU, these are mirrored 1.2TB SAS drives.
- LTS (Long term store) - The rest of the data is stored here. At SLAC, this is bulk storage (with tape backups) that we rent from our computing dept. This is a GPFS filesystem located elsewhere and is mounted over NFS. At MSU, this is a NetApp alliance with 2.8 TB of storage.

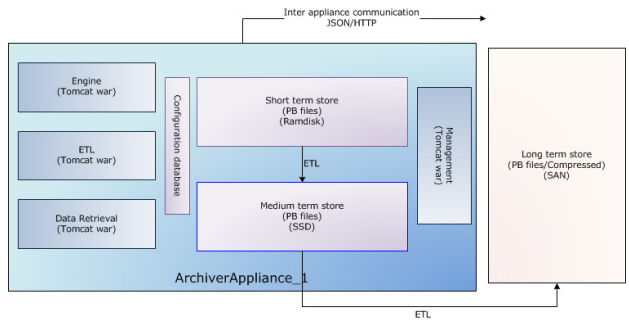


Figure 1: Components of an appliance.

* Work is supported by the U.S. Department of Energy, Office of Science under Contract DE-AC02-76SF00515 for LCLS I and LCLS II
** This work was supported in part by the National Science Foundation under the Cooperative Agreement PHY-11-06007

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Each appliance has 4 processes; these are J2EE WAR files and are deployed on separate Tomcat containers.

- Engine - This component establishes EPICS Channel Access monitors for each PV in the appliance. The data is written into the STS. This component is based on the CS-Studio engine.
- ETL - This component moves data between stores - that is, it moves data from the STS to the MTS and from the MTS to the LTS.
- Retrieval - This component gathers data from all the stores, stitches them together to satisfy data retrieval requests.
- Mgmt - This component executes business logic, manages the other three components and holds runtime configuration state.

Appliances and components talk to each other using various means including JSON/HTTP. The archiving configuration is typically stored in a MySQL database; each appliance has its own configuration database.

SERIALIZATION

The configuration database has an entry for each PV; this is a JSON object that has the details on how the PV is being archived. This includes the sampling mode and rate; it also includes a list of data stores (see listing 1).

```
"dataStores": [
   "pb://localhost?name=STS&rootFolder=...&partition
   Granularity=PARTITION_HOUR...",
   "pb://localhost?name=MTS&rootFolder=...&partition
      Granularity=PARTITION_DAY...",
   "pb://localhost?name=LTS&rootFolder=...&partition
      Granularity=PARTITION_YEAR..."
],
```

Listing 1: Each PV has a list of datastores.

Each data store is implemented using a storage plugin; these are Java objects that implement standard interfaces. The out of the box install uses the PlainPBStoragePlugin; this plugin uses Google's ProtocolBuffers (PB) [3] as the serialization mechanism. ProtocolBuffers provide a future-proof serialization framework with bindings for many languages. Each EPICS V3 DBR type is mapped to a distinct PB message. For EPICS V4 [4], NTScalars and NTScalarArrays are mapped to their V3 counterpart PB messages. All other V4 types are serialized using V4 serialization and stored as generic PB messages; thus giving the archiver the ability to store any V4 type.

The PlainPBStoragePlugin stores its data in chunks; each chunk contain serialized PB messages; one message per archive sample; one sample per line. In addition, the samples are ordered by their record processing timestamps; these are guaranteed to be monotonically increasing. Each chunk has a key that is based on the PV name and the time partition of the chunk; for example, EIOC/LI30/MP01/HEARTBEAT:2012_08_24_16.pb. Partition boundaries are strictly enforced; thus, the chunk key has enough information to identify the boundaries of the contained data. By default, the PlainPBStoragePlugin stores its data in files; one file per chunk with the chunk key as the file name.

These various constraints let us use various search algorithms on PB files without the need for an index. However, in the future, if an index is needed for certain PV's, the system can be enhanced to use indexes for these PV's. PB files try to optimize on storage consumption. On average, a PB ScalarDouble consumes about 21 bytes per sample to store the timestamp, value, status and severity along with several other optional fields.

ARCHIVE PV WORKFLOW

Supporting complex configuration on a per PV basis can be overwhelming if the end user had to make these decisions every time they add a PV to the archiver. In addition, many facilities use automated scripts to manage their archiving requests. This implies that the system must automatically make these decisions on behalf of the user.

When users request PVs to be archived, the mgmt and engine components sample the PV to determine event rate, storage rate and other parameters. In addition, various fields of the PV like the NAME, ADEL, .MDEL, .RTYP etc. are also determined. All of these parameters are passed to an installation specific policy that is implemented as a Python script. The policy examines this information and makes configuration decisions on behalf of the user. For example, you can establish and encode a policy to archive waveforms at a rate no more than a 1Hz.

Optionally, as part of a policy, we can also archive fields in addition to the VAL field. For example, one can establish a policy to archive the HIHI, LOLO in addition to the VAL field for all records of RTYP ai. These fields are stored as part of the data for VAL field.

A clustered solution also implies that a decision must be made as to which appliance is used to archive a PV. To help with this decision, the archivers maintain capacity metrics and use a minimax algorithm to automatically assign PVs to appliances.

DATA RETRIEVAL

The EPICS Archiver Appliance comes with plugins for the ArchiveViewer and the CS-Studio databrowser [5]. We are also currently working on a HTML5 viewer (Fig. 2) for archive data; this is bundled as part of the appliance.

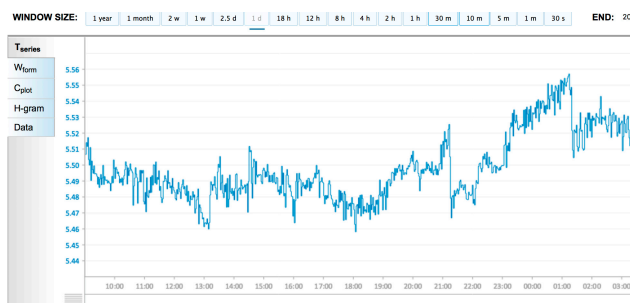


Figure 2: HTML5 viewer (in development).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A principal focus is data retrieval performance; the following chart (see Fig. 3) shows response times for data retrieval requests from data gathered over several weeks from a production system. Data retrieval requests are categorized in terms of their time span (endtime - starttime). Over 75% of the requests are for time spans less than a day and complete within 100ms (on an average). Requests with time spans of up to a week take an average of 250ms. Requests for time spans of a year with data reduction can complete in less than a couple of seconds; this involves binning over 30 million samples into about 8000 samples at runtime.

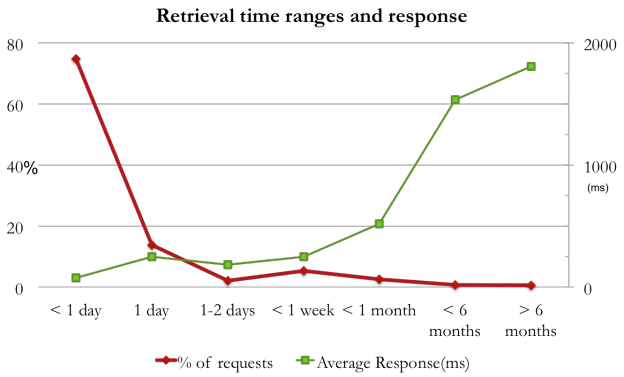


Figure 3: Response times vs time span of requests.

The EPICS Archiver Appliance supports data retrieval over HTTP in multiple formats/MIME types. A binary RAW format is supported that minimizes translation from the PB format. There are Java and Python libraries for clients interested in retrieving data using the RAW format. The carchivetools suite [6] includes command line Python scripts that retrieve data using the RAW format from the appliance. It also includes tools for searching. In addition, we have support for JSON, CSV, MAT, TXT and SVG MIME types. These formats let clients use tools like Matlab, Python, Excel, JMP and others to get data from the archiver.

Getting data into a tool necessitates construction of a data retrieval URL as the first step; this URL contains only the PV name and the start and end times of the data retrieval request. Both data retrieval and business logic requests can be dispatched to any random appliance in the cluster (Fig. 4); the appliance has the functionality to route/proxy the request accordingly.

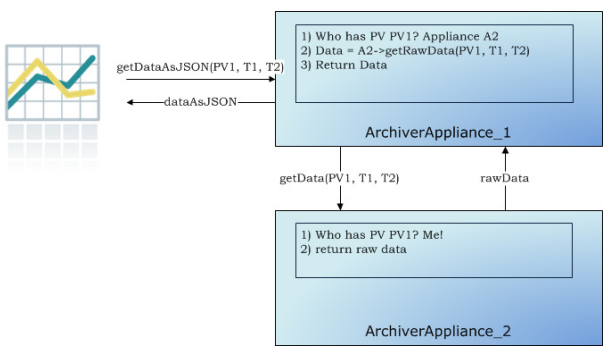


Figure 4: Requests can be dispatched to any appliance.

This enables us to use load balancers like mod_proxy_balancer (Fig. 5) in front of all the appliances in the cluster.

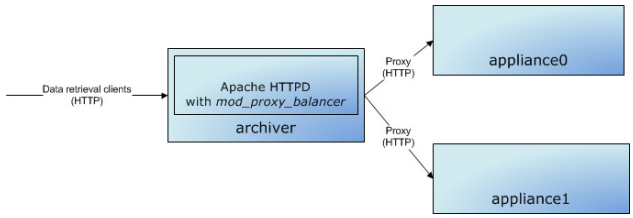


Figure 5: Traffic mgmt between clients and appliances.

Reducing data during data retrieval is a crucial requirement; however, there is no single correct way to reduce data. The EPICS Archiver Appliance has support for processing the data during data retrieval. This includes typical statistics operators like mean, standard deviation etc. These operators bin the data into bins of specified duration and then apply the specified operator on the bin. The appliance has the ability to precompute data reductions using these operators as part of the ETL process. If applicable, these precomputed data reductions are used during data retrieval to speed up the response times. These same operators can also be used to decimate data as ETL moves it from one store to another. Thus, one can store a few days worth of data at full rate and then decimate the data as it is moved to a slower store.

ADMINISTRATION

The EPICS Archiver Appliance offers a web UI (Fig. 6) for typical configuration tasks.

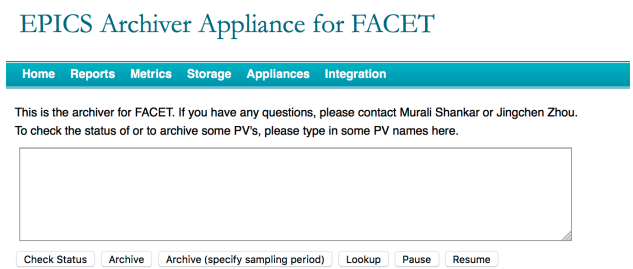


Figure 6: Web based management UI.

The web UI communicates with the appliances using JSON/HTTP and uses business logic exposed as web service calls. All of these web service calls are also available for use from external scripting tools like Python. There is a rich catalog of business logic that lets the user add/modify/delete PVs from the archiver, pause/resume PVs, reshard/consolidate data etc. None of these business logic operations requires an appliance restart. In addition, a wide variety of reports based on static and dynamic information are available. These include reports based on storage rate, disconnected PVs and many others. The reports are also accessible from Python scripts; thus one

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

can script the entire monitoring and administration of a cluster of appliances using Python scripts.

INTEGRATION WITH CHANNELARCHIVER

To facilitate a smooth transition from the ChannelArchiver, the EPICS Archiver Appliance includes the ability to proxy the ChannelArchiver XMLRPC data server. This allows facilities to skip a migration step where massive amounts of data have to be converted to a new format. The appliance also includes the ability to import ChannelArchiver XML configuration files. However, for facilities that do wish to convert their data, MSU has developed utilities to convert ChannelArchiver data into appliance PB files. The carchivetools suite also includes two backend servers, a2aproxy and archmiddle that can be used as a switchable proxy between a ChannelArchiver and an EPICS Archiver Appliance.

INSTALLATION

The EPICS Archiver Appliance requires recent versions of Linux, Java and Tomcat (and MySQL if configuration is stored there). While some installation scripts are provided; the easiest way to get going quickly is to use the provided Puppet modules. In addition, a quick start script is provided to facilitate easy evaluations.

DEPLOYMENTS

Table 1: Facilities with Production Deployments

Name	Lab	PVs	GB/day	Years	Cluster
LCLS	SLAC	200K	19	1.5	3
NSLS2	BNL	61K	42	0.5	1
NSCL	MSU	83K	1	2	2*
FACET	SLAC	34K	1	2	1
TestFac	SLAC	37K	1	2.5	1

* NSCL has two appliances to provide for redundancy

At SLAC, we have significantly reduced the maintenance costs of archiving; many tedious tasks have been eliminated. Many new use cases are currently being explored, including the use of archivers for fault analysis. The ease with which one can set up a small personal archiver that can then be integrated with a wide variety of tools is very useful. Many IOC developers regularly use a personal archiver to aid in development. Some operators use a separate archiver to gather data at high rates for a small set of PVs and then run complex processing on these datasets.

CUSTOMIZATION

It is unlikely that the features of the EPICS Archiver Appliance match a facility's requirements exactly over a period of time. The EPICS Archiver Appliance has been built with customization in mind. In addition to generating custom builds for a site specific look and feel, the product can be customized with site specific policies, configuration stores etc. The storage plugin interface allows a facility to store the data using an alternate storage scheme. If the PB serialization scheme is acceptable, but the notion of storing PB chunks as files is not; then Java NIO.2 can be used to store the chunks in any key value store. If not, a custom type system can be used to support an alternate serialization scheme.

SUMMARY

The EPICS Archiver Appliance has been in production use at SLAC/MSU/BNL and other facilities for two years. It has eliminated tedious maintenance tasks, reduced maintenance costs and allowed us to add more PV's to the archivers. Various facilities are in the process of evaluating/switching to the EPICS Archiver Appliance. A collaborative ecosystem has developed around the product with many labs contributing to the effort.

REFERENCES

[1] ChannelArchiver http://icsweb.sns.ornl.gov/kasemir/archiver/manual.pdf
[2] K. U. Kasemir and L.R.Dalesio, "Overview of the EPICS Channel Archiver", ICALEPCS, 2001.
[3] Protocol Buffers: https://developers.google.com/protocol-buffers/?hl=en
[4] EPICS V4 Normative Types http://epicspvdata.sourceforge.net/alpha/normativeTypes/normativeTypes.html
[5] Control System Studio http://controlsystemstudio.org/
[6] carchivetools https://github.com/epicsdeb/carchivetools

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE EVOLUTION OF THE SIMULATION ENVIRONMENT IN ALMA

T. Shen, R. Soto, N. Saez, G. Velez, S. Fuica, T. Staig, N. Ovando, Joint ALMA Observatory, Santiago, Chile
J. Ibsen, European Southern Observatory, Santiago, Chile

Abstract

The Atacama Large Millimeter /sub millimeter Array (ALMA) has entered into operations transition phase since 2014. This transition changed the priorities within the observatory. Most of the available time will be dedicated to science observations instead of technical time for commissioning activities including software testing. The lack of the technical time surfaces one of the weakest points in the existent infrastructure available for software testing: the simulation environment of the ALMA software. The existent simulation focuses on the functionality aspects but not on the real operation scenarios with all the antennas. Therefore, scalability and performance problems introduced by new features or hidden in the current accepted software cannot be verified until the actual problem explodes during operations. Therefore, it was planned to design and implement a new simulation environment, which must be comparable, or at least, be representative of the production one. In this paper we will review experiences gained and lessons learnt during the design and implementation of the new simulated environment.

OVERVIEW

The Atacama Large Millimeter /sub millimeter Array (ALMA) started the operations phase transition in 2014. This transition changed the priorities within the observatory. Most of the available time will be dedicated to science observations instead of technical time for commissioning activities including software testing. (since software/hardware integration was the former priority during construction phase). The lack of the technical time surfaces one of the weakest points in the existent infrastructure available for software testing: the simulation environment of the ALMA software. In ALMA, simulation capabilities were initially developed to satisfy Control and Correlator subsystem needs, supplying them with virtual hardware devices to interface with the software components being developed. Additional simulation layers and capabilities were added during the years by different teams, but they were focused on the functionality aspect but not on the real operation scenarios with the whole array. Therefore, scalability and performance problems introduced by new features or hidden in the current accepted software cannot be verified until the actual problem explodes during operation. The lack of a representative testing environment will seriously impact the efficiency of the ALMA incremental software release process.

It was planned to design and implement a new simulation environment, which must be comparable, or at least, be representative of the production one. Duplicating the production environment was not an option, since it would be prohibitive from the point of view of the associated cost. Adjustments in the existent simulation architecture had to be introduced, but with special care on keeping the simulation environment comparable in terms of CPU load, network bandwidth throughput, memory usage and software configurations. The selected platform to provide computing power is based on blade technology of Cisco Unified Computing System (UCS). The new simulation platform will provide the required amount of time for testing purposes, at same time it allows us to maximize the efficiency of the reduced technical time available with operational hardware, which will be dedicated only for the final validation of a new release and small set of features that interact directly with the system. In this paper we will review experiences gained and lessons learnt during the design and implementation of a new simulation environment.

PRODUCTION ENVIRONMENT

The hardware related to the ALMA array (antennas, photonic references, correlators, etc) are controlled through an STE [1], which is a collection of servers, real time computers, network switches, storages and databases, that are configured accordingly as a single platform to support the execution of the ALMA observing software [2]. In the Fig. 1, the list of servers, and real time computers are shown.

EXISTENT SIMULATION ENVIRONMENT

The existing simulation platform is a reduced scale version of the STE dedicated for operation, which contains few servers are available to simulate an array of few antennas. This simulation environment has been very useful during the AIV stage [3] where the focus was the commissioning of newly assembled antennas. At that moment, no more than two antennas were usually configured into a single STE.

It became evident that the existent simulation environment needed to be upgraded since the pressure to use the production hardware for scientific observation is extremely high. Currently, the most critical deficiency in the current simulation environment is in the deployment aspects. It differs in terms of network layout, network throughput, computing power, available memory and disk I/O.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE NEW SIMULATION ENVIRONMENT

The following requirements were defined for the new simulation environment:

- To be able to simulate up to 66 antennas.
- To be able to simulate 4 basebands of BL and ACA correlators.
- To use the same network design than production
- To be representative of the production environment in terms of available processing powers, memory, network bandwidth.
- To keep the same O.S., server hardware architecture.

As it is impossible to replicate the amount of servers involved in the operational environment due to the cost, the strategy presented here is to group components within a more powerful server in order to reduce the required number of servers, but at the same time to maintain the deployment characteristic/complexity of the production environment. In this context, the criteria to group components must make sure that critical resources such as network bandwidth, memory allocation, CPU load in the simulated environment, etc. must be representative and comparable with the production environment.

The virtualization approach was tested but with poor results, as the I/O performance required cannot be sustained by virtual machines. In the other hand, adding another layer of abstraction in the testing scenario doesn't add any value when one of the main requirements is to emulate the performance in the operational environment. Instead, a consolidation approach in more powerful servers was explored.

The methodology to find the right consolidation factor is to take a baseline performance measured in the existent hardware during end-to-end tests [4]. Then reconfigured in a more powerful servers with more components, the limit is found when the load stops to grow linearly. Investigation done in [5] allows us to estimate the required computing power in order to simulate an environment up to 66 antennas and the equivalent for the processing of the 4 basebands in the BL correlator and ACA correlator Within this investigation, the foundation to provide computing power is based on blade servers, and the following consolidation can be achieved in each server, as shown in the Fig. 1: i) 8 antenna real time computers (ABM) can be consolidated into a single server, ii) GNS, GAS type of servers will be map 1 by 1 between the production and simulation environment. So far, there are 8 servers of this kind, iii) 4 CDP nodes can be consolidated into a single server and iv) 4 ACA CDP nodes can be consolidated into 1 blade server. In total, the number of the required blade servers summed up to 32.

CISCO UCS as Source of Computing Power

Considering the tests performed, the most critical hardware resources are i) number of processors and cores, and ii) the available memory of the server. Based in these technical needs, a search was developed to find a suitable platform to provide such amount of processing capacity. Finally, CISCO UCS [6] was identified as the best solution. It comprises CISCO B22 servers with 48GB Ram, 2 six cores CPU, and CISCO UCS5108 Chassis, which allow us to have a scalable solution with 32 blades in modular chassis. The UCS5108 Server Chassis can accommodate a maximum of 8 blade servers and provides a total of 1.2 Terabits per second of available Ethernet throughput.

Networking

From the network point of view, the Cisco 6200 series Fabric Interconnect switches are fully compatible with the existing ALMA network backbone (Catalyst 6500 and Nexus 5000), ensuring that the new system will both be able to provide the maximum intended throughput and keep the same network layout than production environment [3]. The latter is a very important decision factor, because it will allow us to use the same configuration and administration tools developed for the production environment. As shown in [3], we just needed to add an additional instance of VRF (Virtual Routing and Forwarding) in the existent network domain. The new VRF instance contains the same Layer-3 network topology than the production VRF instance

Storage

High performance Storage Class provided by EMC VNX2 covered the need for storage in the simulation environment. Another advantage of Cisco UCS is that the same Fabric Interconnect also serves to provide 16 Gpbs redundant fibre channel connections to the storage.

USE CASES

Since the moment the new simulation environment was put into production, it has been the key platform for the verification phase of the incremental software release process [7]. The usual testing scenarios are: (i) regression tests, (ii) scalability tests, iii) new feature/functionality tests and (iv) bug fixes testing.

Besides the aforementioned scenarios, it is worth to mention that this simulation platform has been also very useful to verify the performance of key components of the ALMA software, such as concurrent access to the TMCDB database, bulk data transfer and system start-up parallelisation. It has also been useful to find long-standing integrity issues such as memory leaks and memory corruption that are seen after long observations, and to test new deployment strategies before going into production.

It has been possible to stress the access to the TMCDB database, which allows testing different solutions to alleviate the bottleneck situation. Applying the improvement has resulted in reducing the time spent during subsystems initialization and observations in the operation environment. Having the same network layout, the Bulk data transfer network loads can be tested in a similar way as in the operational environment, helping to find introduced throughput or timeout problems early in the testing process.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

item	Description	type	# Equipment in production	Consolidation factor	# Of required blade server
1	Antenna Bus Master, Central LO RT computers	abm	68	8	8.5
2	CentralLO RT computers, DMCs	lo-x, cob-dmc	7	8	0.875
3	CDP nodes	cob-cdpn	16	4	4
4	CDPMaster, CCC, COJ-CDPMaster, COJ-CC	cob-cdpm, cob-cc, coj-cdpm, coj-cc	4	2	2
5	ACA CDP nodes	coj-cdpn	32	4	8
6	General Network Servers	gns	2	1	2
7	General Application Servers	gas	6	1	6
				Total # of blades	31.375

Figure 1: The different types of servers and real time computers configured the production environment and how they could be consolidated into more powerful servers.

It is now possible to test the system start-up parallelization scalability in simulation, to prevent the introduction of situations that may increase the initialization times in an unreasonable amount. Using the similarities to the production environment it has been possible to recreate real observing scenarios while analysing the processes for memory leaks and memory corruption, which has allowed increasing the continuous uptime of the system, by maintaining the resources usage low and reducing unexpected crashes.

This platform also allows us to prepare the upgrade of the operative system to RHEL 6.6 and to port ALMA software into 64 bit architecture.

FUTURE WORKS

Future works are located in the enrichment of simulation behaviour of hardware devices, such as antennas pointing, correlators modes, etc. We expect to incorporate concepts such as model in the loop or hardware in the loop, which gives us the advantage to use exactly the same software than production (currently the hardware drivers are not being tested, since they are being replaced by simulators [8]), therefore to achieve a better coverage with our testing process.

It is also considered to use the idle time of this simulation environment to run automatic testing. Our goal is to support continuous software integration, as part of the ALMA software delivery process [7], and this environment will be perfect to execute nightly builds.

CONCLUSION

A new simulation environment was designed and implemented. It fulfilled entirely the defined requirement, specially being a representative testing environment of the production environment. After its introduction, the amount of technical time requested on the production environment for software testing has been reduced considerably.

The testing environment has the same network configuration. Servers and devices (simulated) are deployed exactly in the same way as in production. Therefore the same configuration and tools are being used in both places.

Blade servers have demonstrated to be an excellent alternative to provide computing power, which scales, taking account that the observatory's data center is located in the Atacama Desert and providing power and space is not something trivial. Cisco UCS, as a complete solution, has the advantage that non-additional network devices are required to procure in order to put the blade servers chassis into production. It was important that this solution is fully compatible with our existent network design. Internally, Cisco UCS's Fabric Interconnect switches provide high availability and redundancy by design; therefore our former in house mechanism implemented by using network bonding at the O.S. level is not necessary anymore.

Finally, with the experience learnt in this area, it is planned to upgrade to production environment using the same technology in the next year.

ACKNOWLEDGMENT

The authors acknowledge the contribution of all the members of the ALMA Department of Computing to the success of this project.

REFERENCES

[1] T.C. Shen, R. Soto et al., "ALMA Operations Support Software and Infrastructure", Vol. 9149 91492D-6, proceedings of SPIE, SPIE' 12, Amsterdam, Netherlands (2012).

[2] A. Farris, R. Marson, J. Kern et al. "The ALMA Telescope Control System", Proceedings of ICALEPCS, ICALEPCS' 05, Geneva, Switzerland (2005).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

[3] T.C. Shen, N. Ovando, et al, “Virtualization in network and servers infrastructure to support dynamic system reconfiguration in ALMA”, Vol. 8451, 84511N, proceedings of SPIE, SPIE’ 12, Amsterdam, Netherlands (2012).
[4] A. M. Chavan, B.E. Glendenning, J. Ibsen et al, “The last mile of the ALMA software development”, Vol. 8451 84510Q-1, proceedings of SPIE, SPIE’ 12, Amsterdam, Netherlands (2012).
[5] M. Mora, J. Avarias, A. Tejeda, et al, “ALMA software scalability experience with growing number of antennas”, Vol. 8451, 84510W, proceedings of SPIE, SPIE’ 12, Amsterdam, Netherlands (2012).
[6] Cisco Unified Computing System website: http://www.cisco.com
[7] R. Soto, T.C. Shen, N. Saez, and others, "ALMA release management: a practical approach”, proceedings of ICALEPCS, ICALEPCS’ 15, Sidney, Australia (2015).
[8] M. Mora, J. Ibsen, J. Kern, and others, “Hardware device simulation framework in ALMA control subsystem”, Vol. 411, p.462, proceedings of ADASS, ADASS XVIII, San Francisco, USA (2008).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

EPICS PV MANAGEMENT AND METHOD FOR RIBF CONTROL SYSTEM

A. Uchiyama#, M. Komiyama, N. Fukunishi
RIKEN Nishina Center, Wako, Saitama, Japan

Abstract

For the RIKEN Radioactive Isotope Beam Factory (RIBF) project, the Experimental Physics and Industrial Control System (EPICS)-based distributed control system is utilized in Linux and vxWorks. Utilizing network attached storage (which has a high-availability system) as a shared storage, common EPICS programs (Base, Db, and so on) are shared by each EPICS Input/Output Controller (IOC). From the initial development of RIBF control system, it has continued to grow and consisted of approximately 50 EPICS IOCs and more than 100,000 EPICS records. Because RIBF has been constructed by extending RIKEN Accelerator Research Facility (RARF) in a previous project, the controllers for RARF are also utilized for RIBF control system. In this case, the dependence between the EPICS records and EPICS IOCs becomes complicated. For example, it is not easy to know the accurate EPICS record name information using only the device information. Therefore, we constructed a new management system for the RIBF control system to easily call up the detailed information. In the system, by parsing startup script files (st.cmd) to run EPICS IOCs, all EPICS records and EPICS fields are stored in the PostgreSQL-based database. By utilizing these stored data, we succeeded in developing Web-based management and search tools.

INTRODUCTION

The RIKEN Radioactive Isotope Beam Factory (RIBF) accelerator facility consists of five cyclotrons, including a superconducting ring cyclotron, and two linear accelerators as injectors [1]. For the RIBF, we constructed a control system based on the Experimental Physics and Industrial Control System (EPICS) for magnet power supplies, beam diagnostic instruments, vacuum control systems, and so on [2]. Different types of devices are connected via the EPICS Input/Output Controllers (IOCs) as front-end controllers for accelerator operation. For example, VME-based IOCs are adapted for magnet power supplies, and beam diagnostic and vacuum systems (e.g., Faraday cup and beam profile monitor) are constructed using Linux-based IOCs connected with N-DIMs, which are Ethernet-based control devices originally developed by the RIKEN Nishina Center [3]. Additionally, CAMAC, GPIB, Programmable Logic Controller (PLC), and others are also utilized as control devices with EPICS for the various applications in RIBF control system. The types of control devices and EPICS IOCs are listed in Table 1.

Currently, the EPICS-based RIBF control system consists of 51 IOCs and approximately 110,000 EPICS records. We have constructed the EPICS runtime database using an Oracle-based database. In this case, using the Oracle database has some advantages from the viewpoint of management because it is available to automatically make EPICS runtime database files easily using a program. In case of a large number of EPICS runtime database files for the same purpose, the abovementioned approach is efficient. Since RIBF control system has been constructed by extending the control system of RIKEN Accelerator Research Facility (RARF), which is the previous project, the relationship among their controllers are complicated. Therefore, we often manually develop EPICS runtime database files without using Oracle-based database.

For system maintenance and development in this situation, we need to identify the IOC connected to the process variables (PVs) because the RIBF control system does not use the UDP broadcasts for *ca_search* from EPICS clients on the subnet. However, the relationship between the IOCs and the PVs is confused, and an efficient system environment was not provided for developers and operators. Therefore, we have constructed a system to manage the EPICS PV to solve the complication.

Table 1: Current Status of the Type of EPICS IOC in the RIBF Control System

IOC Platform	Connected Control Device	Type	Number of IOCs
Linux x86	N-DIM [3] PLC GPIB Other network-based devices	Soft IOC	24
Linux x86	CAMAC	Embedded IOC	6
Linux f3RP61 [4]	PLC	Embedded IOC	14
vxWorks	NIO [5]	Embedded IOC	7

PV MANAGEMENT SYSTEM

IOC and Shared Storage

In the RIBF control system, common EPICS programs (EPICS-base, application programs, runtime database, and additional extensions programs) are stored in the network attached storage (NAS), and they are shared by all EPICS IOCs using the network file system or file transfer protocol. The system diagram is shown in Fig. 1. The NAS, manufactured by NetApp, provides a centralized system with a high-availability. On the other

#a-uchi@riken.jp

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

hand, the IOCs are mounted common to the EPICS programs as read only, and the common EPICS programs are always compiled on another server, which is mounted in a write-allow configuration for control system development users. In addition to the development server, the management server is also mounted with the shared storage to read common EPICS programs.

Method of Storing PV Information

By reading the startup script files and accessing the EPICS runtime database files, the program can parse the runtime database files. Therefore, we developed a program such that the information is separately stored in the PostgreSQL-based database by parsing the file. The program is also compatible with the EPICS substitution file and macro. The system chart is shown in Fig. 2. Additionally, the information of the network-based devices, which are managed by the IOC, is inserted in the database. The summary of the types of data stored in the database is listed in Table 2. This program for parsing is coded in PHP and is regularly invoked from Crontab services, which run on Linux; thus, the data are always in the latest state.

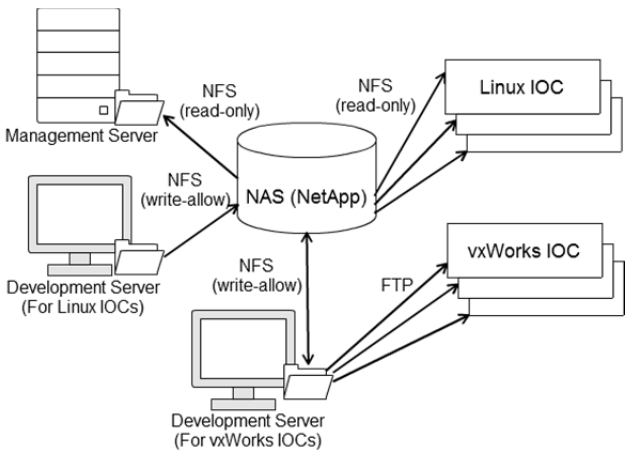


Figure 1: System Diagram for the IOCs and Shared Storage.

Table 2: Database Structure Used for Management System in the RIBF Control System (October 2015)

Table Name	Information Stored in the Column	Number of Records
iocinfo	Hostname of IOC	51
	Directory path of EPICS application	
	Startup script file name	
pvinfo	EPICS record name	110,192
	Record type	
fieldinfo	Field	3,151,383
	Field type	
device2ioc	Hostname for network-based device	432

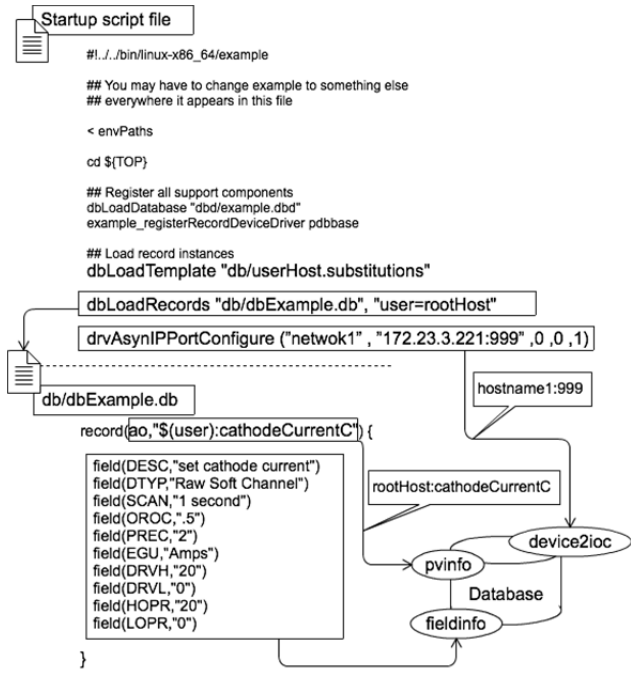


Figure 2: System Chart of the Developed Program and Inserted Data.

STORED INFORMATION USAGE

Command-line Tool

Because the EPICS PVs and fields information are stored in the PostgreSQL-based database, the information is available to use for various purposes by using network communication. Therefore, we developed command-line-based tools to obtain the information, and it was implemented in the operational log system in the RIBF control system. The operational log system is one of the electric log systems for recording and viewing the accelerator operating time and contents of an operated device. To obtain operational information, we use *caMonitor*, which is an event-driven program using EPICS channel access protocol. However, the operational log system requires setting a large number of monitored parameters. Additionally, the specification of the runtime database will be changed for software maintenance. Therefore, manually setting the names of the required EPICS PVs without omission in the previous system is difficult. By using this system, the names of the required PVs are never missed from the list because creating the lists with a program is easy.

Web Application

By developing Web applications, searching the IOC hostname from the EPICS record name becomes possible, which is the same function as *cainfo*, a command-line tool included in the EPICS base software. On the other hand, because this developed system has an autocomplete feature, searching by EPICS record name is possible without requiring the completed EPICS record name. A screenshot of the developed Web application is shown in Fig. 3. Additionally, for all the fields that make up the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

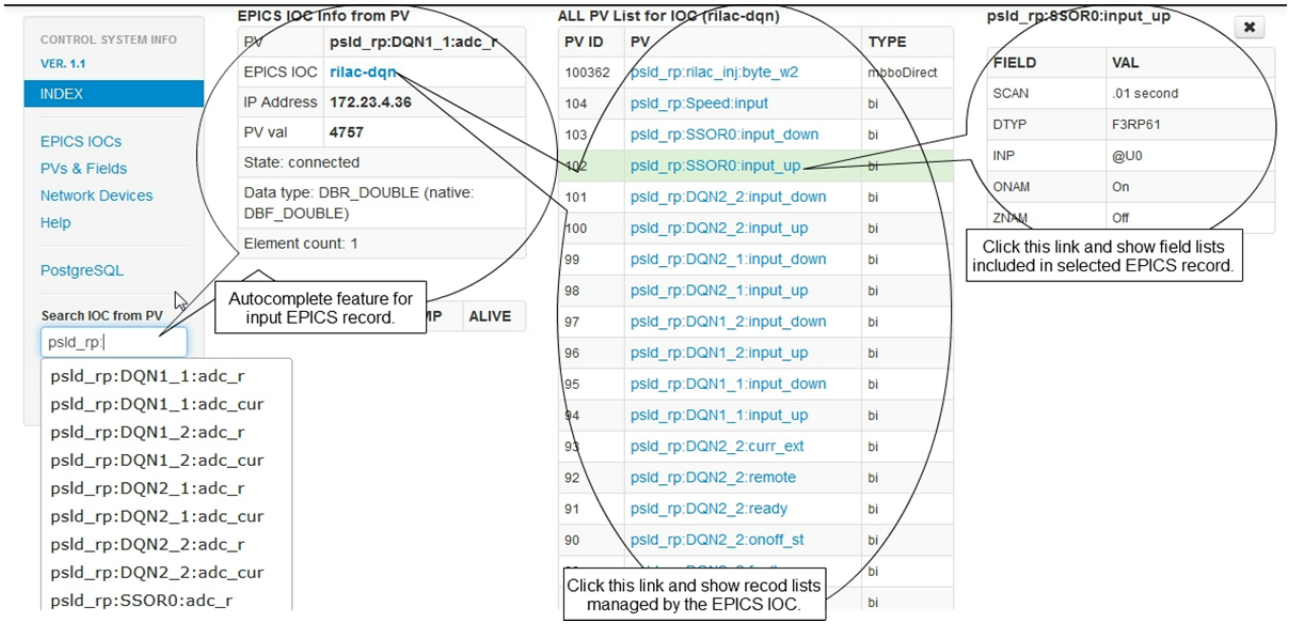


Figure 3: Screenshot of the Developed Web Application to Search EPICS IOC Name from EPICS Record.

EPICS record, accessing them from the record name via Web browser becomes possible. Therefore, we can check the information of the EPICS record and the fields without a source code to manage the system. Because the information of all network devices is also stored in the PostgreSQL-based database from the INP field and startup script file, the Web application has a feature that search the information of network-based devices from EPICS IOC name. Therefore, we can easily observe the relationship of the IOC and the managed network devices.

FUTURE PLAN

Using inserted information, alive monitoring will be implemented in the near future for the RIBF control system. Generally, alive monitoring is performed by a system that checks from the outside whether the computers continuously work. Because the alive monitoring conveys the downtime of the system to the administrator, it is an important system from the point of view of system availability. For EPICS-based system, an infrastructure monitoring system using NAGIOS [6] was introduced at Argonne National Laboratory (ANL) [7]. The NAGIOS-based system obtains information such as IOC from IRMIS [8], which is another management system for EPICS-based system at ANL. The XML files are created as parameters for alive monitoring.

On the other hand, EPICS IOCs are monitored by the *caget* function as a polling program in the NAGIOS-based system. In our system, we plan to implement a monitoring function using *caMonitor*. Additionally, not only IOCs but also all network-based devices will be monitored using the IP address and port number in this system. Therefore, we will be able to check both the port open/close and the ping states.

CONCLUSION

We have developed a system for management of EPICS PVs for the RIBF control system. Because the EPICS runtime database of files and startup scripts are managed in one place, their program analyses and the information of EPICS record, field, and network devices are stored in the database. Using this information, we can check the relationship among the IOCs and EPICS records, fields, and network devices via Web applications as well as the command-line tool. Currently, this feature is used in coordination with the operational log system.

In the future, we plan to implement an alive monitoring system for IOCs and network devices using the information in this management system.

REFERENCES

[1] O. Kamigaito et al., Proc. IPAC'14, Dresden, Germany (2014), p. 800.
[2] M. Komiyama et al., Proc. ICALEPCS'14, San Francisco, CA, USA (2014), p. 348.
[3] M. Fujimaki et al., RIKEN Accel. Prog. Rep. 37 (2004), p. 279.
[4] A. Uchiyama et al., Proc. PCaPAC08, Ljubljana, Slovenia (2008), p. 145.
[5] T. Tanabe et al., Proc. ICALEPCS2003, Gyeongju, Korea (2003), p. 597.
[6] https://www.nagios.org/
[7] D. E. R. Quock et al., Proc. PCaPAC08, Ljubljana, Slovenia (2008), p. 19.
[8] D. A. Dohan et al., Proc. ICALEPCS2007, Knoxville, Tennessee, USA (2007), p. 82.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE POWER SUPPLY CONTROL SYSTEM OF CSR

W. Zhang, S.An, S.Z.Gou, K.W.Gu, Y.P.Wang ,P. Li, M.Yue
IMP, LAN Zhou 730000, P.R. China

Abstract

This article gives a brief description of the power supply control system for Cooler Storage Ring (CSR). It introduces in detail mainly of the control system architecture, hardware and software. We use standard distributed control system (DCS) architecture. The software is the standard three-layer structure. OPI layer realizes data generation and monitoring. The intermediate layer is a data processing and transmission. Device control layer performs data output of the power supply. We use ARM + DSP controller designed by ourselves for controlling the power supply output. At the same time, we have adopted the FPGA controller designed for timing for power supply control in order to meet the requirements of accelerator synchronizes the output of the power supply.

INTRODUCTION

HIRFL-CSR, a new ion Cooler-Storage-Ring (CSR) project, is the post-acceleration system of the Heavy Ion Research Facility in Lanzhou (HIRFL). HIRFL-CSR is a multi-purpose CSR system that consists of a main ring (CSRm), an experimental ring (CSRe), and a radioactive beam line (RIBLL II) to connect the two rings. Figure 1 show an Overall Layout of HIRFL-CSR [1]. The two existing cyclotrons SFC (K = 69) and SSC (K = 450) of the HIRFL will be used as its injector system. The heavy ion beams with the energy range of 8–30 MeV/u from the HIRFL will be accumulated, cooled and accelerated to the high-energy range of 100–400 MeV/u in the main ring, and then extracted fast to produce RIB or highly charged heavy ions. The secondary beams (RIB or highly charged heavy ions) will be accepted and stored by the experimental ring for many internal-target experiments or high-precision spectroscopy with beam cooling. On the other hand, the beams with the energy range of 100–900 MeV/u will also be extracted from CSRm by using slow extraction or fast extraction for many external-target experiments.

Two electron coolers located in the long-straight sections of CSRm and CSRe, respectively, will be used for the beam accumulation and cooling. One internal target in the long-straight section of CSRe will be used for nuclear physics and highly charged state atomic physics, and many external targets of CSRm will be used for nuclear physics, cancer therapy study and other researches.

CSR is a double ring system. In every operation cycle, the stable-nucleus beams from the injectors are accumulated, cooled and accelerated in the main ring (CSRm), then extracted fast to produce RIB or highly charged ions. The experimental ring (CSRe) can obtain the secondary beams once for every operation cycle. The accumulation duration of CSRm is about 10 s. considering the ramping rate of magnetic field in the dipole magnets to be 0.1–0.4 T/s, the acceleration time of CSRm will be nearly 3 s. Thus, the operation cycle is about 17 s [2].

In CSRe, two operation modes will be adopted. One is the storage mode used for internaltarget experiments or high-precision spectroscopy with electron cooling. Another one is the deceleration-storage mode used for atomic-physics experiments.

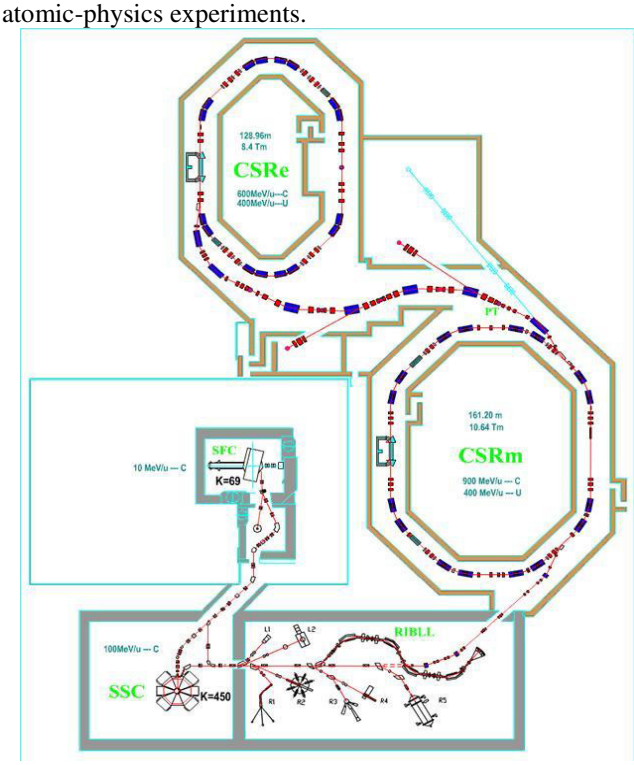


Figure 1: Overall Layout of HIRFL-CSR.

SYSTEM INTRODUCTION

The control system is based on the Ethernet star topology. It takes Ethernet as the transmission medium to connect each part, mainly including: database system, the synchronization server, the front end server, the I/O part (ARM controller and the DSP processor) and the communication module. Database system and synchronization server is the information and control centre of the whole system. There is an independent Ethernet connection between the front end of the server and the I/O component, and a front-end server manages multiple I/O components. The I/O part directly controls the device object, and an I/O part controls one or four device objects. The structure of the power control system is shown in Figure 2. In the system, power control is accomplished by three kinds of data files: process data, event table data and command table data.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

In the power supply control system, the ARM controller is directly used as a network node to realize the information exchange between the upper control centres; at the same time, the DSP control board is used as the control unit, and the real-time and accurate monitoring of the control equipment is carried out.

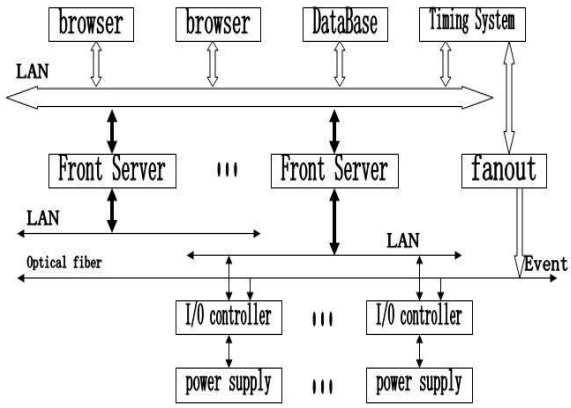


Figure 2: The structure of the power control system.

HARDWARE

There are two types of hardware for the control system. The specific contents are as follows:

The ARM Controller

The ARM controller with S3C4510B as the core, connects DSP through the back, and is provided by a power supply for the backing plate. The structure of the ARM controller is shown in Figure 3.ARM controller with SDRAM, Flash, data CPLD and system Flash. It receives data files (process data, event data, and command table data) through the network; Data transmission to the DSP through the HPI port of DSP; System Flash for curing u-boot, kernel and root file system; Data Flash is used to store the important control parameters and waveform data, so as to avoid the loss of data.

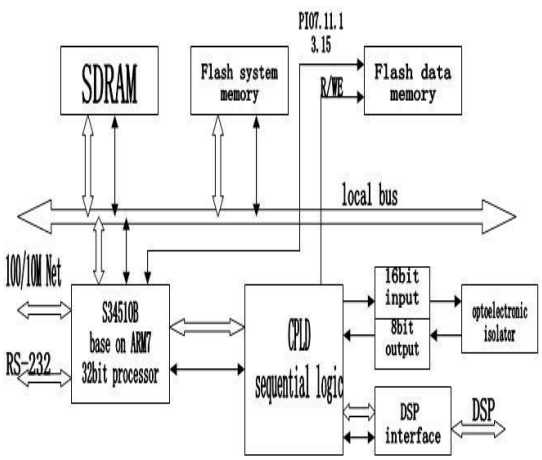


Figure 3: The structure of the ARM controller.

The DSP Processor

The DSP processor is connected to the ARM controller by the backing board, and the power supply is provided by the backing plate. The structure of the DSP processor is shown in Figure 4.It takes TMS320VC5402 as the core, and the SRAM and Flash of 512K*16bit. Through the back board, the DSP processor receives the process data and event data from the HPI port of the ARM controller, and stores it in the SRAM. The trigger event is received by the FPGA case of a light switch. Flash used to cure DSP programs and procedures. The DSP processor has a DAC and an ADC channel [3].

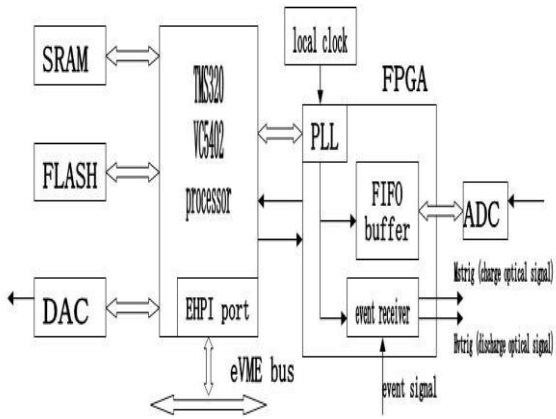


Figure 4: The structure of the DSP processor.

SOFTWARE DESIGN

The power supply control system uses the C/S structure. The structure of the software is shown in Figure 5.The first layer is a synchronization server. It is through the data exchange with the database system to get the accelerator's effective operation data and distribution, and also provides the time system of the trigger event and customer operation procedures. Database service is second layer. It provides the synchronization data file required by the CSR to run an independent cycle. A network connection is used between the synchronous server and the I/O component and the database system.

The software mainly includes: the Oracle database, the embedded database on the server, the application program and the service program of the related peripheral driver, the waveform algorithm program of the front generator. In addition to the Web based debugging interface program and synchronous trigger program, the rest of the software is a server program or automatic implementation of the application, without user intervention.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

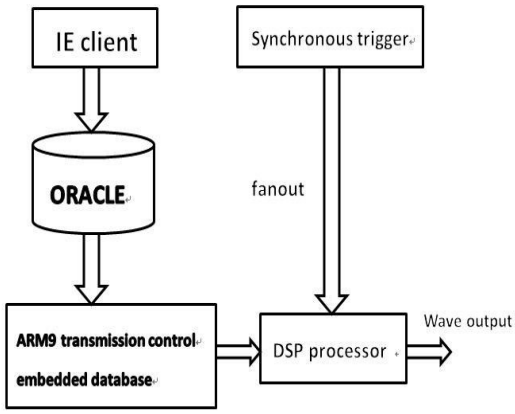


Figure 5: The structure of the software.

We provide a web interface program based on Web technology, which can be easily loaded into the Oracle database in the front end of the binary waveform file. The waveform data is passed to the database file of the front controller by update command and the IP address of the initial set of the front-end controller, at the same time to inform the embedded database SQLite has a new data to arrive. The embedded database stores the new data according to the distribution of the DSP storage space and the current waveform output buffer. When the required data is passed to the DSP storage space, we can start the pulse trigger program to generate the optical pulse signal. After the trigger event is consistent, the DSP processor outputs the waveform. The waveform data download flow chart is shown in Figure 6.

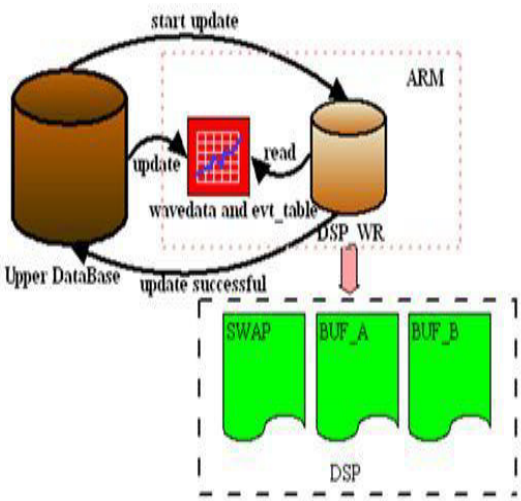


Figure 6: The waveform data download flow chart.

SUMMARY

The control system has been applied to the actual operation in 2008 and passed the acceptance of the CSR expert group. It implements the requirements of the power supply control system for the CSR project. The control system has been stable operation of the seven-and-a-half years, and ensures the normal conduct of the experiment.

REFERENCES

[1] Xia Jia-wen ZHAN Wen-long YUAN You-jin LIU Yong, et al. Heavy Ion Synchrotron and Cooler-Storage-Ring in Lanzhou. Atomic Energy Science and Technology，2009.43 (z1)

[2] Xia Jia-Wen, ZHAN Wen-Long, WEI Bao-Wen, et al. General Design of the CSR Project in IMP. 2006, Vol.30 (4)： 335-343

[3] JI Peng, QIAO Wei-min, JING Lan.Power Supply Control System of CSRm Based on ARM and DSP. Nuclear Electronics & Detection Technology. Jul. 2008. Vol.28 No.4:728-730(In Chinese)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DATA CATEGORIZATION AND STORAGE STRATEGIES AT RHIC*

S.Binello†, K.Brown, T.D'Ottavio, J.Laster, R.Katz, J.Morris, J.Piacentino
Collider-Accelerator Department, BNL,Upton NY, USA

Abstract

This past year the Controls group within the Collider Accelerator Department at Brookhaven National Laboratory replaced the Network Attached Storage (NAS) system that is used to store software and data critical to the operation of the accelerators. The NAS also serves as the initial repository for all logged data. This purchase was used as an opportunity to categorize the data we store, and review and evaluate our storage strategies. This was done in the context of an existing policy that places no explicit limits on the amount of data that users can log, no limits on the amount of time that the data is retained at its original resolution, and that requires all logged data be available in real-time. This paper will describe how the data was categorized, and the various storage strategies used for each category.

INTRODUCTION

The NAS that had served as the main repository of data and software critical to running the accelerators, as well as the initial repository of logged data since 2008 was declared End Of Life by the vendor in 2014. As such, we needed to decide whether to continue using the system and obtain support from a third party, or simply replace the system. As this NAS had other limitations: insufficient storage, outdated small expensive 300GB drives and 1GB network interfaces, along with a hefty $15K annual maintenance fee for a mere 25TB of storage, the decision was made to replace the NAS.

DATA CATEGORIZATION

To facilitate and organize the analysis of requirements for the new system we categorized our data into three tiers: critical operational data, auxiliary operational data and historical/temporary data. Each tier was then assigned a level of required reliability.

Tier 1 – Critical Operational Data

This is data that is needed to run the accelerators. It consists of program executables, basic configuration information, Front End Computer (FEC) boot areas, archives of recent settings, and critical group home directories.

This data should always be available. A complete High Availability(HA) solution is required. That is, the storage device should have redundant components at all levels. In the event of multiple failures which makes this data unavailable, a backup with data as current as possible should be made available. A switch to the backup system should take less than 4 hours.

*Work performed under Contract Number DE-AC02-98CH0886 with the auspices of the US Department of Energy
† sev@bnl.gov

Tier 2 – Auxiliary Operational Data

This data may be important for some operational tasks, but is not critically needed for basic machine operation, for example, logged data from the current run, archived settings from past runs (and possibly archived settings from earlier in this run).

A total HA solution is desired, but is not absolutely necessary. Minimally, RAID should be used to protect from disk failures. If primary storage for this data is not a total HA solution, a backup version of the same data should be maintained on an alternate disk storage device. Data on the backup should be no more than three days old. A switch to the backup should take less than 4 hours.

Tier 3 – Historical or Temporary Data

This data is mostly composed of logged data from past runs, along with any data that may only need to be stored temporarily.

Some high availability features are desired. Minimally, RAID should be used to protect from disk failures. In the event of some other failure, restoration from tape would be necessary. Restoration of most types of data could be done in less than an hour but could take several days.

DATA STORAGE STRATEGIES

Once the data was classified attention focused no how best to support the requirements for each data tier.

Tier 1 – Storage For Critical Data

From the onset, it was understood that a High Availability (HA) NAS was the required solution for critical operational data. The focus was then on how to increase the availability of this critical data in the event of a failure of the primary NAS, and how to provide the most up-to date data possible. The Disaster Recovery (DR) system for the NAS that was to be replaced, was a low cost Linux storage server with internal SATA drives RAID'ed using 3-ware controllers (this is a configuration that is also use to support tier 3 data). Critical data was replicated from the NAS to the DR system using the Linux rsync utility. This pairing proved to have some flaws. The DR system was not identical to the NAS, and as the NAS was asked to store and perform more and more, over time it eventually outpaced the capabilities of the DR system. Additionally, we found that the rsyncs were also negatively impacting the performance of the primary NAS.

It was decided that in order to prevent this type of divergence in capabilities in the future, and in an attempt to limit the impact of data replication between the primary and DR unit, an identical NAS was needed for disaster

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

recovery. Currently, tier 1 data is replicated every 24 hours, but we expect to increase the frequency of replication. Optimized tools, provided by the NAS vendor, are used to efficiently replicate data between the primary NAS and the DR NAS.

Tier 2 – Support for Auxiliary Data

Several options were considered to store tier 2 data. As this data is not critical to running the accelerators, it does not require a complete HA NAS solution. As such, we needed to determine if we should continue storing tier 2 data alongside tier 1 on the NAS, or look for an alternate storage solution. Even though this data is not critical, a fault with the storage device could prevent the capture or availability of data that might still be of significant interest to users.

Two main reasons were identified to support the idea of a separate storage solution. One, was that access to this data might interfere with the read/write performance of tier 1 data. Another, was to provide a less expensive solution for tier 2 data. As such, two options were considered for tier 2 data. The first was to purchase an alternate less expensive NAS. The second, was to use an inexpensive Linux storage server configured with internal drives and protected with RAID controllers (i.e. our tier 3 storage server configuration). In this configuration there is no fail-over capability in the event of a RAID controller failure, nor in the event of a motherboard failure. To circumvent this limitation, we considered the possibility of configuring an alternate system to act as a backup. However, in the event of a failure the transition to this system would require down-time and data loss that we would not have to incur if we were to use a NAS .

Eventually, we resolved to store tier 2 and tier 1 on the same NAS. Financially it made sense to use the same NAS, as it only required the purchase of additional disks. And, even though it was not needed, it endowed tier 2 data with all the same HA capabilities as those provided for tier 1 data.

To address the concern that tier 2 data might negatively impact the reading/writing of tier1 data, we still wanted to separate these two tiers as much as possible. The initial thought was to direct tier 2 data to the DR NAS, while storing tier 1 data onto the primary NAS. In this configuration, each NAS acted as a primary for its designated data type while also acting as the DR system for its partner NAS. This was the preferred solution, however due to financial constraints the DR NAS did not have a redundant head (not a desirable configuration for a NAS acting as primary storage). As a result, tier 1 and tier 2 data were both stored on the primary NAS. To reduce contention between tier 1 and tier 2 data, the primary NAS was configured so that each tier was supported by its own NAS head, network connections, and dedicated disk drives.

Not only were the two tiers stored on separate disk drives, but different types of drives were used for each tier. As the amount of tier 2 data collected over the entire run is quite large (approximately 60TB compared to 6TB for tier 1), and we wanted to have the option of keeping an entire run's data on the NAS, it was decided to store tier 2 data on larger, cheaper, and slightly less reliable NL SATA drives. Tier 1 data was stored on 900GB SAS drives. This not only brought down the original cost of the NAS but also significantly reduced maintenance fees.

While the option exists to store a whole run's worth of tier 2 on the NAS, in fact the NAS is currently used as a temporary staging area for tier 2 data. Tier 2 data is initially captured and then stored for only a few weeks on the NAS, after which it is relocated to low-cost storage servers. This approach provides HA for the initial capture of tier 2, and during that period of time where it is most likely to be of interest to users.

Tier 3 – Support for Historical/Temporary Data

Tier 3 data is composed of older logged data from past runs (in effect tier 3 data is simply older tier 2 data), or large data files deemed not critical by users. As such it comprises the largest amount of stored data.

To provide real-time access to logged data from previous runs, we have had a strategy in place since 2001 where logged data is eventually moved from the NAS to inexpensive Linux storage servers. Data from previous runs is stored under dedicated directories for each run, and accessed through links on central directories that reside on the NAS.

When these storage servers fill up we simply purchase additional systems. We have found that over the years this approach allowed us to take advantage of the ever-increasing disk sizes and decreasing costs of SATA drives. Last year we purchased a 196TB system for $20K. When this strategy was initiated, the typical server stored about 5TB. Presently, storage for tier 3 data is provided by three 196TB, two 96TB and two 48TB systems.

Logged data on older servers, with less storage, is eventually consolidated onto newer servers. This may prove more cumbersome in the future as the amount of data to migrate increases.

Another option considered for tier 3 data was to store it on the the disaster recovery NAS. This NAS has the ability to support 4PB of data, and the ability to efficiently replicate data from the primary. However, there was concern about maintenance fees and cost of disk drives when compared to the generic Linux storage servers.

CONCLUSION

This latest NAS purchase provided us with an opportunity to categorize the data we store and to re-examine our data storage strategies. We considered various options, but in the end the solution was not all too different from the previous one. Tier 1 and 2 data is collected on the same HA NAS device, though tier 2 data is only stored there for a short time. Tier 3 data is stored on inexpensive generic Linux storage servers with limited HA capabilities. An almost identical NAS was purchased for disaster recovery, and is only used to backup tier 1 data and provides storage for tier 2 only in event the primary NAS fails. To be determined is to see how well this approach scales as we continue to store ever increasing amounts of data.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DATA LIFECYCLE IN LARGE EXPERIMENTAL PHYSICS FACILITIES: THE APPROACH OF THE SYNCHROTRON ELETTRA AND THE FREE ELECTRON LASER FERMI

Fulvio Billè[#], Roberto Borghes, Francesco Brun, Valentina Chenda, Alessio Curri, Venicio Duic, Daniele Favretto, Georgios Kourousias, Marco Lonza, Milan Prica, Roberto Pugliese, Martin Scarcia, Michele Turcinovich, Elettra Sincrotrone Trieste, Basovizza, Trieste, Italy

Abstract

Often the producers of Big Data face the emerging problem of Data Deluge. The literature includes experiences and proposals for dealing with this problem. Nevertheless experimental facilities, such as synchrotrons and free electron lasers, may have additional and specialised requirements. This is partially due to the plethora of cutting-edge custom instrumentation but also to the necessity for a complex system that manages the access to the facility for thousands of scientists. A complete data lifecycle describes the seamless path that joins distinct IT tasks such as experiment proposal management, user accounts, beamline software, data acquisition, reduction and analysis, archiving, cataloguing, and remote access. This short paper presents the data lifecycle of the synchrotron Elettra and the free electron laser FERMI. It attempts to go beyond the overgeneralised binary distinction of information between data and metadata since in practice there are many heterogeneous but related data sources. With the focus on the broad concept of data access, the Virtual Unified Office is presented. It is a core element involved in scientific proposal management, user information database, scientific data oversight, and remote access. Eventually recent choices of advanced beamline software are here discussed as well as approaches to distributed control systems. The latter holds the key role to data and metadata acquisition but it also requires integration with the rest of the system components in order to facilitate additional services such as cataloguing, archiving and remote access. With the experience of the past years and the forecasting of future requirements, the expected system upgrades are also here outlined. The scope of this paper is to disseminate the current status of a complete data lifecycle, discuss key issues and hints for future directions. This is potentially useful for similar facilities especially for those under construction or upgrade.

INTRODUCTION

Elettra and FERMI

The research centre Elettra consists of two advanced light sources, the electron storage ring Elettra and the free-electron laser (FEL) FERMI hosting 32 experimental stations.

Elettra is a third-generation synchrotron serving the scientific and industrial community since 1993. It is involved in a broad spectrum of research in physics, chemistry, biology, life sciences, environmental science, medicine, forensic science, and cultural heritage. Currently 28 beamlines, including a storage-ring free-electron laser, utilize the radiation generated by the Elettra source. All of the most important x-ray based techniques in the areas of spectroscopy, spectromicroscopy, diffraction, scattering and lithography are present, together with facilities for infrared microscopy and spectroscopy, ultraviolet inelastic scattering, and band mapping. In 2010 the third-generation electron storage ring source Elettra was upgraded to operate in top-up mode. It is the only third-generation synchrotron radiation source in the world that operates routinely at two different electron energies; i) 2.0 GeV for enhanced extended ultraviolet performance and spectroscopic applications, and ii) 2.4 GeV for enhanced x-ray emission and diffraction applications. A plan for a substantial upgrade named Elettra 2.0 is in motion [1].

FERMI is a new seeded free electron laser (FEL) facility in operation next to the third-generation synchrotron radiation facility. Unique among the FEL sources currently operating in the ultraviolet and soft x-ray range worldwide, FERMI has been developed to provide fully coherent ultrashort 10-100 femtosecond pulses with a peak brightness ten billion times higher than that made available by third-generation light sources. It is a single-pass FEL that covers the wavelength range from 100 nm to 4 nm in the first harmonic. It comprises two separate coherent radiation sources, FEL-1 and FEL-2, that are being brought online sequentially. FEL-1 operates in the wavelength range between 100 and 20 nm via a single cascade harmonic generation, while the latest FEL-2 is designed to operate at shorter wavelengths (20-4 nm) via a double cascade configuration.

Data Deluge and Discontinuity

The era of Big Data has raised relevant problems like that of Data Deluge [2]. Elettra and similar facilities can be seen as information generators. The processes that generate such information flows are rather specialised since the facilities themselves have particular workflows. Such workflows include proposal submission, evaluation, beamline access, execution of experiments with custom instrumentation, data analysis, dissemination of scientific results etc. At this stage the issue of Data Deluge is studied in the focused context of a synchrotron facility.

[#]fulvio.bille@elettra.eu

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

This brief manuscript suggests that the elements with the biggest impact in Elettra are: advances in detectors, upgrades of the sources, demanding data analysis, and archiving requirements. These four elements contribute to a very demanding information flow which is a challenge to manage. The next sections provide the reader with an overview on how it is done at Elettra.

PROPOSAL MANAGEMENT

An experiment in a synchrotron facility requires a very specific procedure. While the core remains the actual experiment, the complete process includes multiple subprocesses like proposal submission and evaluation, access request, data access and processing, remote operations and general accounting. In 1997 Elettra has been among the first institutes of its kind, to develop a complete system named Virtual Unified Office which serves as the backbone for the above-mentioned processes. This document refers to proposal management as the collective of all operation related to it. Even if this section is focused on proposal management for reasons of connecting it with the broader data flow, the VUO extends much further acting as a sort of ERP system.

Virtual Unified Office

The web portal of the VUO manages the whole lifecycle of an experiment: from the first approach of the researchers to the publication of the results. It is based on Oracle and mod_owa while each webpage corresponds to a PL/SQL stored procedure.

The first phase is the registration: it is of paramount importance since the username/password combination is used by the user not only on the web portal but also on the acquisition workstation, on the data storage devices, on the computational cluster for data analysis, on the instruments used for remote operations and Wi-Fi access. It is a AAA system different from the company one and it is dedicated to the scientific users. An alternative option is the Umbrella authentication system [https://www.umbrellaid.org] which is also available.

The proposal workflow is fully managed from the portal and depending on the result of its evaluation it may end up in the beamtime schedule of the portal calendar.

The researcher submitting the proposal can use a 3-level storage: Scratch, Online, and Offline. Scratch and Online are high reliability storages using the Gluster distributed file system. Offline is managed by an external storage server farm at the CINECA supercomputing center and is based on iRODS. Scratch is the working disk, where the acquisition and the data reduction and pre-screening take place. Initially, access is allowed only from the beamline workstations. Eventually when the data are finalised they can be transferred to Online by means of simple operations from the VUO portal. The data store in Online cannot be further edited and serves as the main data deposit.

Even if the main interaction is through a simple web portal, the actual operations take place on different servers and they are performed by agents written in Python that are communicating between each other through TANGO [3]. Such operations include the generation of the unix users, creation of the directory tree, permission setting for POSIX Access Control.

The transfer from Scratch to Online happens through the use of encapsulated rsync commands coordinated by Tango agents. There are cases where this transfer occurs in a scheduled and automatic manner as a task of the acquisition applications; these tasks can also deal with possible data preprocessing (i.e. lossless compression of HDF files).

The whole team of participants of the proposal can access the data through WebDAV and though web browsers. In special cases other systems and protocols can be utilised (e.g. GridFTP) even if are not regularly supported. There are also in place systems for remote access allowing for collaboration between the researchers who stay in their home institutes. These are tunnel systems that allow safe access to the resources through RDP, NX, VNC. Naturally on-site access is faster than off-site not only due to the bandwidth but also due the available protocols.

Once the experiment is finished, the user must provide 3 levels of feedback at different time schedule and with increasing detail. The first is requested immediately after the experiment and it mainly concerns the functionality of the machine. The second one regards the achievements, while the final one is the experimental report. Such procedures may seem purely logistical but in practice may serve a real scientific purpose like associating a specific dataset with a peer-reviewed publication.

The same portal is also used for managing the users' reimbursement of expenses therefore it is integrated with the company's ERP. The VUO portal is also multi-facility since it manages not only Elettra and FERMI but also those related to the European CERIC-ERIC consortium [http://www.ceric-eric.eu/].

Another application of the VUO is the registration of all the publications related to experiments performed at the facility. The publication database is also used for evaluating the future research proposals.

DATA ACQUISITION

The actual acquisition of data is a process of paramount importance. The beamline end-stations of synchrotrons and free electron lasers are complex instruments of interconnected components like motors, pumps, and detectors. The data acquisition software is an advanced system capable of commanding all of these components

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

while at the same time provides the user with a graphical interface [4,5].

Endstation Control Systems

The Scientific Computing team of Elettra has designed a modern, flexible and extensible endstation control based on technologies compatible with the TANGO distributed control system. The user interface is based on QT while the core language is Python. The target data format is custom structured HDF5. In order to avoid multiple developments, the system has a common core that aims at easy adaption to each beamline. Special attention has been paid for allowing easy integration with processing pipelines. Due to the brief nature of this manuscript, the reader is encouraged to refer to the manuscript "*A flexible system for end-user data visualisation, analysis prototyping and experiment logbook*" by Borghes et al. present at the ICALEPCS2015 proceedings.

DATA ANALYSIS

In most experiments, the data produced require various stages of processing prior reaching a stage where they can be interpreted. In their initial form they are often referred to as RAW data but this is an oversimplification. During an experiment there are multiple detectors that acquire data (e.g. images, spectra, beam characteristics, machine parameters etc). As a unique set they get assembled and processed. This processing may include multiple steps and later be separated in different operation pipelines. It soon becomes clear that the workflow is neither standard nor static thus it requires dynamic approaches that often make optimized solutions really difficult to be achieved.

Multiple Approaches

Considering that data analysis through computational processing does always take place even in non ideal situation (i.e. lack of a specialised team/resources), in Elettra presently exist multiple approaches. The minimal is the one where the beamline scientist tries to process their own data with the use of personal computers. On the other extreme there is a specialised team for scientific computing where, in collaboration with the beamline personnel, develop solutions that use high end enterprise hardware according to standardized approaches. In real case scenarios both approaches coexist but also all the in-between options as well. While trying to enforce the more advanced solutions, the scientific computing team is still providing support for a variety of options. The core development language is Python but in most cases it relies on external modules often developed in C or Fortran. For GPGPU computing the current developments are mostly in CUDA but past projects included OpenCL. The target operating system is any modern distribution of Linux but without forcing a specific one. Many systems are command-line but where a GUI is necessary it is developed in QT. The TANGO Python binding is often necessary for collecting metadata useful to the analysis. Nevertheless there are still plenty of data analysis software written in IDL and Matlab for Windows.

Data Formats

Decades ago, before the Big Data era, communities like that of crystallography had identified the need for standard and common formats [6]. It is well established why this is a necessity. Nowadays reproducibility, repeatability, exchange, size, speed, standardization of data analysis are even more relevant thus data formats require special attention. Elettra has participated in relevant projects (PaNData Europe, PaNData-ODI) [7] and has specialized studies leading to implementations for formats for X-ray florescence microscopy (XRF), Ptychography and Computed Tomography. All of the beamlines of the FEL FERMI do use HDF5 as the container of metadata-rich structures [8]. This experience provided the scientific computing team with a good insight regarding the advantages and disadvantages of these formats.

Hardware

The hardware resources for data analysis that is non uniform is a complex issue. They require non only HPC and server class hardware but also high end work stations often equipped with professional monitors (i.e. imaging applications). The advance use of GPGPU requires systems like NVIDIA TESLA. In Elettra their main use is in applications for x-ray microscopy imaging. Such applications often require GPU memory boards with >10GB of memory. For the in-house CDI/ptychography capacity these resources still pose the main limitation regarding the size of scans. Finally, the analysis requires accessing the data but this access, if it is through a slow connection, may become the bottleneck of the process. For this reason, the network and relevant infrastructure should always be adequate enough and this is often a difficult task especially with the modern detectors.

ACCESS

Accessing the data happens in two ways: supervised, where a user access them on will, and unsupervised when it is part of an automated process like a data analysis workflow. The access may allow for read and write operation and have specific requirements of performance and user rights. The facility requires a specific structure to describe in-house and visiting research teams. This is done by suitable mapping to advanced ACL on filesystem. These may require user accounts for instruments and automated group hierarchy based on beamtime proposals. Even if this in some cases requires adhoc system administration operations, the majority of such operations are taking place through automatic and semi-automatic procedures in the Virtual User Office. Finally there is a special class of access requirements, that of remote access.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Remote Operations

Any form of off-site data access or operation is considered remote. It usually takes place after a beamtime experiment and in its simplest form regards data download. This is already included as a service in the VUO. Still more complex forms of remote access may be necessary. Such cases go beyond the remote desktop solutions like NX and cloud computing resources for running analysis programs. Among the most demanding is that of remote beamline operation. The latest challenge in Elettra is remote operation of a beamline in an unattended mode (Xpress beamline) meaning in absence of local personnel. Past projects have successfully implemented scientific data analysis systems based on the model of software and hardware as a service [8,9].

INTERCONNECTION

Among the trends of the computational community, there is the focus on metadata and their importance. A beamline dataset usually includes a set of instrument parameters as metadata for the core raw data acquired by the principal detectors. It often becomes apparent that the conceptual distinction of metadata and data in such specific cases is rather difficult. In real world, acquisition and analysis scenarios, this becomes even more complex when a complete dataset is not enough for a complete processing due to the fact that additional pieces of information (like experiment information) are stored in other systems like the VUO. At this point at Elettra a dataset is only a node of a network that all together provides the necessary information. This set of interconnected nodes uses the RDBMS of the VUO, the HDF5 dataset's data and metadata, realtime information from the TANGO control system, user accounting from the file system, etc. The full analysis of this approach is not the argument of this brief overview paper but the core idea is that of abandoning the single file as a dataset and represent it as a set of interconnected heterogeneous information providers. Defining the systems as such should allow us for optimisations and future upgrades.

FUTURE UPGRADES

The synchrotron Elettra was built in 1993 so most of its beamlines have well established operation modes that permit only gradual upgrades and maintenance. For the past years the Elettra scientific computing team which is responsible of all the activities mentioned in this manuscript, had the opportunity to implement new technologies in the new Free Electron Laser FERMI and its beamlines. To extend this activities there is a plan for an upgrade of the synchrotron to a new source: Elettra 2.0 that should include a substantial upgrade and modernization of all the existing beamlines and procedures [1]. The aforementioned axis of the i) set of systems on the Virtual Unified Office, ii) the cloud based distributed and modular data acquisition model, and iii) the systematic data analysis will be among the core elements of the upgrade plan. At the same time there will be a detailed investigation aiming at the optimization of concepts such that of the dataset as an interconnection of heterogeneous information sources. Eventually this will required advances in both infrastructure and scientific software.

CONCLUSION

Synchrotron and free electron laser facilities like Elettra and FERMI have special needs regarding their data flow. In order to face problems related to data deluge, Elettra implements a complete data lifecycle. This covers all the experimental stages; from proposal submission to beamline access and data analysis. In order to do so key elements like proposal management, acquisition software, and data analysis need to be seamlessly interconnected. This brings the introduction to the concept of data as a set of connected information nodes from heterogeneous sources in contrast to the traditional file centric view. Future work will extend and build on this idea while the planned upgrades to Elettra 2.0 may be the ground for implementing the potential advances.

ACKNOWLEDGEMENT

The authors, members of the Scientific Computing team of Elettra, thank the IT Group and the beamlines Groups of Elettra Sincrotrone Trieste. Moreover they acknowledge the importance of advanced technologies and open source software like that of the distributed control system TANGO.

REFERENCES

[1] E. Karantzoulis, “Elettra 2.0 – The next machine”, Proceedings of IPAC2015, Richmond, VA, USA

[2] Anderson, Chris. The End of Theory: The Data Deluge Makes the Scientific Method Obsolete. Wired, 2008.

[3] Götz, A., E. Taurel, J. L. Pons, P. Verdier, J. M. Chaize, J. Meyer, F. Poncet, G. Heunen, E. Götz, and A. Buteau. “TANGO a CORBA Based Control System.” ICALEPCS2003, Gyeongju, October, 2003.

[4] Borghes, R., V. Chenda, A. Curri, G. Gaio, G. Kourousias, M. Lonza, G. Passos, R. Passuello, L. Pivetta, and M. Prica. “Control and Data Acquisition Systems for the FERMI@ Elettra Experimental Stations.” Proceedings of ICALEPCS2011, Grenoble, France, MOPMU015, 2012, 462.

[5] Borghes, R., V. Chenda, A. Curri, G. Kourousias, M. Lonza, M. Prica, and R. Pugliese. “A Common Software Framework for FEL Data Acquisition and Experiment Management at FERMI.” Proceedings of ICALEPCS2013, San Francisco, CA, USA, 2013, 6–11.

[6] Hall SR, Allen FH, Brown ID. “The Crystallographic Information File (CIF): a new standard archive file for crystallography”. Acta Crystallographica A47 (6): 655–685 (1991)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

[7] Prica, Milan, George Kourousias, Alistair Mills, and Brian Matthews. “Requirements for Data Catalogues within Facilities.” Work 501 (n.d.): 63688.

[8] Brun, Francesco, Serena Pacilè, Agostino Accardo, George Kourousias, Diego Dreossi, Lucia Mancini, Giuliana Tromba, Roberto Pugliese, “Enhanced and Flexible Software Tools for X-ray Computed Tomography at the Italian Synchrotron Radiation Facility Elettra”, Fundamenta Informaticae 141 (2015) 233–243

[9] Brun, Francesco, Lucia Mancini, Parnian Kasae, Stefano Favretto, Diego Dreossi, and Giuliana Tromba. “Pore3D: A Software Library for Quantitative Analysis of Porous Media.” Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 615, no. 3 (2010): 326–32.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A FLEXIBLE SYSTEM FOR END-USER DATA VISUALISATION, ANALYSIS PROTOTYPING AND EXPERIMENT LOGBOOK

Roberto Borghes*, Valentina Chenda, Georgios Kourousias, Marco Lonza, Milan Prica, Martin Scarcia, Elettra-Sincrotrone Trieste S.C.p.A, Basovizza, Italy

Abstract

Experimental facilities like synchrotrons and free electron lasers, often aim at well defined data workflows tightly integrated with their control systems. Still such facilities are also service providers to visiting scientists. The hosted researchers often have requirements different than those present in the established processes. The most evident needs are those for i) flexible experimental data visualisation, ii) rapid prototyping of analysis methods, and iii) electronic logbook services. This paper reports on the development of a software system, collectively referred to as DonkiTools, that aims at satisfying the aforementioned needs for the synchrotron Elettra and the free electron laser FERMI. The design strategy is outlined and includes topics regarding: dynamic data visualisation, Python scripting of analysis methods, integration with the TANGO distributed control system, electronic logbook with automated metadata reporting, usability, customization, and extensibility. Finally a use case presents a full deployment of the system, integrated with the FermiDAQ data collection system, in the free electron laser beamline EIS-TIMEX.

INTRODUCTION

At the present moment there are 32 experimental stations operating at the synchrotron Elettra and the free electron laser (FEL) FERMI [1]. The data acquisition and control software of each end-station is almost unique, being highly focused on the specific technique applied. Nevertheless, for the user-oriented applications it is not unusual to meet the requirements that are common to a number of end-stations. For those, the most efficient solution would be a product with a high level of customization reusable in a multitude of situations. As a general idea, such an approach may be compared to a drill tool: it has been designed to perform a specific task but it needs to mount accessories for the alternative uses like drilling in metal, wood or concrete.

The Scientific Computing team had the chance to apply this strategy in different situations, building a collection of software tools called the DonkiTools. Any of the DonkiTools aims at becoming a standard in its field of application at Elettra and FERMI. The name of the suite recalls that of a donkey, a robust and versatile beast of burden, so highly appreciated in the Mediterraen region.

* roberto.borghes@elettra.eu

DONKITOOLS OVERVIEW

Each application of the DonkiTools suite (Fig. 1) originated to solve a specific requirement, but has been designed from the start using a plug-in based architecture that makes it flexible and reusable. The plugin-based extensibility allows for constant evolution of the DonkiTools. It is possible to add a new functionality without the hazard of destabilizing others or the core behaviour.

As a programming language, Python was the natural choice as it is supported by TANGO and offers a high degree of dynamicity and a large set of scientific and visualization libraries.

PyQt4 [2] has been adopted as the GUI development framework. PyQt ensures that the DonkiTools will run on all platforms supported by the Qt, including Windows, MacOS/X and Linux. The plug-ins that can extend each application are basically PyQt Widgets included in the main panel at runtime.

In the next sections, we will present a short overview of three successful applications used by the staff and visiting researchers at Elettra and FERMI:

- DonkiLOG: an extensible electronic logbook for experiments with automated metadata reporting.
- DonkiCCD: a flexible visualizer of scientific images that may be customized and integrated to the end-station control system.
- DonkiPY: a tool for rapid prototyping of data collection and analysis scripts.

Figure 1: DonkiTools icons.

DONKILOG

The first release of the DonkiLOG (Fig. 2) was developed for the needs of scientists working at the free electron laser FERMI. Each experimental shift at FERMI consists of several scan operations. Scans are often

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

performed at the speed of 10Hz making the classical paper notebook completely unfit for the logging task. That created a pressing demand for a user-friendly logbook application capable of taking automated snapshots of the instrumentation status before and during each experimental scan. Initially we investigated the existing solutions but finally the decision was made to develop a brand new application that should be portable, customizable and re-usable.

Main features of the DonkiLOG include:

- WYSIWYG text editor
- HTML based, compatible with any web browser
- export to PDF
- screen-shot insertion
- integrated Data server for uploading images and text from remote machines
- automatic file saving
- plug-in based extensibility

Currently the logbook has been installed and integrated with control and acquisition systems on six different end-stations running both Linux and Windows. The plug-in based extensibility allows the customization of each installation with the addition of extra buttons to the user interface in order to grab data and images from beamline-specific external devices.
Another appreciated feature is the embedded data server that allows to push data from the remote clients: the data acquisition system can log directly into the user logbook or another scientists can upload screen-shots and annotations from another workstation through a remote text editor.

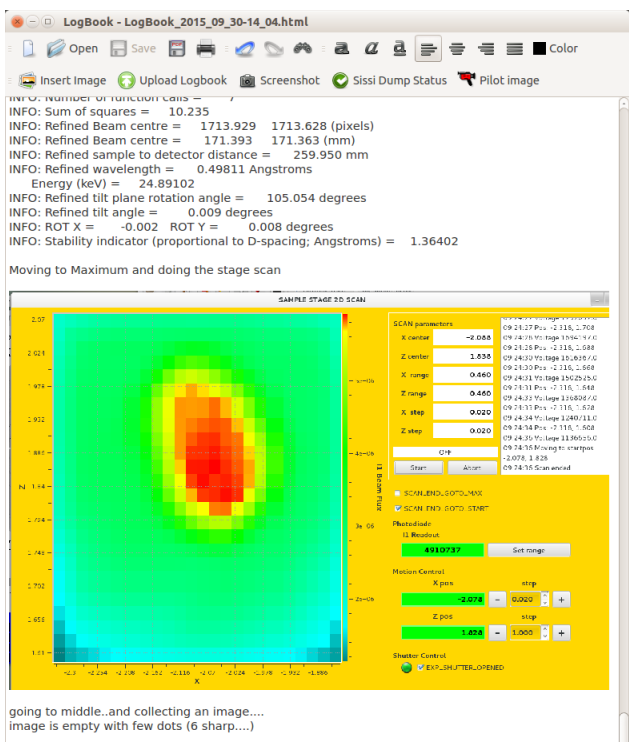


Figure 2: DonkiLOG electronic logbook.

DONKICCD

There is a wide variety of image visualization tools installed at Elettra and FERMI endstations but during the data acquisition phase, the basic needs of all the labs are more or less the same. Based on this observation, we started the development of an application capable of visualizing most of the scientific image formats present at Elettra and FERMI creating a new standard visualizer, powerful and flexible enough to fit the needs of any new imaging end-station.

DonkiCCD (Fig. 3) has been developed using the guiqwt framework [3], a 2D data-plotting library based on PyQwt and on the scientific modules NumPy and SciPy. In order to support different file formats produced by 2D X-ray detectors, the FabIO [4] python library has been used.

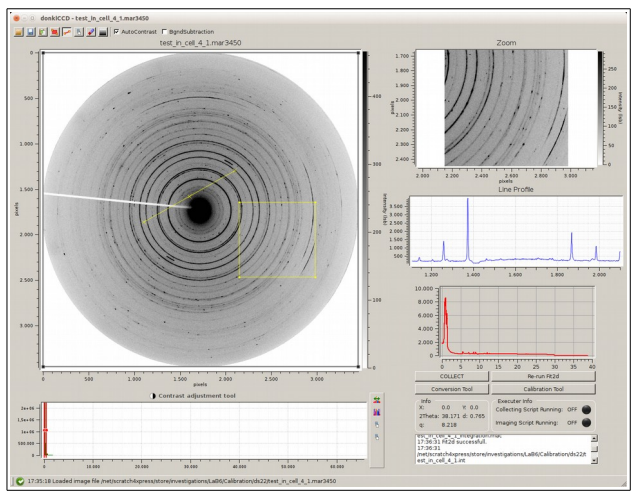


Figure 3: DonkiCCD scientific image viewer.

Main features of the DonkiCCD application include:

- support for common scientific image formats: CBF, TIFF, MAR345,...
- region of interest (ROI) zooming
- line profile tool
- automatic and manual contrast adjustment
- background subtraction
- plug-in based extensibility

Currently, the DonkiCCD is used on seven end-stations for the visualization and near real-time data analysis of images from different scientific instruments: Dectris Pilatus, mar345 imaging plate, Princeton Instruments CCD, Basler and Andor cameras. Through the plug-in architecture each installation can interact with the data acquisition system in order to automatically update the visualized images. Furthermore, upon the requests from the scientists, new data processing tools have been added:

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

averaging display, projections plotting, ROI intensity trend, etc...

DONKIPY

Experimental data acquisition and analysis systems are usually based on well defined data workflows, but sometimes it is necessary to investigate a new experimental technique or a data analysis pipeline. In such cases, a rapid prototyping tool is needed that permits scientists to develop new methods in a fast and simple manner.

Once again, Python turned out as the optimal choice for the scripting language. Its immediacy and readability allows any scientist or student even with basic programming skills to learn the essentials quickly and write simple scripts with a limited effort. This led to the design of a tool for rapid prototyping based on Python and a set of customizable libraries with a user friendly help system.

DonkiPY (Fig. 4) is an extension module of Spyder [5], the well known Python integrated development environment that features advanced editing, interactive testing, debugging and introspection tools. The DonkiPY extension allows the usage of custom libraries within the Spyder environment. Furthermore, the IDE is enriched with an interactive help system that visualizes documentation and examples relative to the supplementary libraries. These are standard Python scripts written with a specific header used to fill the help system.

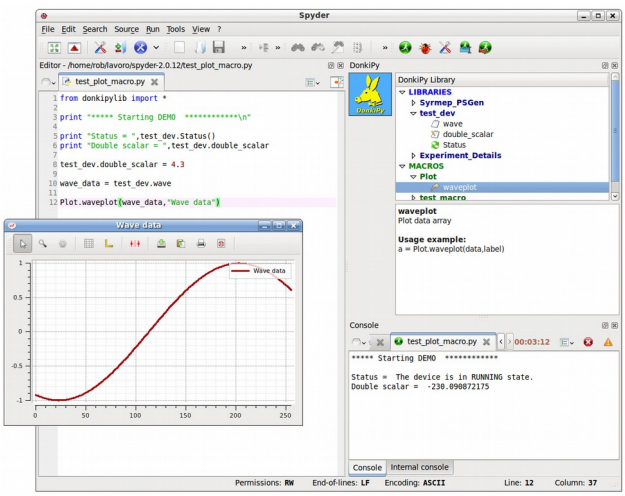


Figure 4: DonkiPY rapid prototyping of analysis methods.

This tool has an extremely wide application field. It has been developed in order to let scientists apply their ideas without a detailed knowledge of the control system or the mathematical methods involved. The DonkiPY custom libraries, also called *macros,* are developed and tested by the specialized staff and afterwards deployed and shared on the experimental end-stations.

Currently DonkiPY is being successfully used at Elettra and FERMI for two main purposes: fast prototyping of new data collection sequences or implementation of new data analysis pipelines. In the first case the added macros simplify the interaction with the instrumentation, in the latter case they are more focused on mathematical algorithms and plotting utilities.

EIS-TIMEX USE CASE

EIS-TIMEX end-station [6] has been designed to exploit the high intensity, energy domain and time structure of the FERMI Free Electron Laser to probe fundamental properties of dense matter under extreme thermodynamic conditions. The experimental chamber and the sample environment configurations have been kept fairly flexible in order to accommodate various possible configurations for single-shot experiments, including simple EUV and soft x-ray absorption/reflection and pump-probe experiments where the probe can be either an external laser or a FEL pulse.

The control and data acquisition system, as on all the FERMI end-stations, is based on the TANGO framework [7]. At the end-station three workstations are used: two are dedicated to instrumentation control and data acquisition while the third one is used for data visualization and analysis (Fig. 5).

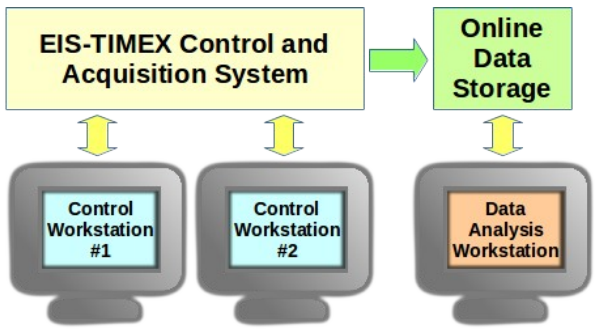


Figure 5: FERMI end-station EIS-TIMEX.

All the DonkiTools described in this paper are fundamental resources for the scientists during the experimental period at EIS-TIMEX. As an experimental shift may last an entire week, 24 hours a day, it is extremely important to ensure the support of easy to use applications, capable of maximizing the automation of the tasks.

Working with Image Detectors

Different experimental techniques have been applied at EIS-TIMEX. Some of them require image detectors like Basler GigE cameras or Princeton Instruments CCD. For those, we have successfully applied DonkiCCD to visualize and perform fast data analysis on acquired images (Fig. 6). The basic functionalities of the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DonkiCCD have been extended according to the needs of scientists:

- multiple regions of interest support
- running average plotting over multiple images
- image intensity axis projections
- data thresholding
- integrated instrumentation control panel

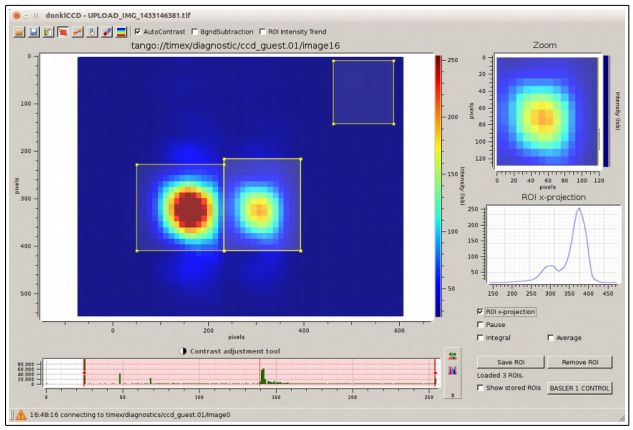


Figure 6: DonkiCCD is used to visualize Basler cameras and Princeton Instruments CCD.

Experiment Log

Experimental activities are logged using the electronic logbook DonkiLOG integrated with the TANGO data acquisition system in order to automatically log additional metadata about FERMI status and the experimental scan in progress (Fig. 7). Furthermore, using a remote client application, it is possible to append data analysis results and screen-shots from other workstations. At the end of the experimental shift the logbook is exported as PDF file and saved with the scientific data on the online storage.

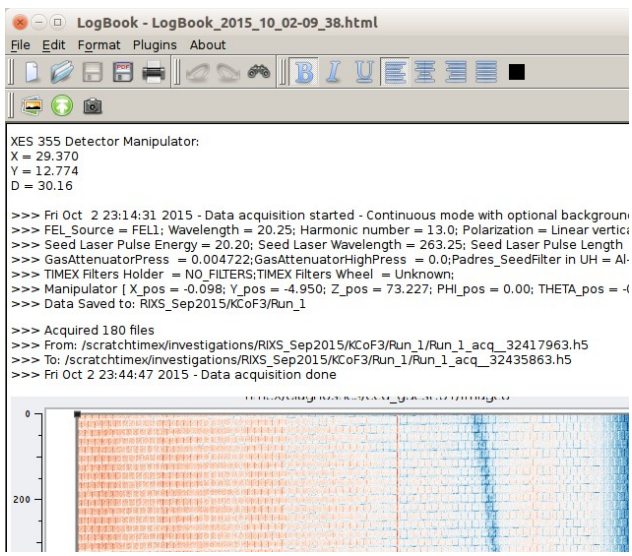


Figure 7: DonkiLOG accepts log messages from data acquisition system.

Data Analysis

An experimental data collection may produce hundreds of HDF5 files that need to be analysed, correlated and plotted. Scientists can access and analyse data on the central storage using a dedicated Linux workstation with the DonkiPY tool installed that permits the rapid prototyping of analysis methods. A set of custom macros developed ad-hoc simplifies the procedures of correlating data coming from multiple HDF5 archives. Users may immediately add results and charts to the user logbook using the DonkiLOG remote client application.

CONCLUSIONS

The DonkiTools approach has been repeatedly put to test at Elettra and FERMI end-stations and was proven to be efficient and successful. DonkiTools applications are flexible, portable over different platforms and user friendly. The strength of this suite resides in its design choices: a modern and powerful language like Python at the core, a highly portable graphics library like Qt for the GUI and a plug-in based architecture that permits a high degree of customization.

ACKNOWLEDGEMENT

The authors, members of the Scientific Computing team of Elettra, thank the IT Group and the beamline Groups of Elettra Sincrotrone Trieste. Moreover they acknowledge the importance of advanced technologies and open source software like that of the distributed control system TANGO.

REFERENCES

[1] Elettra and FERMI Light Sources website: http://www.elettra.eu/lightsources/index.html
[2] PyQt application framework website: https://riverbankcomputing.com/software/pyqt/intro
[3] GuiQwt Python library website: https://pythonhosted.org/guiqwt
[4] FabIO Python I/O library for images website: https://pypi.python.org/pypi/fabio
[5] Spyder the Scientific PYthon Development website: https://pythonhosted.org/spyder/
[6] A. Di Cicco et al., "Probing phase transitions under extreme conditions by ultrafast techniques: Advances at the FERMI free-electron-laser facility", Journal of Non-Crystalline Solids, Vol. 357 – 14, (2011)
[7] R. Borghes et al., "A Common Software Framework for FEL Data Acquisition and Experiment Management at FERMI", ICALEPCS2013, San Francisco, USA, (2013)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MONITORING MIXED-LANGUAGE APPLICATIONS WITH ELASTICSEARCH, LOGSTASH AND KIBANA (ELK)

A. De Dios Fuente, O. O. Andreassen, C. Charrondière, CERN, Geneva, Switzerland

Abstract

Application logging and system diagnostics is nothing new. Ever since we had the first computers scientists and engineers have been storing information about their systems, making it easier to understand what is going on and, in case of failures, what went wrong. Unfortunately there are as many different standards as there are file formats, storage types, locations, operating systems, etc. Recent development in web technology and storage has made it much simpler to gather all the different information in one place and dynamically adapt the display. With the introduction of Logstash with Elasticsearch as a backend, we store, index and query data, making it possible to display and manipulate data in whatever form one wishes. With Kibana as a generic and modern web interface on top, the information can be adapted at will. In this paper we will show how we can process almost any type of structured or unstructured data source. We will also show how data can be visualised and customised on a per user basis and how the system scales when the data volume grows.

INTRODUCTION

At CERN, as in any other large organization, lots of data are generated and stored at a rate that makes it difficult for humans to analyse them. In addition, when considering the many different file formats, storage types, and physical locations, treating the data manually can become an unfeasible and overwhelming task. Manual treatment would go from data analysis to bug or error tracking. Having a tool that helps engineers understand the insights of the data stored, generate reports and gather statistics from them in a faster and easier way, is often the key to prevent future problems and where the challenge lies.

Moreover, with any kind of system, the logfile is the first place to look for clues when the system behaves in an unexpected manner. In our case, we have a variety of applications written in LabVIEW, C++ and Java, running on the most popular operating systems (Linux, Windows and OS X) and applications that make use of our LabVIEW Rapid Application Development Environment (RADE) [1], which has part of the services distributed over multiple servers and which is accessible from more than ~100 users at CERN. In such systems, most of the logs have different or weak structures. This led us to investigate how to unify all the different data formats and sources, and how to store all the information in a common place in order to have a tool that monitors and keeps track of all the online data easily and effectively in real time. However, since the needs of the users are different according to the application in hands, the way of getting statistics and reports should be customizable without web-development knowledge or specialised skills.

According to the motivation explained above, a conscientious study was made, in which several tools were identified, studied and tested.

STUDY OF TOOLS

One of the first things to consider was the data format. There is a large variety of sources; LabVIEW applications running on CompactRIO and PXI targets, Apache Tomcat servers, Java services, C++ applications and extensions, where each of them have different formats and purposes. In computing, syslog is a widely used standard for message logging [2] so this was the starting point to define the main format. In addition, the main requirements for the desired system are listed below:

- Support multiple log formats but mainly syslog.
- Support different communications and network protocols.
- Centralised data messages.
- Have a web-viewer to analyse the logs.
- Be able to get statistics of the stored data.
- Be fully compatible with Linux Environment.
- Be easy to use, install and configure.
- Be scalable if data grows over time.

Analysis of Logging Tools

Considering the advances in data mining and the fact that the analysis of “Big Data” [3] is not a new field, we decided to look for an already existing tool.

After searching and evaluating some promising tools found in the market, Logstash and Fluentd [4] were selected for a deeper study and test. Since these tools are evolving and have been redesigned several times due to their increasing popularity, some characteristics and features might have changed after our initial test. The versions used were Logstash 1.3, Fluentd 1.1, Elasticsearch 0.90 and Kibana 3.

Logstash and Fluentd are aimed to unify and manage data collection for better use and understanding. Both are free, open source and based in plugin models to extend functionality. The two of them can be combined with the popular Elasticsearch as the backend data store and Kibana as a front-end reporting tool to complete all the requirements.

Logstash is based on inputs, filters, codecs and output plugins where the inputs are the sources of the data, the filters are processing actions on the data under certain conditions, the codecs change the data representation and finally the outputs are the destinations where the data is sent. Fluentd has the same behaviour for the inputs and outputs but internally all the data are converted to JSON

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

(JavaScript Object Notation) [5] in order to give structure to an unstructured log message.

As a general comparison (Table 1), the two projects have similar features and capabilities; the most significant difference is that Fluentd insists on simplicity, versatility and robustness whereas Logstash focuses on flexibility and interoperability.

Table 1: Comparison between Logstash and Fluentd

Name	Logstash	Fluentd
Inputs/Outputs	Has ~30 inputs and ~50 outputs.	By default has ~10 inputs and outputs.
Filters	Parse logs into different formats	Not possible
Platforms	Any Java VM compatible platform	Not in Windows
Installation	All embedded in a unique jar	Need to install modules separately
JVM	Needed	Not needed
Data storage	Meant to work with Elasticsearch	Works with Elasticsearch, MongoDB…

Initial Testing

As a proof of concept both were installed and intensively tested to be sure about the amount of data they can receive and how they perform.

The conceptual tests were performed on two SLC 6 x64 based machines. One machine was used for Logstash and the other for Fluentd. The installation was really straightforward; it took less than a day with the basic configuration. Then, for Logstash, the following inputs were included: ZeroMQ [6], log4j [7] and UDP. ZeroMQ was used in the CERN middleware libraries for distributed messaging, log4j was the common Java logging utility and UPD was the internal transport protocol used by syslog. For Fluentd, we included the ZeroMQ and UDP inputs, but for Java logging we used its own implementation of log4j: fluent-logger-java.

Therefore, small test examples were implemented in C++ and Java using these protocols where the messages were sent from a different machine to the Logstash and fluentd instances. All machines were in the same network with a Gigabit Ethernet connection. For these tests, we sent 5000 messages per second to both instances and they worked seamlessly for several weeks in a row.

Finally, since both tools are really promising for our needs, we looked again into the rich collection of input and output plugins. Only Logstash supports RabbitMQ [8] (message broker software) and log4j, which are really useful for our National Instruments components and the Apache Server logs where our RADE Java libraries are running. Adding the fact of having all embedded into one JAR made it easier to install, so we selected Logstash as our collector and management of the log data.

ELK STACK

Although Logstash is a separate project, it has been built to work exceptionally well combined with Elasticsearch and Kibana. Together, known as the ELK stack, they become a powerful tool designed to search, analyse, and visualise data, allowing to get insight in real time [9].

ELK Architecture

The ELK stack is meant to be used in a distributed system where each component has different functionality and the usual architecture has five important components, which can be customised according to your system:

- Remote Shipper: collects logs from different machines and sends them to the central Logstash instance.
- Broker: a temporary buffer between the shippers and the central Logstash server. Typically, Redis is used as a queue-cache server, but also messages brokers can be used, such as RabbitMQ, ActiveMQ, etc.
- Indexer processes: they index and save the logs to the data storage.
- Elasticsearch: the storage and search engine.
- Kibana: the web interface.

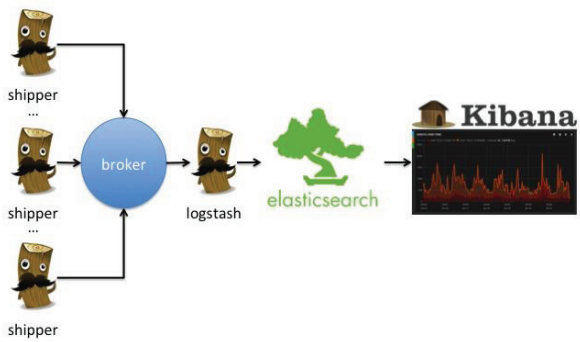


Figure 1: ELK Stack Architecture

Elasticsearch

Elasticsearch is based on Apache Lucene [10] that is the most powerful full-text search library available as open source and written completely in Java.

Moreover, it is a RESTful server so any action can be performed using its REST API using JSON over HTTP [11].

Elasticsearch provides an indexing service and a data storage without the need of defining a database scheme, but it is possible to provide one if needed.

It is meant for handling real time data that needs to be processed and analysed in a rapid manner. Also it is distributed and horizontally scalable which allows starting small and easily add nodes into the cluster if the data volume grows. The cluster consists of one or more nodes, which are established by adding a cluster name to each

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

node. One node in the cluster is elected to be the master node, which is in charge of managing the cluster changes. The data are stored in an index and are composed of a series of fields. Indexes are similar to a database and are mapped shards that hold a slice of all the data in the index. Shards can be either a primary shard or a replica shard. A replica shard is a copy of a primary shard used to provide redundant copies of the data to avoid loss of data in case of failure of any of the nodes. The number of primary shards in an index remains the same since its creation time, but the number of replica shards can be changed at any time [12].

Kibana

Kibana is the web interface embedded into Elasticsearch that helps to understand large volumes of data and rapidly detect patterns or irregularities in them. It provides a flexible and easy way to create dashboards thanks to the widget-based interfaces, which contains a huge variety of charts, graphs or maps. Most of the data is showed as time-series based.

The interface is simple, friendly and intuitive, there is no need to have a web development background to start using it. Since the data and the purposes are different per user, each of them can customize a dashboard according to his needs.

Using Logstash filters, one can create tags according to certain criteria and then it is possible to search and identify messages in Kibana.

In Figure 2, messages were indexed and tagged by Logstash depending on the log message content, then they were filtered with Kibana to visualise the statistics and metrics.

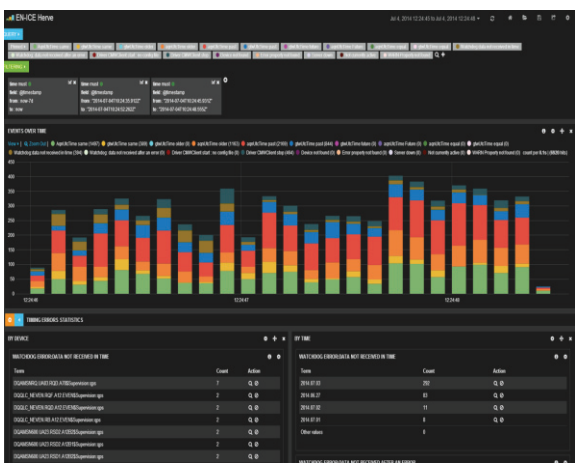


Figure 2: Kibana Dashboard

INTEGRATION, REMARKS AND ISSUES

After the initial tests, we analysed the data structure of the logs deeper and we decided to create in Logstash tags per host name, application name, error code, error log level and type of protocol used in order to identify messages and get statistics easily.

Moreover, we developed a Logger tool in LabVIEW where the messages are sent by UDP to the Logstash instance. The implementation of the utility took less than a day and it is included in the RADE framework where all LabVIEW/RADE users can benefit from it. As a result of this, it was possible to unify multiple log formats from different applications and users and, in some cases, give format to an unstructured data log.

In addition, the horizontal scalability feature that the ELK system provides was really useful. After months of use, we decided to add a new node, in order to avoid loss of data in case of failure. For doing this, it was needed to install a new instance of Elasticsearch in another computer and just give the same cluster name in the configuration file, Elasticsearch take care of the load balancing.

Also, at a certain moment, it was necessary to ship messages from CompactRIO at the Technical Network to the General Purpose Network where the ELK stack is installed. This was solved by adding a remote shipper, as was already explained earlier.

So for the bug diagnostics inside the RADE framework, especially for the Java services, it has been an important and valuable tool. Until now, when a problem appeared, we had to check hundreds of logs into several servers at multiple locations, spending a lot of time trying to find out where the error came from. Now, with the introduction of the ELK stack, a simple query can be run that will point to the problem instantly.

Although, the ELK stack offers many great features, it also has its downsides:

- The lack of security is one of its biggest defects. There is no access control to Kibana or ElasticSearch, so just knowing the URL of the host server, anyone with access to the machine can copy, modify or delete the dashboard or, even worse, the index directly from the database.
- Logstash has a limitation of its buffering capabilities, so if the number of messages keeps increasing, the internal buffer can fill up. In order to solve this, it is necessary to add a broker instance to hold and queue the events.

CONCLUSION AND FUTURE IMPROVEMENTS

Thanks to the introduction of the ELK stack, all the log messages have been unified into a common format and the data storage is centralised. The management and analysis of all these data has greatly improved, users have created their own dashboard according to their needs.

The bug diagnostics has been improved a lot thanks to the ELK stack; all the data logs are centralised in a single application and errors can be identified easily.

The time the developers spent identifying bugs under the RADE framework has been reduced.

One of the improvements we have in mind is to add an access control to avoid interactions of one user's

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

dashboards with other users' dashboards and also to add a security layer on top of all the stored data.

We are also planning to add new graphical components in Kibana and to extend it to be used in other websites outside the ELK stack.

REFERENCES

[1] O. Ø. Andreassen et al. "The LabVIEW RADE framework distributed architecture", ICALEPCS 2011, Grenoble, France, (2011)
[2] Definition Syslog: http://en.wikipedia.org
[3] Big Data: http://en.wikipedia.org/wiki/Big_data
[4] Fluentd website: http://www.fluentd.org/
[5] JSON website: http://json.org/
[6] ZeroMQ website: http://zeromq.org/
[7] log4j website: http://logging.apache.org/log4j/2.x/
[8] RabbitMQ website: https://www.rabbitmq.com/
[9] ELK products definition: https://www.elastic.co/products
[10] Lucene website: http://www.lucenetutorial.com/
[11] Elasticsearch website: https://www.elastic.co/products/elasticsearch
[12] C. Gormley, Z. Tong, "Elasticsearch: The Definitive Guide", (O'Reilly Media, Inc, 2015, 25-29)

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

SCALABLE WEB BROADCASTING FOR HISTORICAL INDUSTRIAL CONTROLS DATA

B. Copy, O. Andreassen, P. Gayet, M. Labrenz, H. Milcent, F. Piccinelli
CERN, Geneva, Switzerland

Abstract

With the widespread adoption of asynchronous web communication, thanks to WebSockets and WebRTC [1], it has now become possible to distribute industrial controls data (such as coming from devices or SCADA software) in a scalable and event-based manner to a large number of web clients [2] in the form of rich, interactive visualizations. There is yet, however, no simple, secure and performant way to query large amounts of aggregated historical data.

This paper presents an implementation of such an architecture, which is able to make massive quantities of pre-indexed historical data stored in ElasticSearch available to a large number of web-based consumers through asynchronous web protocols. It also presents a simple, Opensocial-based [3] dashboard architecture that allows users to configure and organize rich data visualizations (based on Highcharts Javascript libraries) and create navigation flows in a responsive, mobile-friendly user interface.

Such techniques are used at CERN to display interactive reports about the status of the LHC infrastructure (*e.g.*, Vacuum and Cryogenics installations) and give access to fine-grained historical data (such as stored in the LHC Logging database [4]) in a matter of seconds.

THE CERN ENGINEERING DASHBOARD

The CERN Engineering Dashboard is primarily a data streaming facility, giving secure and scalable web-based access to live, critical data coming directly from the CERN Accelerator infrastructure.

The CERN Engineering Dashboard also promotes the usage of modern, standards-based data visualizations, leveraging the best open web standards (*e.g.*, CSS 3, HTML 5, SVG), thereby ensuring that the solution can cope with technological debt and clearly separate the data broadcasting mechanism from the way data is rendered.

Figure 1 below gives an overview of the Broadcasting Dashboard architecture, allowing the scalable distribution of both live and historical data and the rendering of said data through modular and reusable visualizations.

STREAMING HISTORICAL DATA

While our previous publication on the topic of web broadcasting (MOPPC145 [1]) focused on the ability to stream live data, the CERN Engineering Dashboard project has aimed for the past two years to provide the same functionality for historical data, introducing in the process a whole new set of challenges:

- Instantly retrieving and serving large amounts of data, instead of atomic values, while still being able to serve in the vicinity of one thousand web clients concurrently.
- Aggregating data on the fly so that mobile clients are not submerged with unnecessary information that they will, in any case, be unable to display on a small viewport.
- Not making any assumptions on the data at hand, yet still providing efficient indexing and filtering capabilities.

Over the course of this project, it soon became obvious that traditional data storage techniques, such as relational databases, were insufficient:

- Connections to relational databases must usually be pooled (about a dozen concurrent connections are tolerated on the central CERN Oracle database); serving the needs of thousands of clients concurrently was not possible.
- Relational data access times impose dedicated connection drivers and too much latency, and transmitting this data over the web imposes yet more delay and processing cost in data format translation.

A new generation of data storage solutions, identified under the umbrella term “NoSQL,” has emerged in the past few years. NoSQL solutions (thus coined for their breaking free from traditional SQL relational data storage constraints) rely on asynchronous, massively parallel processing and cloud clustering techniques in order to address the aforementioned requirements.

NOSQL DATA STORAGE SOLUTIONS

There are several NoSQL data storage solutions on the market at present. Our goal was to identify a solution that:

- was web-friendly, could natively communicate via HTTP, and support a query language as expressive and feature-capable as SQL;
- could operate in cluster mode, so as to increase availability and support data partitioning;
- could integrate with online processing frameworks such as Apache Storm;

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

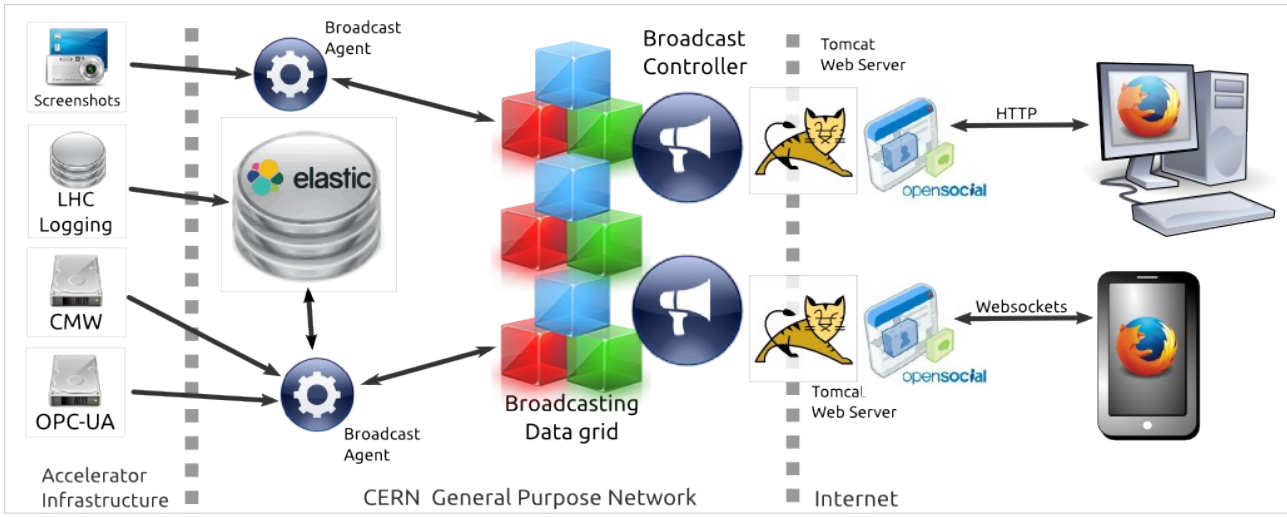


Figure 1: Broadcasting Dashboard architecture

- and would be able to natively manipulate the JSON data format, which is currently used by our web broadcasting framework.

Among the multitude of options available, the open-source solution ElasticSearch [5] fulfils all of the above criteria and benefits from a very strong momentum from large Internet companies and its developer community. Able to operate in isolation or as part of a cluster, it is also extremely easy to embed in any unit tests or integration tests, making it much more comfortable than a monolithic, relational database with regards to development.

Using ElasticSearch, we can store a moderate amount of LHC activity data (typically forty days of history), that we can aggregate on the fly, so as to serve end-users with only the appropriate amount of information suitable for their browsing platform, while retaining the initial expressiveness of the source data.

INTUITIVE NAVIGATION AND REUSABLE VISUALIZATIONS

While ElasticSearch features a data visualization technology called "Kibana," an online web tool that allows one to create dashboards interactively, it suffers from several important shortcomings:

- Visualizations inside Kibana cannot be integrated into another website – they must run inside a Kibana portal interface;
- Kibana is very much focused on ElasticSearch monitoring – it offers a limited number of visualizations that are mostly suitable for line charts, data tables, and tag clouds.
- Extending Kibana with custom visualizations is difficult, and would, in any case, not comply with any web standards, thereby forcing one to develop Kibana-only data visualization modules.

There are, however, existing web standards for contents reuse, such as Opensocial, which defines the concept of **Gadget**, a page fragment that can be parameterized and dropped into any web page (provided certain security constraints are respected).

We therefore developed all our data visualizations as Opensocial gadgets. Even when combining multiple visualizations into a single page layout, each gadget can be easily extracted and reused in another, unrelated website while preserving its interactivity and appeal.

Data visualizations offered by the Engineering Dashboard include Scalable Vector Graphics (to bind data to scalable, vectorized diagrams that offer a pixel-perfect rendering at any display size), Image streaming (to stream bitmap images such as camera captures or screenshots), Highcharts (to display data in a large variety of interactive, zoomable charts), and Impact tree (to render hierarchical, tree-like data and support simple root cause analysis).

DASHBOARD USER INTERFACES

Unfortunately, offering single, isolated data visualizations is not sufficient to provide a comfortable browsing and analysis user experience. Visualizations typically need to be combined, offering different perspectives over the same data (for example, in the LHC cryogenic systems to compare the global temperature versus time of a given sector in a line chart against a temperature profile of individual equipment).

The Engineering Dashboard uses the concept of page, employing a layout to arrange gadgets dynamically on the screen. Since the Dashboard project targets mobile devices, layouts must be responsive [6], that is, automatically adaptable to a range of screen geometries.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The Dashboard's responsive layout implementation relies on the popular Bootstrap [7] framework, which offers a grid abstraction, making it possible to express the most complex layouts in terms of extensible columns and cells, coping gracefully with any display size ratio. Figure 2 below provides an example of visualization, (available at http://dashboard.web.cern.ch/LHC).

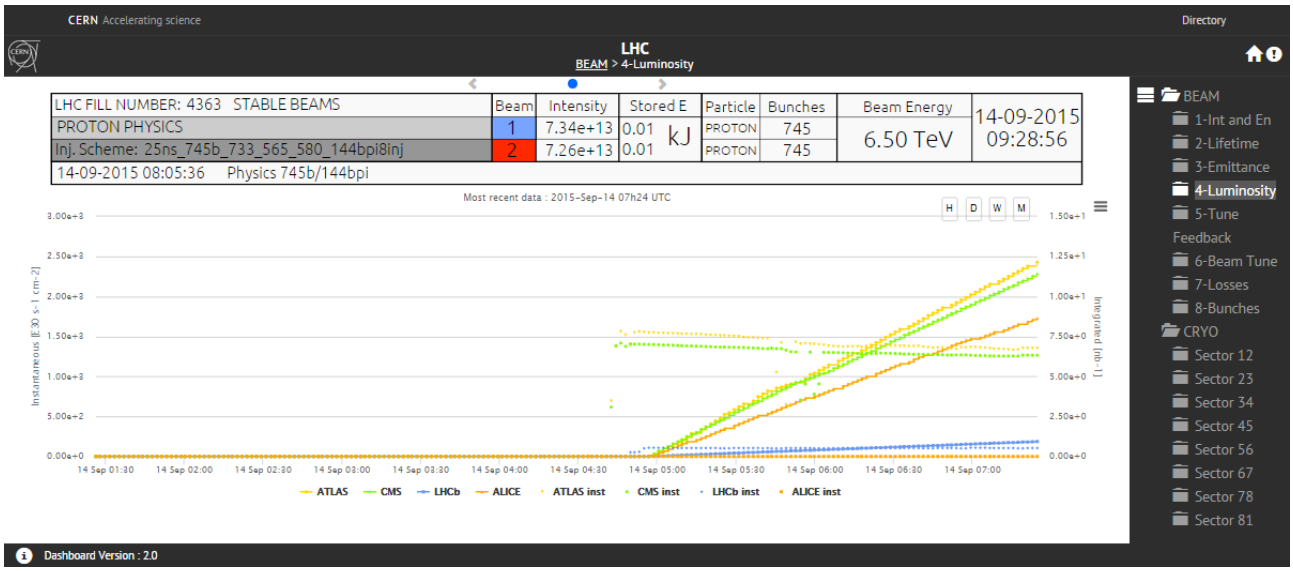


Figure 2: Dashboard view of LHC experiments luminosity history

CLUSTERING AND FAILOVER SUPPORT

Another phase of improvements to the CERN Engineering Dashboard has been the search for a clustering and high-availability solution. To this day, the central CERN web hosting infrastructure does not support high-availability beyond the usage of simple DNS round-robin practices (for example, if three web servers have been defined as part of a cluster then every third request will be sent to the same server; this practice provides basic load-balancing, but no redundancy).

A simple workaround, and a great way to provide high availability and scalability, is simply to bypass altogether the restrictions of traditional web development (relying on Apache web server proxy definitions and HTTP sessions) by adopting a data grid technology [8]. Data grids extend the concept of random-access memory (RAM) by distributing it over a high-speed network. Data grids are transparent to the application programmer, who employs familiar data structures – such as maps, arrays, queues, locks, and even ring buffers – as if they were in local memory. In a data grid, all these data structures are distributed and synchronized across multiple hosts via low-latency data exchanges.

In-memory data grids constitute an ideal way of making our data broadcast service scalable and resilient to failures by maintaining, in a distributed manner, the entire state of data distribution, that is:

- the state of client subscriptions (*i.e.* "who needs what?");
- the availability, health statistics, and heartbeat of broadcasting agents;
- the publication-subscription channels that are currently active;
- last known value cache (no need to re-query the data source if nothing has changed);
- and even supporting the archiving of live data into queues that can later be indexed in a NoSQL data storage.

After a period of evaluation and research, the Engineering Dashboard project adopted Hazelcast [8] as its in-memory data grid implementation. Hazelcast [9] is a high-performance, open-source data grid library that proves extremely simple to use: adding it to any Java application immediately enables a data grid cluster.

Thanks to clustering support from Hazelcast, our broadcasting infrastructure can go from accommodating five hundred concurrent clients from a single server to transparently accommodating five hundred clients over N nodes, without a complex reconfiguration, a code rewrite, or even requiring the restart of any part of the broadcasting cluster.

CONCLUSION AND PERSPECTIVES

Advances made in the past few years in the matters of Java clustered data storage and in-memory data grid technologies have made affordable the deployment of

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

large and scalable data distribution media, such as the one used by the CERN Engineering Dashboard.

The reuse of web content, however, is still a field in mutation:

- Security issues continue to plague the World Wide Web and content reuse is only supported by software vendors insofar as it is done within their own social platform. For example, Microsoft Social and Facebook still do not allow the data visualization mechanisms they provide to be used in the context of other web hosting products. Importing their data into another web site implies relying on their proprietary data formats and protocol specifications, even if open alternatives are readily available.

- Over the past year, Opensocial, as a social data exchange and visualization platform, has been abandoned by Google and converted into a W3C working group (most likely opening an era of lengthy specifications and recommendations that will be ignored by major web companies).

The emergence of a new web standard entitled "Web Components" [10] is, however, poised to revitalize web content reuse and offer a widely accepted method of implementing content sharing. Already supported on all major web browsers, Web Components offers the right balance between the ease of usage and the implementation of strict content separation and code security practices.

The CERN Engineering Dashboard is certainly bound to explore the capacities of this social technology, as a means of overcoming the artificial boundaries imposed by software vendors in the free exchange of data and interactive data visualization gadgets.

REFERENCES

[1] H. Hickson, "Real-time communication between browsers", Feb 2015, W3C working group, http://www.w3.org/TR/webrtc/
[2] B. Copy et al., "Mass-Accessible Controls Data for Web Consumers", MOPPC145, Oct 2013, ICALEPCS'13, San Francisco, USA
[3] T. Çelik, "Opensocial and the Social Wg", July 2014, W3C working group, http://www.w3.org/Social/WG
[4] C. Roderick et al., "The LHC Logging Service", WEP005, Oct 2009, ICALEPCS'09, Kobe, Japan
[5] S. Banon, "ElasticSearch: you know, for search", 12 June 2012, http://thedudeabides.com/articles/you-know-for-search-inc
[6] E. Marcotte, "Responsive Web Design", May 2010, http://alistapart.com/article/responsive-web-design
[7] M. Otto, "Bootstrap from Twitter", 17 January 2012, http://www.markdotto.com/2012/01/17/bootstrap-in-a-list-apart-342
[8] J. Belzer et al, "Very large data base systems to zero-memory and Markov information source", Encyclopedia of computer science and technology, Vol. 14, 1980, Dekker, New York, USA
[9] T. Ozturk, "Clustering your application with Hazelcast", 16 Dec 2013, JAXLondon Conference, London, UK
[10] D. Glazkov and H. Ito, "Introduction to Webcomponents", 24 July 2014, http://www.w3.org/TR/components-intro/

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

METADATASTORE: A PRIMARY DATASTORE FOR NSLS-2 BEAMLINES

A. Arkilic, L. Dalesio, D. Allan, W.K. Lewis, T. A. Caswell, NSLSII, Upton, NY, USA

Abstract

The beamlines at NSLS-II are among the highest instrumented, and controlled of any worldwide. Each beamline can produce unstructured data sets in various formats. This data should be made available for data analysis and processing for beamline scientists and users. Various data flow systems are in place in numerous synchrotrons, however these are very domain specific and cannot handle such unstructured data. We have developed a data flow service, metadatastore that manages experimental data in NSLS-II beamlines. This service enables data analysis and visualization clients to access this service either directly or via databroker API in a consistent and partition tolerant fashion, providing a reliable and easy to use interface to our state-of-the-art beamlines.

INTRODUCTION

Historically, the experimental data in beamline experiments are either hosted in purpose-specific relational database management systems or text files. Relational database systems generally provide robust and high performance solutions for most experiments. However, they can only be adopted by certain experiments since two experiments in the same experimental floor usually have very different data formats and indexing requirements. Using text and/or hierarchical file systems such as HDF5 addresses some of these needs. File based data storage has served the legacy system in NSLS-I for almost 3 decades but the dramatic increase in data rates of NSLS2 beamlines made file I/O a massive bottleneck. In order to address these issues and provide NSLS2 beamlines with a uniform data storage solution that is applicable to all our beamlines, we have developed metadatastore service using NoSQL backend MongoDB [1].

ARCHITECTURE

metadatastore is the primary source of storage for scans, sweeps, etc. metadatastore is implemented as a RESTful web service on Tornado with a NoSQL MongoDB backend. Since it is document based, MongoDB does not have the traditional notion of tables. Each entry to the database is a document that is analogous to an "entry" in relational database terms. MongoDB collections serve as the containers of documents, providing its clients with means to meaningfully group their documents [2]. The metadatastore backend consists of 4 major collections: RunStart, EventDescriptor, Event, RunStop. One can think of RunStart and RunStop as the head and tail of a series of events that happen throughout an experiment (Figure 1). This approach eliminates the need to update the header of the experiment, which eliminates the problems that could occur due to lack of transactions in MongoDB. EventDescriptor(s) contain information regarding the data acquisition Event(s). In other words, they contain information about what are the

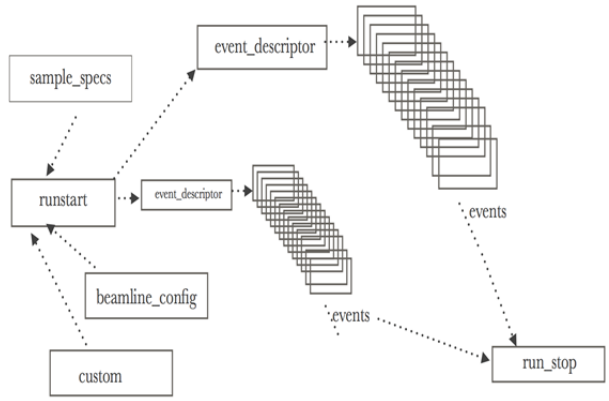


Figure 1: metadatastore Collection Relationship Overview.

"data_keys" that are being captured, what their shapes and data types are. This is immensely useful for data analysis toolkit in NSLS2 as this sort of header information is a major requirement to set up the data analysis environment prior to the arrival of actual data payload. In addition to the advantages provided so far, because of immense flexibility and performance of MongoDB, these RunStart, RunStop, and EventDescriptor documents support addition of any name-value pair or object in any hierarchy. Since Event(s) contain the actual data payload, they are defined as immutable documents. This semi-structured design pattern allows NSLS-II middle-layer to tackle many challenges including asynchronous scans, custom hardware brought from an external source, and varying data formats among experiments. For instance, a coherent X-ray diffraction and protein crystallography beamline utilize metadatastore in order to save their experiments despite their very different use-cases.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

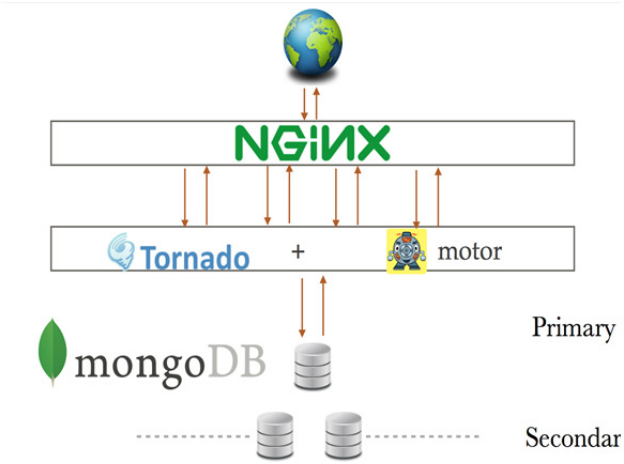


Figure 2: RESTful Web Service Overview.

The web service metadatastore is implemented in Python 2.7+ and Python 3. It includes some of the cutting edge developments in Python open-source community including asynchronous I/O. Historically, due to global interpreter lock, Python web applications has suffered from performance issues. This approach eliminates most of those problems, providing the scientific experiments with insert/query speeds far exceed their requirements (Figure 2). Tornado servers with 4 front-ends deployed with NGiNX load balancer on an Intel i7 processor can handle 8k+ requests per second.

FUTURE

Given the impact of metadatastore RESTful web service, there is an immense interest in developing GUIs and applications around metadatastore. JSON encoding might work for other industries that utilize MongoDB back-ends, however for scientific applications, they are not desirable due to performance and complexity of development as the GUIs need to have an understanding of data delivered for fast data processing pipelines. Due to its well-defined data layer pvData and high performance network protocol pvAccess, EPICS v4 is the best tool for this purpose. metadatastore documents can be served as NTTable or NTUnion over pvAccess to clients such as Control System Studio, which has support for EPICS v4. metadatastore utilizes the pipeline method that is custom built for streaming data in an extremely robust way. One of the major benefits of this protocol is the fact that all client side applications are essentially monitors. This allows extremely agile development, as one of the biggest over-head of development is to write layers that handle such handshake.

CONCLUSION

Metadatastore provides flexibl, high-performance, and rich semi-structured data storage for all beamline metadata. This service provides users with the ability to store asynchronous and hierarchical data of their choice, providing them with complex data regression capabilities. Compared to legacy approaches, due to its state of the art backend coupled with its narrow interface, metadatastore significantly reduces the time to locate any experimental run and its data points.

REFERENCES

[1] https://docs.mongodb.org/manual/core/data-model-design/

[2] NSLS-II High Level Application Infrastructure and Client API Design, G. Shen, L. Yang, K. Shroff Presented at the 2011 Particle Accelerator Conference

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

FILESTORE: A FILE MANAGEMENT TOOL FOR NSLS-II BEAMLINES

A. Arkilic, L. Dalesio, D. Chabot, W.K. Lewis, D. Allan, T. A. Caswell
Brookhaven National Laboratory, NSLSII, Upton, NY, USA

Abstract

NSLS-II beamlines can generate 72,000 data sets per day resulting in over 2 M data sets in one year. The large amount of data files generated by our beamlines poses a massive file management challenge. In response to this challenge, we have developed filestore, as means to provide users with an interface to stored data. By leveraging features of Python and MongoDB, filestore can store information regarding the location of a file, access and open the file, retrieve a given piece of data in that file, and provide users with a token, a unique identifier allowing them to retrieve each piece of data. filestore does not interfere with the file source or the storage method and supports any file format, making data within files available for NSLS-II data analysis environment.

INTRODUCTION

Current state-of-the-art detectors are capable of producing megapixel images with frame-rates in the kilohertz range which may result in peak data rates of up to 800 Mb.s-1. As a result, the volume of data associated with an experiment may now exceed several tens of terabytes so simply moving data from one location to another is prohibitive. As such, it may not be possible for external users to take a copy of all the data on portable media or to easily transfer over the Internet. Thus it requires high-performance data storage systems and advanced processing capabilities to allow users to access their data both during their experiment at NSLS-II and to continue the analysis following the NSLS-II visit. The middle-layer architecture of NSLS-II beamlines provides such capabilities using a blend of NoSQL databases and distributed file systems [2]. **fileStore** is the component of the middle-layer that keeps track of experimental files. These experimental files are generated via EPICS areaDetector module or any custom camera. filestore's NoSQL mongo backend allows users to save any sort of information about their images, while providing links to metadatastore runs[1]. Filestore provides clients with high-performance data mining techniques that were not possible via legacy applications in place.

ARCHITECTURE

filestore provides information regarding the location of the file within the General Parallel File System (GPFS). MongoDB serves as the back-end to filestore due to its flexibility and performance. Just like metadatastore, filestore has a set of required standard fields and allows users/data acquisition scripts to add any field in any hierarchy, making it a primary storage environment for image headers [1].

```
In [1]: from filestore.api import insert_resource, insert_datum

        resource_id = insert_resource('csv', 'example.csv')
```

Figure 1: filestore Python API Resource Insert Example.

Filestore's capabilities far exceed providing URL information. It also serves as a platform for performing data file formatting (via handlers) and data regression (via its complex querying capabilities). Experimental files can be stored completely independent of filestore's acknowledgement. Since filestore does not need to be aware of the file at time of creation, it is possible to migrate data files from legacy systems into the modern middle-layer service framework of NSLS-II beamlines (see Fig. 1).

```
In [5]: insert_datum(resource_id, 'some_id1', {'line_no': 1})
Out[5]: <Datum: Datum object>

In [6]: insert_datum(resource_id, 'some_id2', {'line_no': 2})
Out[6]: <Datum: Datum object>

In [7]: insert_datum(resource_id, 'some_id3', {'line_no': 3})
Out[7]: <Datum: Datum object>

In [8]: insert_datum(resource_id, 'some_id4', {'line_no': 4})
Out[8]: <Datum: Datum object>

In [9]: insert_datum(resource_id, 'some_id5', {'line_no': 5})
Out[9]: <Datum: Datum object>
```

```
In [10]: from filestore.api import register_handler

In [11]: register_handler('csv', CSVLineHandler)
```

Finally, we are ready to retrieve that data. All we need is the unique ID.

```
In [12]: from filestore.api import retrieve

In [13]: retrieve('some_id2')
Out[13]: 'b\n'
```

Figure 2: Datum and Handler example.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Write a Handler

```
In [14]: import h5py

In [15]: class HDF5DatasetHandler(object):
   ....:     def __init__(self, filename):
   ....:         self.file = h5py.File(filename)
   ....:     def __call__(self, key):
   ....:         return self.file[key].value
   ....:
```

Make a Record of the Data

```
In [16]: from filestore.api import insert_resource, insert_datum

In [17]: resource_id = insert_resource('hdf5-by-dataset', 'example.h5')

In [18]: insert_datum(resource_id, 'some_id10', {'key': 'A'})
Out[18]: <Datum: Datum object>

In [19]: insert_datum(resource_id, 'some_id11', {'key': 'B'})
Out[19]: <Datum: Datum object>

In [20]: insert_datum(resource_id, 'some_id12', {'key': 'C'})
Out[20]: <Datum: Datum object>
```

Retrieve the Data

```
In [21]: from filestore.api import register_handler, retrieve

In [22]: register_handler('hdf5-by-dataset', HDF5DatasetHandler)

In [23]: retrieve('some_id11')
Out[23]:
array([[1, 9, 4, 2, 4],
       [0, 1, 1, 8, 6],
       [6, 8, 3, 0, 2],
       [7, 3, 7, 8, 5],
       [1, 2, 8, 8, 2]])
```

Figure 3: A sample HDF5 file handler, save, and retrieve file example.

In NSLS-II experiments, experiment session manager library, **BlueSky**, notifies filestore regarding the arrival of the experimental data using filestore's Python client API. In order to save/retrieve image file(s), filestore requires two pieces of information: how to access and open the file (or, generically, "resource") and how to retrieve given pieces of data in that file (Fig. 2). Just like metadatastore, filestore provides a token, a unique identifier, which clients can use to retrieve each piece of data. This way, given an **event** in metadatastore, clients can access files generated during that specific event. **File Handlers** aim to eliminate differences in N-dimensional data representation by converting files with any extension to a **numpy** array (Fig 3). This allows filestore to support various beamlines as it provides support for any file format given clients provide the file handlers. It also makes it possible to develop data analysis tools that expect a standard input for N-dimensional data.

FUTURE

We are aiming to use EPICS v4 in order to stream the N-dimensional arrays in NSLS-II experiment files to clients. As seen in Fig. 4, specific v4 normative type, NTUnion, allows filestore to ship complex data structures that include the MongoDB documents that contain image header information and images themselves in rates that are much higher than traditional EPICS Channel Access Protocol.

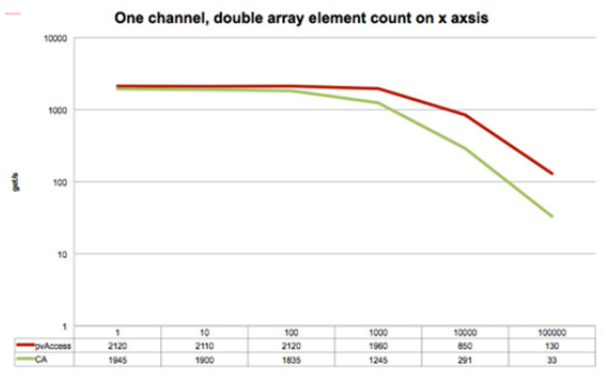


Figure 4: N-dimensional Array Stream CA vs pvAccess.

CONCLUSION

Filestore reduces the latency of accessing and moving of large data sets from minutes to seconds. Its modularity and ability to provide links to metadatastore events, allow NSLS-II data analysis environment to perform complex data regression operations. File Handlers makes it possible to write data analysis tools that expect single standard data format, numpy arrays.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REFERENCES

[1] metadatastore: A primary data store for NSLS-II beamlines, A. Arkilic, L. Dalesio, D. Allan, W. K. Lewis, Presented at the ICAPLEPCS 2015.

[2] NSLS-II High Level Application Infrastructure and Client API Design, G. Shen, L. Yang, K. Shroff Presented at the2011 Particle Accelerator Conference.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

LARGE GRAPH VISUALIZATION OF MILLIONS OF CONNECTIONS IN THE CERN CONTROL SYSTEM NETWORK TRAFFIC: ANALYSIS AND DESIGN OF ROUTING AND FIREWALL RULES WITH A NEW APPROACH

Luigi Gallerani, CERN, Geneva, Switzerland

Abstract

The CERN Technical Network (TN) TN was intended to be a network for accelerator and infrastructure operations. However, today, more than 60 million IP packets are routed every hour between the General Purpose Network (GPN) and the TN, involving more than 6000 different hosts. In order to improve the security of the accelerator control system, it is fundamental to understand the network traffic between the two networks and to define new appropriate routing and firewall rules without impacting operations. The complexity and huge size of the infrastructure and the number of protocols and services involved, have discouraged for years any attempt to understand and control the network traffic between the GPN and the TN. In this paper, we show a new way to solve the problem graphically. Combining the network traffic analysis with the use of large graph visualization algorithms we produced usable 2D large color topology maps of the network identifying the inter-relations of the control system machines and services, in a detail and clarity, not seen before.

INTRODUCTION

Improving the security between the GPN and the TN, where accelerator systems and control systems run, is a priority for the infrastructure experts in the CERN Beams Department, Control Group, and the Computer Security team.

The separation between the GPN and the TN is defined by routing rules controlled by the Network Operations database (NetOps or LanDB). Only GPN hosts that are TN-TRUSTED are able to connect to the TN, and only TN hosts that are GPN-EXPOSED can communicate with the GPN. From the CERN NetOps records, routing tables are generated and loaded in to the routers.

We have today more than 1000 machines that are in the TRUSTED or EXPOSED lists with a justified operational reason. Around 6000 machines are involved in the communications between the two networks. The TRUSTED – EXPOSED list mechanism is clearly too weak compared to the modern network protection standards. The goal is to have in place well defined firewall and routing rules and have full control on the traffic between TN and GPN. To achieve that on a large control system in operation, it is fundamental to and understand all the relations and dependencies between the system involved.

A good starting point is the record of the real traffic on routers.

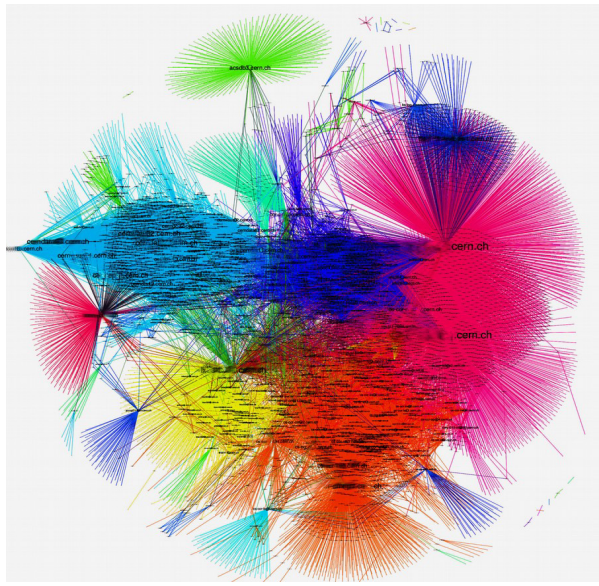


Figure 1: One hour of UDP-TCP traffic between CERN Technical Network and General Purpose Network plotted and clustered using Fruchterman Reingold algorithm.

RECORDING THE TRAFFIC

A large set of network restrictions rules can be easily defined by the knowledge of the system. However, this is not sufficient. With hundreds of different interconnected systems running on the TN and GPN, it is almost impossible to predict all the relations and dependencies between hosts and ports by the knowledge of the system functionality. The risk of compromising the connectivity of a running system because of a too strict or wrong firewall rule is what the Controls group wants to avoid most. Routers between the TN and GPN can record the traffic live producing as output text files with the following fields

Timestamp	Protocol	Packets count	Size bytes
Source host	Source port	Dest host	Dest port

DATA PROCESSING

Classical Approach by Statistics

In 1 hour, 60 million connections are recorded. 1.4 billion lines per day are the source of the analysis. Statistics, filtering, grouping and sorting are the obvious approaches for useful information extraction from a large dataset. For example it is easy to identify by queries the most connected hosts, the most used port between hosts of name N, the top 10 interconnected hosts and so on.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

However, finding anomalies in the network connections, searching for hidden dependencies between multiple interconnected systems, getting the flow of traffic from histogram and results from statistics and queries are not easy or practical at all.

Graphs, Conversion in .dot Language

Graphs are the ideal mathematical tool to display host interconnections. Hosts are shown as a node N. Each connection is an edge E with source-dest ports as edge property. Packet size or the connection count is the weight of the edge. Timestamps can be used as the 4th dimension to display the evolution of the graph in time. Protocols such as TCP or UDP can be easily separated or plotted with different graphic like style or colored edges.

To give an example, two simple recorded connections between 3 host, two clients and one server, Figure 2 can be converted to Graphiz ".dot" language [1] with:

```
strict digraph G {
hostA -> servA [label="80 to 36642"
                bytes=85920 color="red" flows=50]
servA -> hostB [label="58139 to 46113" bytes=628
                color="blue" flows=6 ]
}
```

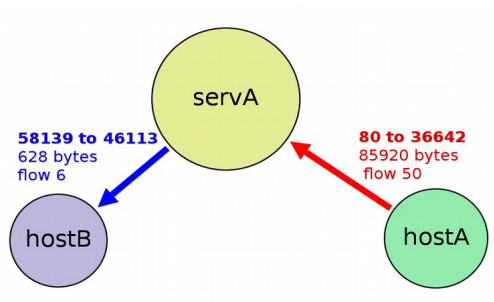


Figure 2: Direct graph automatically generated from the .dot language above using Graphiz and Gephi.

Graph Pre-Processor

Graphs directly generated from the recorded data have too many edges. Dot language interpreters automatically merge all the nodes entries with the same ID, but it is not immediate to do the same for the edges. We wrote a custom graph pre-processor script to perform the following operations and generate different types of sub-graphs in .dot format ready to be plotted.

1. Merge and count connections together hosts and port numbers (weighted edges)
2. Delete already defined connections (strict graph)
3. Ignore directions (undirected graph)
4. Separate in different graph for UDP and TCP

First Graph: Plotting 60M Nodes

To plot the obtained .dot graph "Gephi" [2] was used on "Debian 8" with "pekwm"[3] as X windows manager (Gephi is unstable on sophisticated windows managers like gnome/kde). Figure 3 shows the first graph obtained without edge labels and with random node positions. The complexity of the problem is evident, so additional steps are required to get better results.

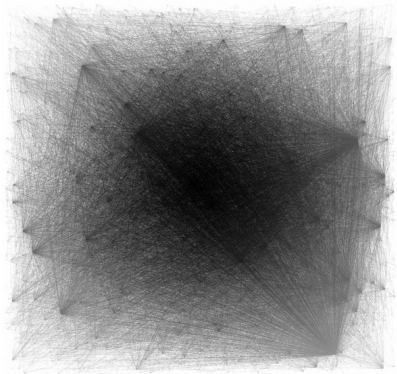


Figure 3: One hour of traffic: ~6K nodes with random position interconnected by 60M edges. Each node is an host, each edge is a TCP or UPD connction recorded.

GRAPH PROCESSING

Clustering and Neighbor Discovery

The first graphical computational step is called clustering [4] and Gephi already implements many different algorithms. Starting from a random node position graph like the one in Figure 3 a filter on nodes degree like K>3 is applied, then the Reingold Fruchterman [5] clustering process is started, and only at the end of the process, the filter is removed. A sequence of the clustering process is shown in Figure 4. The nodes with high degree K are kept in the center, and clusters of isolated hosts are moved to the edges.

Neighbours are discovered with the Chinese Whisper algorithm [6] so that nodes in the same cluster have the same colors and are easily identified. Finally, labels are applied. A full graph processing with this method is shown in Figure 1. In this article we show graphs with masked labels or replaced with generic ones.

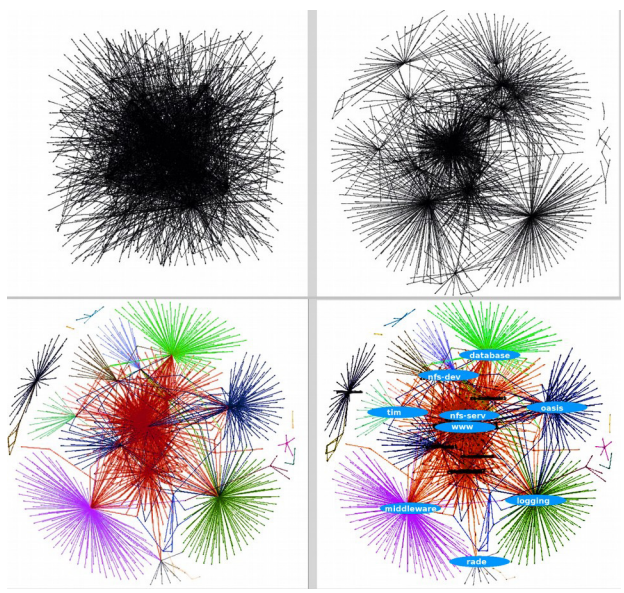


Figure 4: Clustering process sequence with Reingold Fruchterman algorithm followed by neighbor discovery colors by Chinese Whisper algorithm and label of nodes with high K degree.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

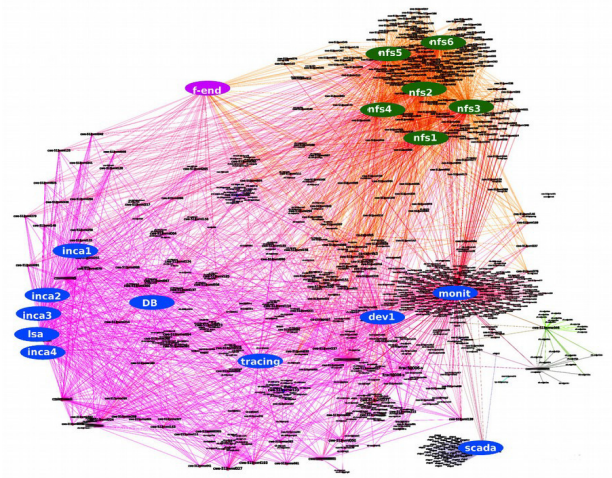


Figure 5: Cluster of around 400 nodes showing controls development machines dependencies in a strict direct graph over 24 hours in full resolution.

Sub-Graphs at Full Resolution

Once the main clusters of machines are identified, smaller and full detailed sub-graphs can be created. With the combined usage of the pre-processor and the filtering by queries inside Gephi, it is possible to generate full resolution sub-graphs like the one shown in Figure 5 where the TCP traffic generated in 24 hours by more than 400 development virtual machines is clearly plotted. To read all the information we print the high resolution graph in large format (ARCH E size). Using different pre-processor configurations, a full graph of port-to-port interconnections for same cluster is generated: Figure 6

Multi Graph Integration

Multiple graphs can also be combined together. A graph of host relations from NetOps has been combined with the recorded traffic graph. Hosts in the same network group are seen as a single large node. Connections of the similar

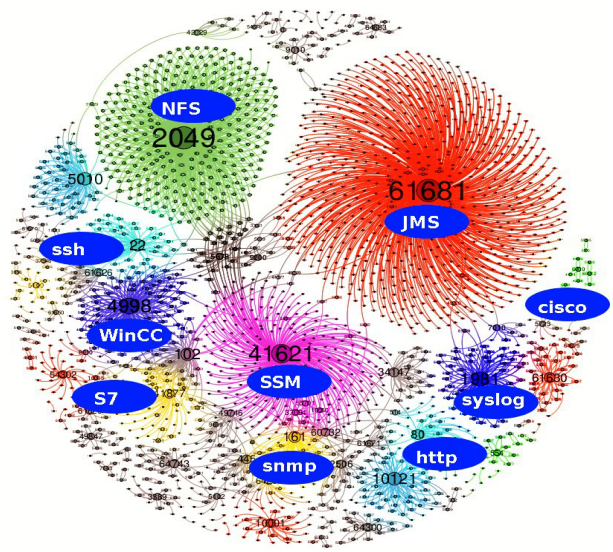


Figure 6: Port to port subgraph of the virtual machine cluster allows easily identifying the interconnections.

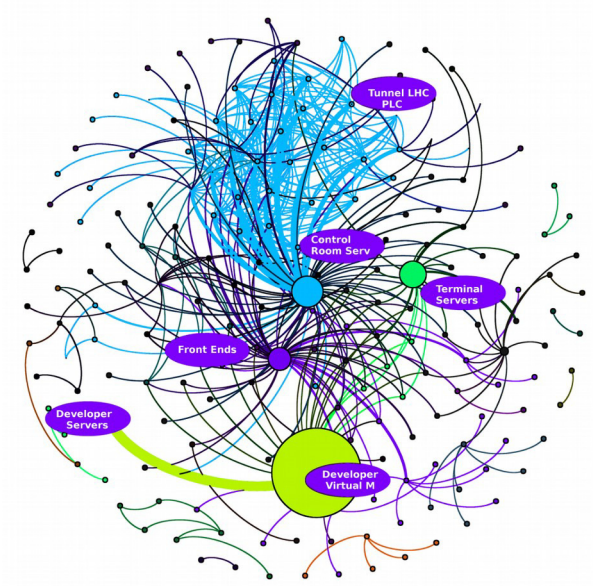


Figure 7: Combination of recorded data full resolution graphs with Network database info in a weighted edge graph where similar nodes are grouped together.

types are also grouped together in a larger edge line proportional to the enumeration, as visible in Figure 7

SECURITY FROM GRAPH ANALYSIS

Graph analysis returned highly valuable information that we converted in practical actions to improve the network security. For example, we identified and removed hidden dependencies for terminal servers, forgotten port scanners, wrong inclusions of development machines in operational PLC set. We moved the monitoring cluster out of the TN for more then 500 TRUSTED hosts. We have started to define firewall rules based on the analysis.

CONCLUSION

We used large graphs visualization techniques to plot million of recorded connections for machines involved in the controls system network. With the results we have been able to identify and fix many issues, have a view of all the dependencies and a new tool in place to define firewall and routing rules. This method can be easily adapted and widely used by the community involved in large control system network administration and protection.

REFERENCES

[1] Graphviz library www.graphviz.org
[2] Gephi Open Graph Viz platform gephi.github.io
[3] Pekwm Win manager www.pekwm.org
[4] V. Dongen, “Graph Clustering by Flow Simulation” PhD Thesis, University of Utrecht, The Netherlands.
[5] T. Fruchterman,;E. Reingold "Graph Drawing by Force-Directed Placement", Software – Practice & Experience (Wiley).
[6] C. Biemann, "Chinese Whispers an Efficient Graph Clustering Algorithm" University of Leipzig.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

TOWARDS A SECOND GENERATION DATA ANALYSIS FRAMEWORK FOR LHC TRANSIENT DATA RECORDING

S. Boychenko, C. Aguilera-Padilla, M. Dragu, M.A. Galilee, J.C. Garnier, M. Koza, K. Krol, R. Orlandi, M.C. Poeschl, T.M. Ribeiro, K. Stamos, M. Zerlauth CERN, Geneva, Switzerland
M.Z. Rela CISUC, University of Coimbra, Portugal

Abstract

During the last two years, CERNs Large Hadron Collider (LHC) and most of its equipment systems were upgraded to collide particles at an energy level twice higher compared to the first operational period between 2010 and 2013. System upgrades and the increased machine energy represent new challenges for the analysis of transient data recordings, which have to be both dependable and fast. With the LHC having operated for many years already, statistical and trend analysis across the collected data sets is a growing requirement, highlighting several constraints and limitations imposed by the current software and data storage ecosystem. Based on several analysis use-cases, this paper highlights the most important aspects and ideas towards an improved, second generation data analysis framework to serve a large variety of equipment experts and operation crews in their daily work.

INTRODUCTION

The consolidation and upgrade activities during the recent long shutdown has allowed almost doubling the collision beam energy to 13 TeV. This implies as well a much increased damage potential in case of the serious failure in comparison to previous runs. As the rest of the machine, the systems which protect the LHC have equally undergone major improvements based on the experience collected during the first operational run between 2010 and 2013. Besides the increased number of machine protection devices installed in the machine, the amount of data which they produce also increased. The software which is used for analysis of accelerator diagnostic data requires a major upgrade to keep up with the LHC expansion and the unprecedented amounts of data to be processed [1].

Based on experience of the previous hardware commissioning we have identified the main shortcomings with the current data analysis approach. Combined with the use cases collected over several operational years and the feedback provided by operators and hardware experts, we were able to define the main goals and requirements for the next generation accelerator data analysis framework. In order to solve the identified problems, an extensive study was performed, followed by a comparison between existing data storage and processing tools. Although, the integration of modern data analysis frameworks might improve the situation with respect to the currently implemented approach, we believe that specificities of collected data and environment itself pose a greater challenge. Based on the initial research, a new workload-driven approach for organizing the data is proposed, aiming to serve efficiently heterogeneous workloads. The strengths and the weaknesses of the proposed solution are being analysed, as well as we discuss the details of the integration of the proposed technique with currently deployed accelerator software stack. Finally, the goals for the future research and development of the new analysis framework are defined.

The remaining document is organized as follows: next section provides a short overview on the current analysis framework shortcomings and desired requirements for the new solution. The following section describes the novel approach which is believed to allow building the system capable of satisfying the majority of the use cases. In future work we outline the goals for future research and development. Finally in the last section a summary on the main conclusions is presented.

CURRENT ANALYSIS FRAMEWORK

During the last few years, significant efforts were made to initiate the process of LHC transient data analysis centralization. The operators and equipment experts were previously relying on multiple service-specific graphical tools to study the underlying data. The process required significant effort and concentration, especially during accelerator commissioning when thousands of tests are executed in order to validate the behaviour of control systems. Furthermore, the data correlation between different storages was either performed manually or required the development of custom modules, which would take care of the required merging process. A recent attempt on automatizing the validation process for LHC protection systems resulted in the organization and systematization of the analysis process. The results were consolidated into the AccTesting framework [2] which has proven its efficiency during the last LHC hardware commissioning phase.

Infrastructure

There are two major sources of LHC diagnostic data: The Post Mortem (PM) framework [3] and CERNs Accelerator Logging Service (CALS) [4]. Despite acquiring the similar data from a given LHC device, the two services have fundamentally different use cases and properties. The main differences between the two frameworks is the resolution of collected data. The Post Mortem system only collects data around interesting events, by retrieving all the entries from the internal buffer of the monitored devices.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Internal device buffers record the data with very high frequency (up to nanosecond precision), allowing to reconstruct very precisely the accelerator and equipment system states around events such as the beam extraction. Given the amount of the devices and produced data quantities it is impossible to store all this information continuously with the same acquisition frequency during operation. On the other hand, the collection of continuous low-frequency data provides a broader overview on eventual root causes of problems. This requirement is satisfied by the CERN Accelerator Logging Service, which retrieves the data on change with a maximum frequency of a few Hz. The collected information is used not only to understand failure sources but also to analyse the long-term performance and evolution of LHC operation.

On top of the aforementioned storage systems, users can either implement external modules to perform automatic data analysis or run arbitrary data extractions. In the first case, the modules either subscribe to determined devices or poll the data continuously from the storage, in order to identify or classify complex events. In case of manual analysis, users either use the available service-specific graphical tools (which mostly do not allow for correlation of data between the sources), or provided APIs [5] which require programming skills to be used efficiently. Recently, developments towards a data querying engine have started based on specifically designed for accelerator environment Java embedded Domain Specific Language (eDSL) [6]. This eDSL provides an abstraction layer over data extraction from different storages (see Figure 1). Queries are being written in English-like syntax language, specifically designed for CERNs accelerator environment. Currently eDSL supports assertions on multiple signals, which were required to analyse the majority of the hardware tests during the last LHC commissioning phase.

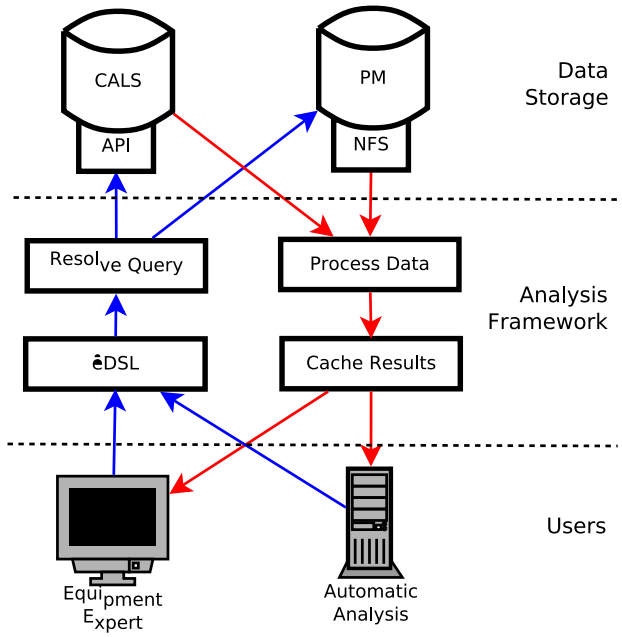


Figure 1: Java embedded Domain Specific Language data analysis framework work flow. Instead of pushing the responsibility of data analysis to users, the eDSL based solution performs analysis its servers.

Next Generation Analysis Framework Requirements

The experience of the last commissioning and operational period of the accelerator indicates the heterogeneity of the workload to be handled by the framework. When the accelerator systems are being tested the analysis of localized device types or sector based data is being performed for short time intervals (related to test execution periods). When the machine is in operation the data from the past few weeks might be retrieved to study the behaviour of specific equipment. During long shutdown phases a more extensive analysis of the overall performance of the LHC is conducted to detect the most common failure sources or to determine the efficiency of machine operation and the resulting physics output. This use case requires the extraction of data for large periods of time.

The analysis of the downsides of the currently deployed solution suggests that the new framework should be flexible enough to perform calculations close to the data source. In many use cases an expert is interested only in a small fraction of data resulting from simple aggregation operations. The limitations of the current systems impose the transfer of the whole data set in order to perform the required calculation on the users' infrastructure. The data transfer can be further optimized through efficient caching mechanisms.

A successful integration of the system into the existing accelerator environment implies the integration of the currently developed and maintained eDSL, specifically designed for accelerator diagnostic data analysis. The integration with eDSL is among the most prioritized requirements, since most of the operators are coming from areas other than software engineering. Providing a querying language specific to their respective domain would flatten significantly the learning curve needed with more generic languages and leave more time to focus on the data analysis.

Among the main constraints which prevents the system development to advance more rapidly is the performance of the retrieval of the data from underlying storages. Currently deployed storage systems provide restrictive APIs and limited access to the infrastructure, preventing the execution of optimizations which would allow the analysis framework to satisfy many of the aforementioned requirements. While the Post Mortem system is partially maintained by the Machine Protection and Electrical quality assurance group, the Accelerator Logging Service is maintained by the controls group of CERN in conjunction with the IT teams. In a recent collaboration agreement the details of possible extension of the CALS service were discussed in order to improve the access performance to the stored information.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

MIXED PARTITIONING SCHEME REPLICATION

After identifying the problems of the current solution and defining requirements for the proposed new analysis framework, a detailed study of the currently existing solutions for analysis of a large data sets was conducted. An important factor which was taken into account during the study was the predominant time series nature of the data. The signals with respective variables are being stored in simple structures with corresponding measurement timestamps. Once collected and stored in a database, the signal value will never be updated, thus both CALS and PM are following "write once read many" paradigm. On the other hand, the identified requirements suggest that other data dimensions should be taken into account when optimizing the storage for heterogeneous workload (for example: device type, accelerator state the measurement is related to, etc). Finally we concluded that there are multiple tools and techniques which could improve the current situation and boost data read throughput. Multiple characteristics of the data combined with the orthogonal use cases will however quickly bring forward the limitations of the current infrastructure. To overcome these predicted limitations, we propose a new strategy for data storage, named mixed partitioning scheme replication.

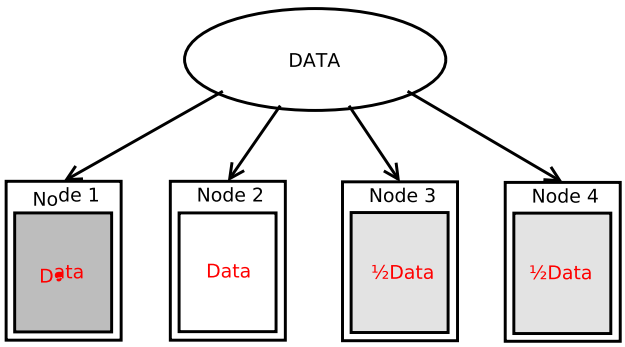


Figure 2: Basic hypothetical mixed partitioning scheme replication. The same data is partitioned differently on infrastructure nodes.

The infrastructure which is taken as an example and presented in Figure 2 consists of four nodes, with a replication factor of three (which is the recommended configuration in HDFS [7]). The number of nodes is chosen specifically to demonstrate how the data is organized when there are additional available resources. The replicas which are connected to infrastructure organize the data using different criteria. The color inside the box corresponds to different partitioning strategies. In case of Node 3 and Node 4, the data is organized using the same strategy, but is shared amongst the available resources to distribute the load evenly. Since the proposed technique is workload based, the decision to share the data among the available resources depends on the load of the participating replicas (additional resources will reduce the workload on the most busy nodes). It is important to note, that once the strategy is chosen and the data is stored, the schema does not evolve with time.

The main advantage of the proposed solution is the capability of handling efficiently highly heterogeneous workloads since different partitioning strategies can be adapted independently for particular query categories. On the other hand, the presented approach has a significant drawback related with node load balancing dependency on determined time periods. For example, during the accelerator commissioning phase, it is very unlikely that there will be some long term LHC performance analysis executed, while during the shutdown phase the probability of serving such queries is much higher in comparison to equipment test specific workloads.

To solve the load balancing problem, we propose a slightly different approach, based on the same principles of data replication using different partitioning techniques. In the Figure 3 the scenario remains the same, but in this case partitioning is done in two phases. An initial data division is being performed according to multi-dimension criteria which depends on data and workload characteristics adapted to the existing infrastructure configuration.

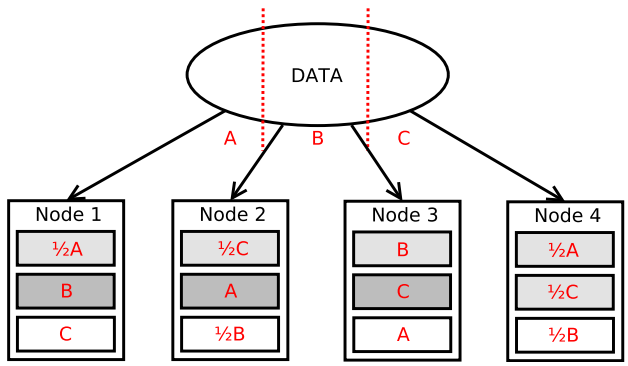


Figure 3: Complex hypothetical mixed partitioning scheme replication. The data is partitioned in two phases.

After the initial partitioning, data segments are being stored within the replicas using different partitioning strategies. Similarly to simple mixed partitioning scheme replication, the data organization is workload-driven and the busiest data segments are the first to be shared with additional resources. The main advantage of the proposed solution is the possibility of finding the ideal combination of storage strategies to optimize the load distribution throughout the whole infrastructure. On the other hand, there are several shortcomings as well with this approach. First of all, we foresee that strong data consistency mechanisms which are imposed by analysis types could impose some limitations on the system's scalability. Additionally, the complexity of the solution might raise maintainability challenges for the underlying infrastructure administrators especially in case of failure recovery mechanisms since the recovery process will require the translation of data from one scheme to another. All aforementioned factors will

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

require detailed analysis to understand if the gained performance is enough to outweigh the downsides of the approach.

The proposed solution could be initially deployed on an intermediate storage component enhanced with the functionalities of modern engines for large-scale data processing (Spark [8] or Impala [9] for example). In addition to the possibility of working with data through eDSL, users will have the possibility to execute arbitrary requests on the data using the data processing engines native query language. The choice of data serialization and storage solution will depend on its compatibility with the chosen data processing engine and its performance with heterogeneous workloads. Initially the data will be fetched from CALS and PM. Before being available for users, the data will be pre-processed, correlated and merged, to abstract the data source where it was originally fetched from. A possible integration scenario of a mixed partitioning scheme replication with new technologies and the proposed infrastructure is depicted in the Figure 4.

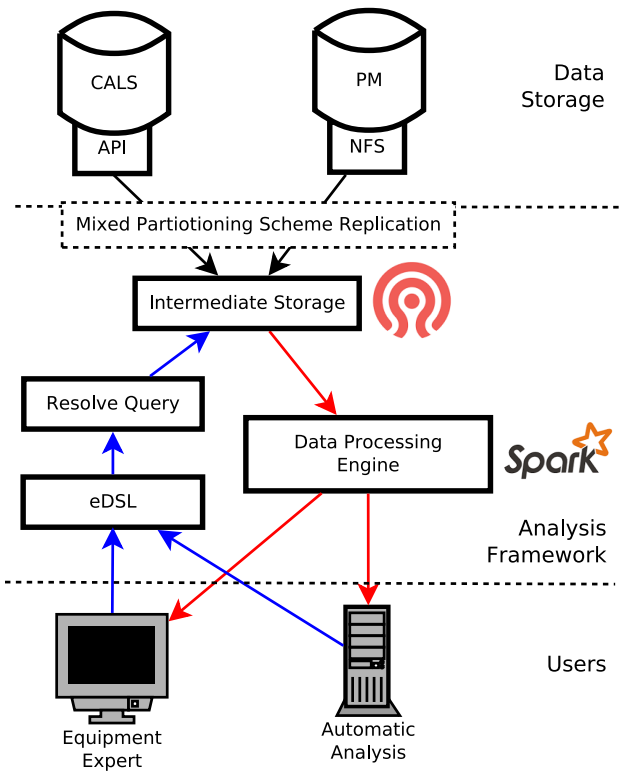


Figure 4: Mixed partitioning scheme replication integration into current infrastructure. In portrayed scenario, the data storage solution and data processing engine chosen are CEPH [10] and Spark, respectively.

FUTURE WORK

In the near future our efforts will focus on building a model approximating the real algorithm implementation which will allow to achieve early feedback about the benefits and eventual shortcomings of the proposed approach. Besides determining the efficiency of the proposed solution, the model would allow to understand how different parameters could affect the final system behaviour for different workloads. The plan is to start with a simple one node simulation, extend it to multi-node environments and finally validate the predicted performance by benchmarking the currently deployed infrastructure. In the end, the precision of the model will directly influence the request routing in the mixed partitioning scheme replication, since the algorithm should be able to determine efficiently the best resource to execute the query.

CONCLUSION

In order to address the new requirements and enhancement of the LHC for its second and subsequent operational running periods, the accelerator transient data analysis framework requires a, more performant data storage solution. Studies of the currently deployed analysis framework and literature suggest that the currently operating solutions will only be able to cover partially the required features of the data storage system for a highly performing analysis framework. A novel solution is being proposed to address both the desired characteristics as well as enhancing the storage performance.

REFERENCES

[1] K. Fuchsberger et al.,"Concept and Prototype for a Distributed Analysis Framework for LHC Machine Data", ICALEPCS'2013, San Francisco, CA, USA (2013).

[2] D. Anderson et al.,"The AccTesting Framework: An Extensible Framework for Accelerator Commissioning and Systematic Testing", ICALEPCS'2013, San Francisco, CA, USA (2013).

[3] O. Andreassen et al.,"The LHC Post Mortem Analysis Framework", ICALEPCS'2009, Kobe, Japan (2009).

[4] C. Roderick et al., "The CERN Accelerator Measurement Database: On The Road To Federation", ICALEPCS'2011, Grenoble, France (2011).

[5] C. Aguilera-Padilla et al., "Smooth migration of CERN post mortem service to a horizontally scalable service", WEPGF047, these proceedings, ICALEPCS'2015, Melbourne, Australia (2015).

[6] M. Audrain et al., "Using a Java Embedded Domain-Specific Language for LHC Test Analysis", ICALEPCS'2013, San Francisco, CA, USA (2013).

[7] https://hadoop.apache.org/

[8] http://spark.apache.org/

[9] http://impala.io/

[10] http://ceph.com/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SMOOTH MIGRATION OF CERN POST MORTEM SERVICE TO A HORIZONTALLY SCALABLE SERVICE

C. Aguilera-Padilla, S. Boychenko, M.Dragu, M.A. Galilee, J.C. Garnier, M. Koza, K. Krol, R. Orlandi, M.C. Poeschl, T.M. Ribeiro, M. Zerlauth, CERN, Geneva, Switzerland

Abstract

The Post Mortem service for the CERN accelerator complex stores and analyses transient data recordings of various equipment systems following certain events, like a beam dump or magnet quenches. The main purpose of this framework is to provide fast and reliable diagnostic to the equipment experts and operation crews to help them decide whether accelerator operation can continue safely or whether an intervention is required. While the Post Mortem System was initially designed to serve the CERN Large Hadron Collider (LHC), its scope has been rapidly extended to include as well External Post Operational Checks and Injection Quality Checks in the LHC and its injector complex. These new use cases impose more stringent time-constraints on the storage and analysis of data, calling for a migration of the system towards better scalability in terms of storage capacity as well as I/O throughput. This paper presents an overview of the current service, the ongoing investigations and plans towards a scalable data storage solution and API, as well as the proposed strategy to ensure an entirely smooth transition for the current Post Mortem users.

INTRODUCTION

The Post Mortem (PM [1]) service has been providing data collection, storage and analysis of LHC event data since 2008. The PM system usage grew significantly from the first use cases covering the understanding of the conditions in which the machine aborted its operations: fault on an equipment, powering event, etc. It currently covers the Injection Quality Checks (IQC [2]) from the SPS to the LHC and the Extraction Post Operational Checks (XPOC [3]) to verify that the beam extraction from the LHC was correctly executed. A new use case was just implemented to analyze every single cycle of the SPS. A possible integration of LINAC4 is under study. The initial design of the PM systems was to collect data on every beam dump, e.g. every 8 hours under normal operation conditions, or multiple times an hour when problems arise. The newer use cases call for real-time constraints. The analysis result for XPOC must be delivered within 10 seconds, an SPS cycle analysis must be given within 7 seconds in order to use the result on the next cycle. A LINAC4 cycle lasts about 1 seconds so an analysis that interlocks the next cycle requires real-time constraints for the data transfer protocol and for the analysis framework.

In its current design, the PM system provides a good vertical scalability, meaning that resources can be added to the nodes with minimum downtime and impact on the service availability. This way it could satisfy the original use case and withstand the first extensions. The increase in data throughput to the PM system and the SPS use case integration both demonstrated that its horizontal scalability must be improved to sustain all its current and future use cases as well as to accommodate larger events and analysis processes.

The first Section presents the current functional design of the Post Mortem service. The second Section presents the architecture of the PM system to support the functional design and identifies the current shortcomings. The third Section presents the improvements we propose, the steps that are planned to achieve the migration, and the first results of the migration.

FUNCTIONAL BEHAVIOR

The LHC control devices are continuously recording information about their current state in a rolling buffer which they are ready to dump on demand from an external event, or on their own trigger. The data dump is identified by the device information and by an “event time stamp” given by the device. Typical events are beam dump, powering faults, self triggers, etc. If the data dump could not be performed completely and the data were not serialized to the permanent storage despite the fallback mechanism, the device is warned so that it can store the data locally on the front-end controller storage and try to dump them later on.

All the collected data dumps are forwarded to the Post Mortem event-builder. The event-building is data driven. When it receives a first data dump coming from a control device, it sets a data collection time window. All the subsequent data dumps arriving during this time window with an event time stamp close to the one of the first one will be associated to the event. The event-builder identifies and characterizes the type of the event according to the data dump patterns. If the event-builder only received data from power converters and quench protection systems, it is considered a powering event. If it received data from every LHC system, this is a global event which is in average 190 MB.

The event is then analyzed by a hierarchy of analysis modules. Modules can be dedicated to the analysis of a single type of system, or they can perform correlation between multiple systems.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ARCHITECTURE

Control devices around the LHC use a Post Mortem client library to create and dump data to the Post Mortem service. The data are first received by Front-End servers that are dedicated to a system type, e.g. one Front-End server will collect all the data from the Quench Protection System, another one will collect all the data from the LHC Beam Dump System, as shown in Figure 1. The destination Front-End system is selected by the client library.

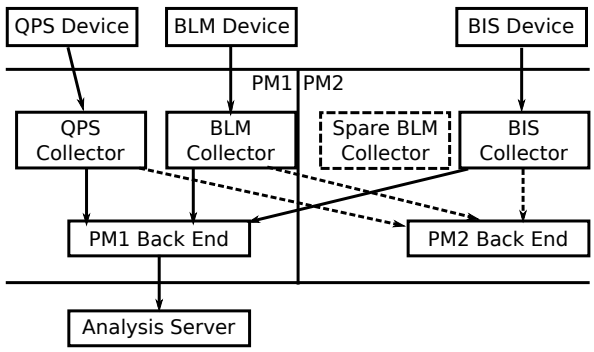


Figure 1: The Post Mortem data collection. The dashed lines represent the redundant collection path to the secondary Back-End server. The event-builder and the analysis tools are only attached to the main PM1 Back-End server. Devices can also dump to spare Front-End servers if the main one is not reactive.

Static Load Distribution

At first, the data flow was optimized between the redundant server nodes so that the load coming from the Front-End servers dump requests would be more or less evenly distributed. The data sizes, patterns and frequencies changed with time. The initial load-distribution is not adapted anymore and one server is clearly more stressed than the other. For a global event, in the current situation, one node collects in average 125 dumps that amounts to 40 MB of compressed data, while the other node collects in average 4073 dumps that amount to 150 MB. The total size of a global event collection varies depending on the cause of the event as some devices send buffers only when they are concerned. Other types of events follow different distributions of data per nodes. In addition, the PM system accepts a continuous flow of data. Due to the way the load distribution is implemented, it is difficult to scale it to cover optimally all use cases.

Monolithic Implementation

The production Front-End and Back-End servers are currently implemented in C++. The application protocol guaranteeing to the device that its data dump was completely stored is based on synchronous calls. The device dumping process will know that the data are completely written if the call to dump data returns successfully, e.g. without any exception.

The data collection servers are implemented in such a way that they do not handle data dumps in parallel but rather serializes them. The bigger the event, the more time is spent on collecting the data. It happens very rarely that a dumping device receives an RDA timeout while dumping data.

Data Consistency

Front-End servers store the data on a fast access non-permanent file system. They forward the data to Back-End servers that will store it on a permanent storage and notify the event-builder. There are two Back-End servers for redundancy. The Back-End processes enrich the data with information, like the reception time on the back-end system, or the path to the data on the file system. While the PM service ensures data storage redundancy, the data consistency cannot be ensured by standard tools due to the information added by the Back-End processes.

Data Storage Exposition

The file system is exposed via NFS to all the Post Mortem users, e.g. the analysis services and the equipment experts running custom analysis written in LabVIEW or Java. Until now it was not possible to perform analysis in another technology. Clients access the file system structure directly, either to use the raw data or to convert it into other formats and store it back on the Post Mortem storage. Therefore, PM data are duplicated to serve different use cases.

When the file system is under heavy load it causes delays in the file writing, which subsequently causes delays in the handling of the dump request by the data collection servers, which ultimately can trigger a time out in the dumping device.

Shortcomings

We identified the following shortcomings in the current architecture:

- Static load distribution
- Lack of data consistency verification
- Storage architecture and data format exposed to the end-users
- Data duplication
- File system performance limits and back pressure
- Monolithic implementation of data collectors
- Neither delivery nor ordering guaranty of data dumps

All this leads to performance limitations of the PM system that impact the implementation of new use cases. This calls for upgrades of the current system. The shortcomings are mainly due to in-house design of redundancy and distribution paradigms. The goal of the upgrade is to re-use available technologies to achieve fully horizontally scalable data collection, storage and analysis.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The data transfer from devices to the event-builder relies entirely on the CERN Control's MiddleWare (CMW [4]) RDA protocol. The current data collection uses RDA2 based on CORBA [5], with an on-going migration to RDA3 based on ZeroMQ [6]. The migration from RDA2 to RDA3 is on-going at the level of control devices. As long as some devices rely on RDA2, the current PM infrastructure must be maintained. There are currently 999 PM clients based on RDA2 and 915 based on RDA3. Multiple control devices can share the same RDA.

The challenge is therefore to upgrade the PM system while keeping it backward compatible for clients that would migrate to RDA3 slowly.

ROADMAP

The long term plan is to make the Post Mortem service horizontally scalable in all aspects: data collection, storage and analysis. This requires a drastic modification of the storage system. To modify the data format and the data storage, it is necessary to encourage users to use a common API to read and dump data as a first step. That way the data format and storage will be considered as a black box and it will be then possible to modify it at will.

Data Collection

The client API used to dump data is providing proper abstraction to the users. The API is implemented in C++ and in Java. It exposes the RDA serialization of the data to the user. This means a user knows RDA is used to dump data, and it knows that the data are serialized with CMW/RDA for the transport to the Post Mortem service. The client API currently supports RDA2 and RDA3 protocols. The exposure of the RDA protocols through this API is not a big issue, as user processes already rely heavily on these technologies. The client API is now integrated into the FESA framework [7], which is used by most of the control devices at CERN.

The missing feature for the data collection is dynamic load distribution, meaning that a client would dump Post Mortem data to any of the available data collection servers. This feature must be integrated into CMW/RDA3 before being used in the PM system. Alternatives could be to rely on other message protocols, like the underlying ZeroMQ protocol. The challenge is not to introduce a single point of failure.

Having introduced a dynamic load distribution, the client library will not be coupled anymore to specific PM targets. This means that a client will not dump to a specific data collection process anymore, but to any PM data collection process. This will increase the availability of the service and its maintainability. It will also allow the PM system not to rely on the Front-End/Back-End process anymore, but on a single layer of data collection process, as shown in Figure 2.

Data Access

The APIs to access data are implemented in Java or based on files. They both rely on the NFS exposure of the data.

A REST (Representational State Transfer) API is currently being implemented. The aim of this API is to serve multiple language technologies. Many users would like to perform analysis in Python or MATLAB. It will also provide data to the LabVIEW analysis framework. This way, no user will directly depend on the data serialization format or the file system. The data are exposed in plain JSON [8] or in Apache Avro [9]. The REST API is built incrementally. It currently provides the minimum amount of features for users to search for data using a few basic criteria. The aim of the first release is to provide the features that users implemented on top of the storage system, e.g. extracting directly one signal from the data without reading it entirely. This will encourage them to use the API and hide the file details. The overhead of accessing data through the REST API is very low. It adds an average delay of 8 ms to the file access, and a worst case delay of 73 ms when retrieving today's largest PM data.

The long term goal for the Post Mortem API is to enable complex queries, e.g. "Get all the Quench Protection Data when the Beam Energy was greater than 6.5 TeV". This can be achieved by extending the first release, adding indexing and correlating features. Open source products will be studied more thoroughly to address this requirement.

Storage

Once the data will be accessed only through the Post Mortem APIs and not directly by the users, the main upgrade of the Post Mortem storage will be possible.

The goal here is to provide complete data redundancy and integrity checks, as well as horizontal scalability. The requirements for the Post-Mortem storage are:

- Data replication
- High availability
- Error detection and correction
- High storage capacity
- High read and write throughput

The PM data which can be considered as semi-unstructured data. The structure is defined by the users and evolves with time. Quench Protection System data from 2008 does not look like data from 2015. In these conditions, it is difficult to restructure the data in a different format to store it. The data can only be stored in the same structure it was sent, as an immutable snapshot. Distributed object storage systems can therefore be considered as a good alternative to the current file system implementation. Distributed file systems are also a good option as they share many features with distributed objects storage systems.

Numerous solutions are available on the market. The Ceph [10] object storage, the Gluster [11] and HDFS [12] file systems are being considered for this upgrade. The plan

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

is to make an inventory of these technologies and to evaluate them according to our storage requirements.

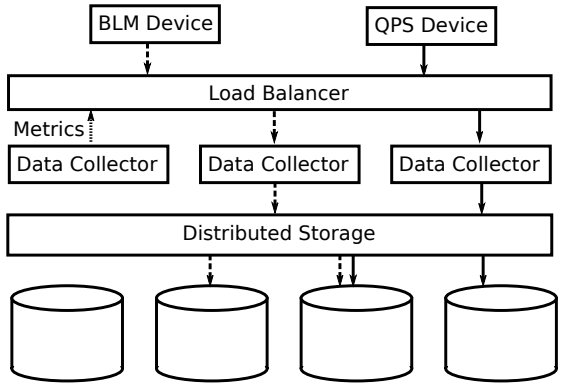


Figure 2: The targeted simplified Post Mortem architecture. Devices dump Post Mortem data to Data Collectors that are part of a load-balancing scheme. This way the load can be distributed among the Data Collectors of the Post Mortem cluster. The Data Collector that processes the dump writes the data to the distributed storage. The data are automatically replicated by the storage infrastructure.

The way to serialize files will also evolve. The plan is to enhance the usage of PM data for analysis. Considering that users may run analysis only on parts of the data dumps, the files must be splittable. Compression is also required to optimize the occupied space in the storage system. Indeed the signals often contain zeros and therefore have a high compression rate. After a thorough evaluation of multiple technologies, Apache Avro has been selected. The files can be compressed and still splittable at the same time, as the compression is performed on blocks of the Avro format, and not on the entire file itself. Avro also brings the flexibility required for the data structure to evolve. There is no need to generate specific classes to interpret the data, as the data themselves can embed the schema required to interpret them. The data will therefore still be self-described. The fact that Avro is a key technology for Mapreduce [13] jobs is another motivation for the choice of this file format.

Upgrade Steps

In order to perform an upgrade as smooth as possible for the users, the following steps are necessary:

1. Provide APIs to deprecate direct storage access
   (a) Unique API for all languages to access data
   (b) Data dumps from both CMW RDA2 and RDA3
2. Commission new data storage and serialization in parallel to operations
   (a) Keep the current storage and its APIs running
   (b) Automatic or semi-automatic data collection forwarding to new storage
   (c) Run spare data analysis tools on new storage using same compatible API
3. Decommission old PM system
   (a) Transfer all data to new storage
   (b) Move production APIs to new storage, keeping them backward compatible
   (c) Switch off

The first step is currently in the development phase and incremental releases are already performed for production usage. Storage solutions are being studied in parallel. As the PM system will not expose the storage nor the data format, it will be easier to perform migrations in the future.

CONCLUSION

The current Post Mortem system is still fulfilling its use cases successfully, and a second instance has been deployed to perform the SPS Quality Checks. The restart of the LHC after the first long shutdown and the growing need of more accurate analysis on higher data resolution in a deterministic time has pushed the Post Mortem system to its limits. The limitations of the current systems are clearly identified and the work already started to scale horizontally the Post Mortem system. This entire upgrade is the opportunity to make the system more maintainable and ease the future upgrades, providing a clear segmentation of the different areas of concern: APIs, storage architecture, data format, data analysis. It is also the opportunity to bring new features, e.g. allowing any language to work on PM data. Improving the storage layer is furthermore the opportunity to move toward data analytics on Post Mortem data, and enabling correlation of data from other CERN data storage as presented in [14].

REFERENCES

[1] O. Andreassen et al., "The LHC Post Mortem Analysis Framework", proc. of ICALEPCS 2009, Kobe, Japan.

[2] V. Kain et al., "Injection Beam Loss and Beam Quality Checks for the LHC", proc. of IPAC10, Kyoto, Japan.

[3] N. Magnin et al., "External Post-Operational Checks for the LHC Beam Dumping System", proc. of ICALEPCS 2011, Grenoble, France.

[4] A.Dworak et al., "The new CERN Controls Middleware", proc. of CHEP 2012, New-York, NY, USA.

[5] http://www.omg.org/spec/CORBA/3.3/

[6] http://zeromq.org/

[7] M.Arruat et al., "Front-End Software Architecture", proc. of ICALEPCS 2007, Knoxville, TN, USA.

[8] http://www.json.org/

[9] http://avro.apache.org

[10] http://www.ceph.com/

[11] http://www.gluster.org/

[12] http://hadoop.apache.org/

[13] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", proc. of OSDI 2004, San Francisco, CA, USA.

[14] S. Boychenko et al., "Toward a Second Generation Data Analysis Framework for LHC Transient Recording", WEPGF046, these proceedings, ICALEPCS'2015, Melbourne, Australia (2015).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

THE UNIFIED ANKA ARCHIVING SYSTEM – A POWERFUL WRAPPER TO SCADA SYSTEMS LIKE TANGO AND WINCC OA

D. Haas, S. Chilingaryan, A. Kopmann, D. Ressmann, W. Mexner, KIT, Karlsruhe, Germany

Abstract

ANKA realized a new unified archiving system for the typical synchrotron control systems by integrating their logging databases into the "Advanced Data Extraction Infrastructure" (ADEI). ANKA's control system environment is heterogeneous: some devices are integrated into the Tango archiving system, other sensors are logged by the Supervisory Control and Data Acquisition (SCADA) system WinCC OA. For both systems modules exist to configure the pool of sensors to be archived in the individual control system databases. ADEI has been developed to provide a unified data access layer for large time-series data sets. It supports internal data processing, caching, data aggregation and fast visualization in the web. Intelligent caching strategies ensure fast access even to huge data sets stored in the attached data sources like SQL databases. With its data abstraction layer the new ANKA archiving system is the foundation for automated monitoring while keeping the freedom to integrate nearly any control system flavor. The ANKA archiving system has been introduced successfully at three beamlines. It is operating stable since about one year and it is intended to extend it to the whole facility.

INTRODUCTION

ANKA is a third generation synchrotron light source operated by the Karlsruhe Institute of Technology (KIT). ANKA is operating seventeen beamlines and three more are under construction.

The control system of the ANKA beamlines is based on two independent components. The beamline instrumentation like the pressure, temperature or vacuum system is integrated in the SCADA system WinCC OA [1]. The experimental control, which includes movement of motors, triggering detectors et cetera is managed by Tango [2].

At ANKA the number of analogue sensors for long term archival per beamline can be estimated to 175 in WinCC OA plus 100 in Tango. Considering all 20 beamlines, these results to a number of approximately 5500 sensors, which have to be logged and archived at ANKA. Both systems have a separate solution to visualize the data for the user.

The main goal for the new unified ANKA archiving system is an easy and flexible configuration tool to select the data points for archival. And secondly to provide an unified fast graphical data browser [3]. Therefore this project splits in two parts. The first part was to interface the existing separate archiving solutions and the second part is to access all logged data of a beamline and the experiment. This data should be presented and retrieved in a modern, state-of-the-art web interface with fast data access. This offers the user a convenient way to analyse the data. ADEI [5], a web based interface for database query, developed by the Institute for Data Processing and Electronics (IPE) at KIT, fulfils exactly these requirements. Using ADEI as a viewer, respectively analysis tool and connecting it to the databases of the Tango archiving system and the WinCC OA Archiver plugin creates a platform to track and monitor the status of a beamline.

Due to the structure and available tools of this project, the main challenge was to combine and connect the three different parts, consisting of the Tango Archiving system, the WinCC OA History Database and ADEI.

INTEGRATION OF CONTROL SYSTEM HISTORY ARCHIVES

The integration of all separated archiving system in an overall distributed archiving system requires a TCP/IP based database interface of all sub systems, preferably SQL.

The Tango archiving system [4] allows logging of all Tango-attributes in a MySQL database. In 2014 this archiving system was already evaluated and implemented at ANKA to log the data of the experiment. The database of the SCADA part based on WinCC OA has no TCP/IP based SQL interface and is only accessible via a proprietary programming interface (API). Therefore ANKA developed a WinCC OA archiver plugin to record the logging data of preselected WinCC OA data points into a common SQL database. As the Tango Archiving System has been stored its logged data already into a MySQL database, we decided to use MySQL for the whole project. We found that the effort for the second option, to implement a new SQL interface to the existing C++ History API of WinCC OA, was too high.

The existing Tango Archiving System provides different advantages for logging the data points of the Tango control system. This system is a well-known and maintained archiving tool, which also includes with Mambo an easy to use graphical user interface.

Archiving the data-points of the SCADA system WinCC OA in a MySQL database required a new development. As the internal C like script language of WinCC OA is quite powerful, it was possible to use it to write all updated values directly to MySQL. For a convenient graphical user interface to configure archiving of the analogue values to be logged (Figure 2) a WinCC OA panel was developed. This GUI also allows different modes of logging. The data can be archived in different time periods or can be logged only if the value is changing.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

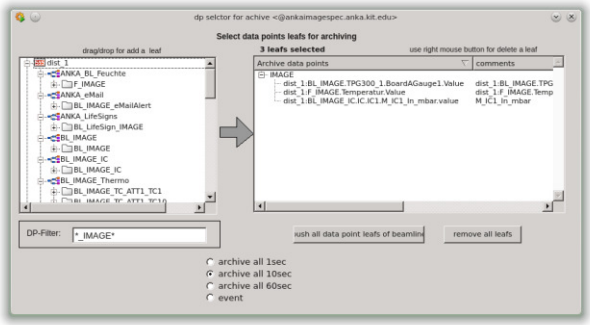


Figure 1: Screenshot of the Archiver plugin of WinCC OA. The sensors, can be added to the list of archived data-points by drag-and-drop.

ADEI

The "Advanced Data Extraction Infrastructure (ADEI)" has been developed to provide ad-hoc data exploration capabilities to a broad range of long-running scientific experiments dealing with time series data. Such experiments have varying characteristics and are often composed from multiple subsystems developed by different vendors. As a result, the underlying storage engines and the data formats often differ even between subsystems of a single experiment. On the other hand, the users want to get uniform access to all the data. Easy correlation of data produced by any components of the system is desirable. Beside this, operators need a tool providing the possibility to examine all collected data, checking the integrity and validity of measurements. The ADEI architecture shown in Figure 2 is modular. New data sources can easily be included. The back-end provides the desired uniform access to the data. The web-based front-end allows quick inspection of data archives. The communication between front-end and back-end is realized using the AJAX (Asynchronous JavaScript + XML) paradigm.

The back-end consists of multiple components organizing the data flow from the data source to the client application. The "Data Access Layer" hides details of the underlying data sources and provides other components of the system with uniform access to all types of data. This is realized using independent source drivers implementing ADEI data access interface. Furthermore, the data is passed through the chain of the configured data processing plugins which analyze the data, control the data quality, and optionally apply correction coefficients or filter out bad values. Hence, the rest of the system can fully rely on approved data quality.

ADEI is designed to deal with data sampled at high data rates and stored for long periods of time. It is still impossible to access large volumes of data interactively. Therefore, newly recorded data is continuously preprocessed. The data is aggregated over several predefined intervals, so called cache levels. For each cache level statistical information is calculated and stored in the caching database. The higher ADEI subsystem, then, can request the raw data, or to reduce the amount of received data, any statistical information from the selected cache level. To illustrate the concept, the rendering module can request the mean values from 60 seconds cache level effectively receiving minute averages instead of the raw data.

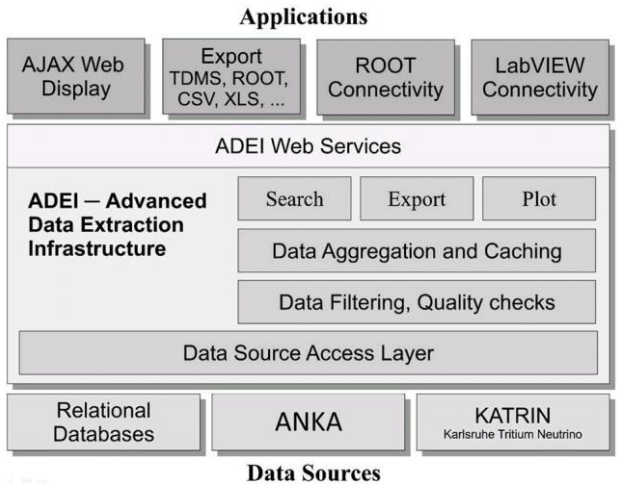


Figure 2: Architecture of the "Advanced Data Extraction Infrastructure ADEI". The "Data Source Access Layer" unifies the access to the time series data stored in different formats. After filtering and quality checks the data is aggregated and stored in intermediate caching databases. Access to the data is provided by the ADEI web display; web services are used to communicate with client applications.

ADEI provides several interfaces to access the data. The export module provides direct access to the data in a variety of formats implemented as plugins. The search module allows to quickly finding required data channels or time intervals where certain channels possessed the specified characteristics. The plot module is used to quickly generate preview charts. All modules are interfaced by a web service interface.

The ADEI front-end facilitates fast and intuitive (Google maps-style) navigation. Based on the caching subsystem described above, the users can request overview plots over long time intervals interactively. Since the quality checks were executed during the data preprocessing, the problematic intervals are highlighted in the overview charts. A user then can easily navigate to the interval of interest and zoom-in using the mouse only. The complete plot-generation time does not normally exceed 500 ms for any type of requested data.

The ADEI web service interface allows platform and programming language independent design of application-specific data management solutions. To bring an example, we have developed a WebGL application that shows the temperature distribution on the provided 3D model based on the specified mappings of the ADEI data channels onto the model. Also, we developed software libraries to use ADEI data within the National

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Instruments LabVIEW environment and CERN's ROOT data analysis suite.

THE ANKA ARCHIVING SYSTEM

It is essential that the ANKA Archiving System is a stable, reliable and convenient overall system for archiving analogue data-points of a complex control system. To install and configure the different tools a certain hardware structure is needed. Figure 3 shows the efficient modular hierarchy. At every beamline there is a dedicated small server, which is responsible to archive the data of this certain beamline into a database. All these archiving servers are connected to one server where the web front-end ADEI is running. It was sufficient to build the servers from standard PC-parts, which make it cost effective. For the archiving servers standard Intel Xeon 8 core boards with 8 GB RAM and 3 TB disk space are used. The ADEI server differs in the disk space with 12 TB and a memory of 32 GB.

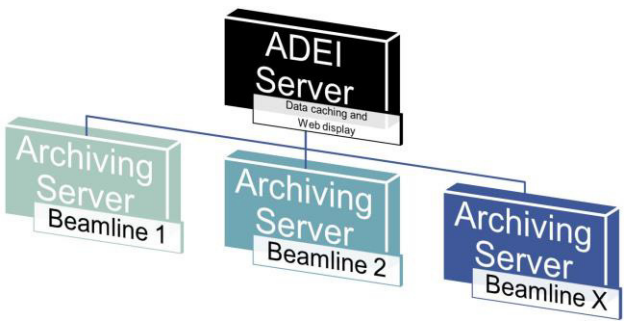


Figure 3: Structure of the different servers and their function in the ANKA Archiving System.

To connect the different software layers a well-structured concept was introduced.

As described in the previous chapters the user has two graphical user interfaces where he can configure the logging data-points of the two main control systems (see Figure 4). To configure the archiving data for Tango the user can use the GUI Mambo and for the SCADA system WinCC OA there's the developed Archiver Plugin available. Both of these GUIs are saving data into independent MySQL databases. Finally these two databases are connected to an ADEI instance which is pre-processing and caching the data and providing the web-interface to the user.

This software concept can be introduced at every beamline to provide the unified ANKA Archiving System. The complete ANKA archiving system is currently been embedded and implemented into the control system at three beamlines, Topo-Tomo [6,7], Image and NANO.

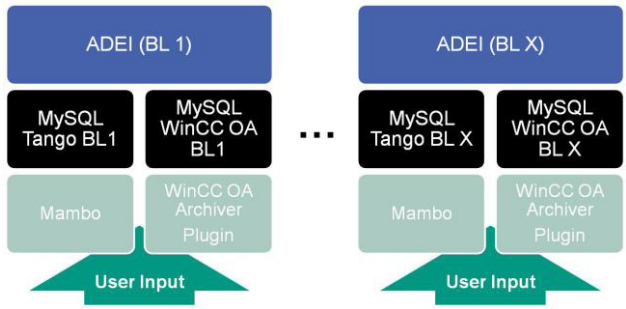


Figure 4: The different software layers of the ANKA Archiving system. The user configures the logged data, which is afterwards saved in a MySQL database and finally presented by ADEI.

THE WEB INTERFACE

The web interface of ADEI (Figure 5) is intuitive and self-describing. There is a list of data sources existing where you can choose the beamline from which you want to retrieve the archived data. The available logged data is shown in a tree structure. In this tree the user can easily select the data to plot and evaluate. The plot is shown in a big window where a zoom-in, save or export function is available. As export function mainly the excel- and cvs-file export is supported.

CONCLUSION & OUTLOOK

The unified ANKA archiving system was implemented and tested at three beamlines. It is a powerful combination of different interconnected tools providing a convenient, state-of-the-art and user friendly way to log, archive and represent data of a synchrotron beamline. The system is easy to setup and does not required deep programming knowledge. The test system is in operation since nearly one year and turned out to be stable and reliable. It can be easily extend by further history database sources. The next steps will be to extend our system to mobile devices like smartphones and tablets and to roll out the unified archiving system to the remaining ANKA beamlines.

The sources will be available under GNU GPL2 on the official Tango website www.tango-controls.org.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

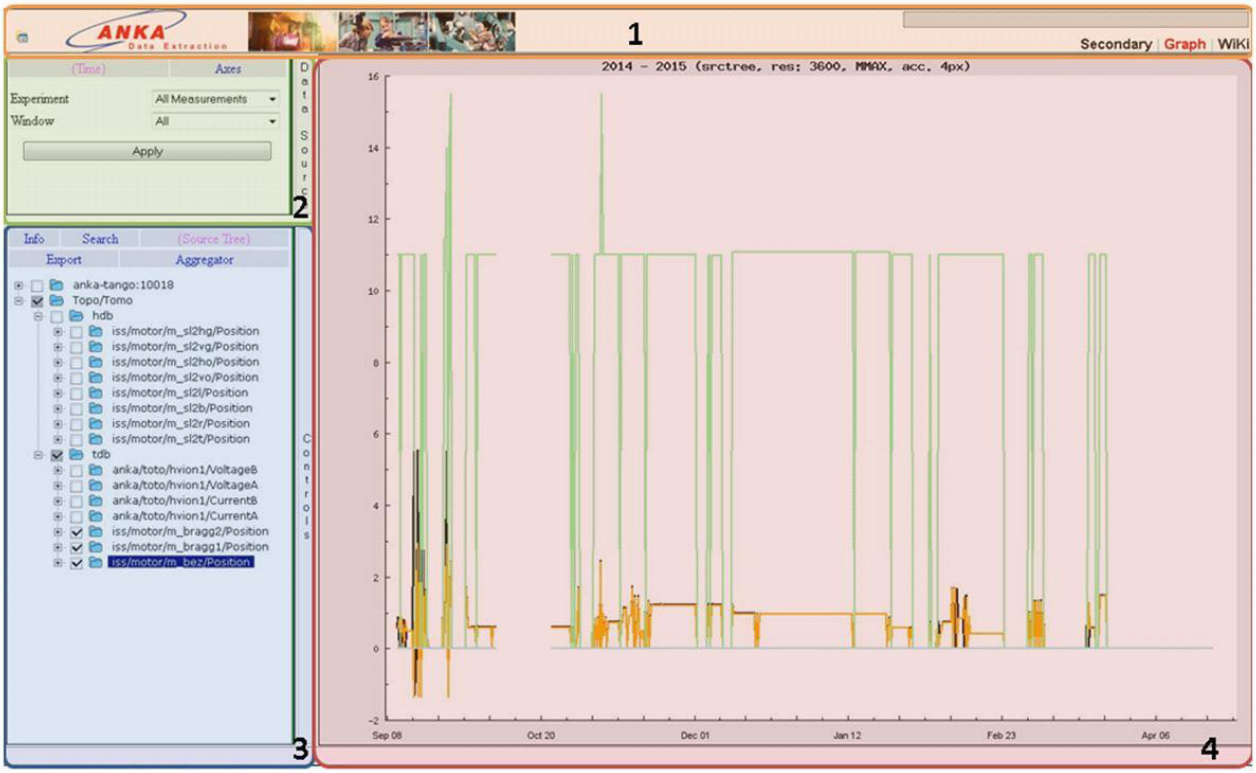


Figure 5: Screenshot of ADEI Web Front-end for the ANKA Tango Archiving System. (1) header bar with main menu, (2) dropdown-menu for selecting server respectively beamline, (3) data selection, (4) plot of the selected data.

REFERENCES

[1] WinCC OA, http://www.etm.at

[2] J.-M. Chaize, A. Goetz, W.-D. Klotz, J. Meyer, M. Perez, E. Taurel, "TANGO – an object oriented control system based on corba", *International Conference on Accelerator and Large Experimental Physics Control Systems*, (1999): p. 475 – 479.

[3] D. Haas, S. Chilingaryan, A. Kopmann, D. Ressman, W. Mexner, "ADEI and Tango Archiving system – a convenient way to archive and represent data", *International Workshop on Personal Computers and Particle Accelerator Controls*, (2014): Proceedings.

[4] Tango archiving system, http://www.tango-controls.org/tools/archiving-system-2/archivingsystem

[5] S. Chilingaryan, A. Beglarian, A. Kopmann, S. Voecking, "Advanced data extraction infrastructure: Web based system for management of time series data," *Journal of Physics: Conference Series*, 219 (2010), Part 4, 10 pages

[6] A. Rack, T. Weitkamp, S. Bauer Trabelsi, P. Modregger, A. Cecilia, T. dos Santos Rolo, T. Rack, D. Haas, R. Simon, R. Heldele, et al., "The micro-imaging station of the topotomo beamline at the ANKA synchrotron light source", *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 267(2011): p. 1978 - 2009.

[7] P. Vagovic, T. Farago, T. dos Santos Rolo, S. Chilingaryan, T. Müller, A. Cecilia, T. van de Kamp, E. Hamann, A. Riedel, D. Haas, W. Mexner, M. Fiederle, T. Baumbach, "Recent upgrades at the micro-tomography station at TopoTomo Beamline at ANKA light source", *will be published in Journal of Synchrotron Radiation.*

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INTEGRATED DETECTOR CONTROL AND CALIBRATION PROCESSING AT THE EUROPEAN XFEL

A. Münnich, S. Hauf, B.C. Heisen, F. Januschek, M. Kuster, P.M. Lang, N. Raab,
T. Rüter, J. Sztuk-Dambietz, M. Turcato
European XFEL GmbH, Albert-Einstein-Ring 19, 22761 Hamburg, Germany

Abstract

The European X-ray Free Electron Laser is a high-intensity X-ray light source currently being constructed in the area of Hamburg, that will provide spatially coherent X-rays in the energy range between 0.25 keV and 25 keV. The machine will deliver 10 trains/s, consisting of up to 2700 pulses, with a 4.5 MHz repetition rate. The LPD, DSSC and AGIPD detectors are being developed to provide high dynamic-range Mpixel imaging capabilities at the mentioned repetition rates. A consequence of these detector characteristics is that they generate raw data volumes of up to 15 Gbyte/s. In addition the detector's on-sensor memory-cell and multi-/non-linear gain architectures pose unique challenges in data correction and calibration, requiring online access to operating conditions and control settings. We present how these challenges are addressed within XFEL's control and analysis framework Karabo, which integrates access to hardware conditions, acquisition settings (also using macros) and distributed computing. Implementation of control and calibration software is mainly in Python, using self-optimizing (py) CUDA code, numpy and iPython parallels to achieve near-real time performance for calibration application.

INTRODUCTION

The European X-ray Free Electron Laser (XFEL.EU) is an international research facility currently under construction in the area of Hamburg, Germany, which will start operation at the end of 2016 [1]. The superconducting linear accelerator of the facility will deliver electron bunches with an energy of up to 17.5 GeV, arranged in trains of typically 2700 bunches at a repetition rate of 4.5 MHz. Each train will be followed by a gap of 99.4 ms, during which data is read out from the detector front-end. The X-ray pulses will be particularly intense and ultra-short, down to a pulse length of less than 100 fs with a peak brilliance of 10^{33} photons/s/mm^2/mrad^2 per 0.1 % bandwidth [2].

The high repetition rate and the high peak brilliance pose unique challenges in terms of performance and radiation tolerance on the X-ray detector systems employed at the facility. As a result, each scientific instrument uses a detector optimized for its specific needs in terms of energy range, resolution, operation conditions, etc. Three large 2D area imaging detectors are being developed: the Adaptive Gain Integrating Pixel Detector AGIPD [3], DePFET Sensor with Signal Compression DSSC [4, 5] and a Large Pixel Detector LPD [6], which can run at the high repetition rate of 4.5 MHz. Smaller detectors like pnCCDs [7, 8] and Fast CCDs [9, 10] can be operated at 10 Hz. In addition, detector technologies such as Silicon Drift Detectors (SDDs), Micro-Channel Plates (MCPs) or silicon strip detectors [11] are being evaluated for specialized applications.

The large data volumes produced by the MHz-rate 2D detectors (10 to 15 GB per second per detector) and the huge number of calibration parameters (of the order of 10^9) require new concepts in data handling, calibration and processing in an integrated manner, aspects of which we will present.

THE KARABO DETECTOR CONTROL SOFTWARE FRAMEWORK

The software framework Karabo [12] will be used to control and manage the photon beamlines at the European XFEL as well as components within the experimental hutches including the detectors. Karabo's generic functionality covers all aspects from data acquisition (DAQ) over equipment control to data analysis. Figure 1 illustrates the different component categories available in the framework and their interaction via a JMS message broker or direct peer-to-peer (p2p) interfaces. In the following we present examples of the usage in the aforementioned categories: the control of a pnCCD-detector setup including the associated vacuum systems employing device composition, the control, DAQ and data-processing of an SDD-detector as well as pipeline-data analysis in the context of detector data correction and calibration.

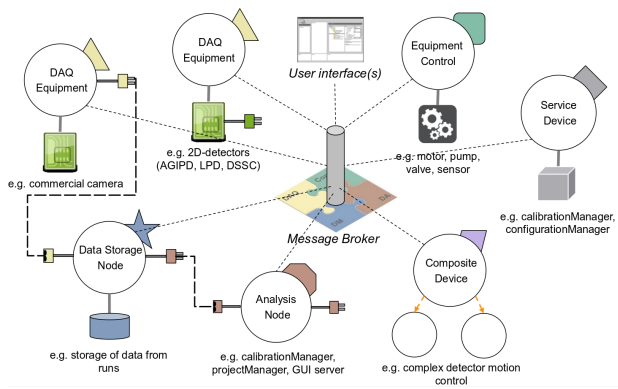


Figure 1: Schematic overview of Karabo's design concept. Functionality is implemented into the framework as devices (either `C++` or `Python3`), which can communicate either via a JMS message broker (shown here as monolithic, but in fact supporting dynamic fail-over and clustering) or p2p interfaces.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Control and Composition – the pnCCD Setup

The pnCCD detector operates in vacuum and at low temperatures, in order to be able to image X-rays at energies down to 50 eV. For system safety venting has to occur in a controlled fashion and it has to be insured that the device is warmed prior to doing so. Figure 2 shows the control GUI for the vacuum and cooling system of the pnCCD setup which consists of several valves, pumps and a chiller. The status of each component is indicated by its color. User interaction is realized using buttons and relevant information about temperature and pressure is shown. The buttons shown next

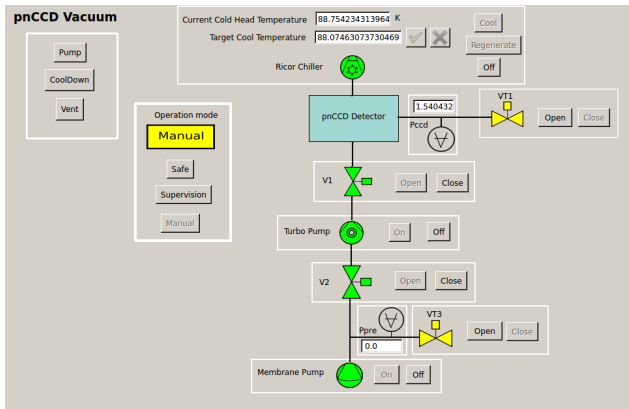


Figure 2: The control GUI for the vacuum and cooling system of the pnCCD setup, laid out as a functional diagram.

to components in the GUI are meant for manual interaction, i.e. when they are controlled through the respective Karabo device individually. More complex procedures like pumping to nominal operating conditions, venting or cooling of the system can be implemented either as macros or composite devices on top of this manual control. Figure 3 shows the UML activity diagram, which describes the program flow the ”pump” button starts. Several devices are involved relying on conditions set from values from Karabo-controlled pressure and temperature sensors. Such diagrams are created for each procedure and document program states, error conditions and flow.

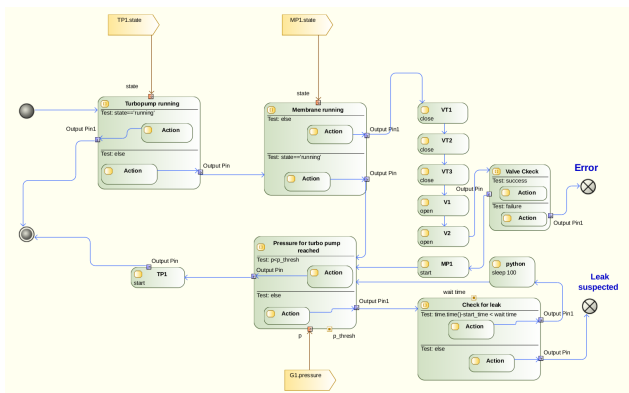


Figure 3: UML activity diagram of the pumping procedure for the pnCCD setup.

Detector Control, DAQ and Data Processing of an SDD Detector

Silicon drift detectors (SDDs) are 1D detectors used at XFEL.EU as reference detectors for characterization of X-ray sources and for cross-calibration with 2D detectors. The control and readout of an Amptek Fast-SDD detector [13], via Ethernet, has been integrated into Karabo as a device. Through Karabo macros this allows the SDD to be used in automated procedures: e.g taking data at intervals determined by a positional scan using linear stages. Figure 4 shows the corresponding Karabo GUI; in addition to monitoring and control the acquired spectra can be visualized, written into an HDF5 file or streamed via a p2p connection to be processed by another device. As currently implemented, such a ”compute” device monitors this output channel, calibrates the incoming data and finds peaks in the spectra in order to identify chemical elements by their fluorescence emissions.

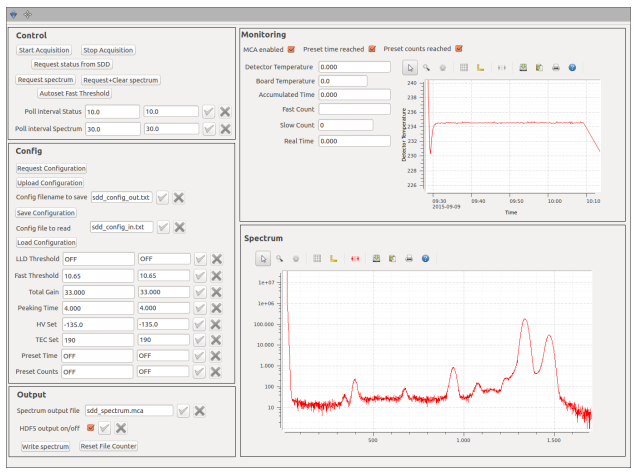


Figure 4: The Karabo GUI for controlling and monitoring an AmpTek Fast SDD detector.

Python-based Calibration and Data Analysis Suite

European XFEL facility users will not interact with raw detector data, which is the facilities main archival product. Instead, calibrated data sets will be the main user-accessible data product, and calibration and correction will be facility-provided [14]. The data-flow from detector acquisition to end-user availability is illustrated in Fig. 5.

Calibration of the MHz-rate 2D detectors poses a challenge due to the large number of calibration constants. For the non-linear gain calibration of the 1 Mpixel DSSC detector [15] these for example amount to about 10^9, with similar numbers for LPD and AGIPD where per-pixel, per-gain and per-memory cell calibration is required. The diverse range of experiments, where some have single photon events with low event rates and others densely filled images mandate different detector operation modes, which may need (partially) different calibration data.

In order to manage calibration-associated information, which may be produced at intervals of per-run to a few

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

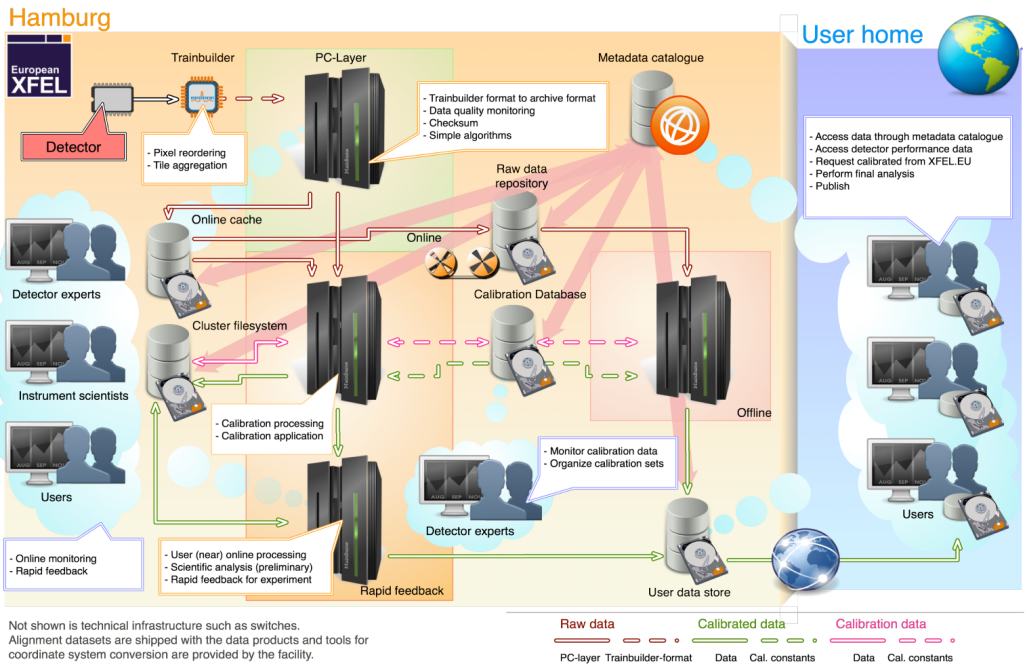


Figure 5: The data processing chain at the European XFEL. Raw data from the detectors (red arrows) is persisted and (selectively) forwarded via Karabo p2p interface to online calibration. Calibrated data (green arrows) is produced either from the online stream or by processing offline data. Calibration data (pink arrows) is processed from the online cache and inserted into the calibration database. In all cases facility users will only access calibrated datasets.

months, a MySQL database has been developed, which interacts with dedicated python devices through RESTful interfaces. It is able to resolve calibration constants by detector operating condition, maintain a set of current calibration data and associated validity periods thereof, as well as keep track of detector alignment information.

The production and application of the calibration and correction constants managed by this database may occur as part of two general scenarios: interactive data analysis and automated, pipeline processing, the latter ideally with near-real-time performance. A python library which provides a common codebase for both applications is being developed. It can be used as part of an iPython (notebook) based interactive analysis [16] or automated in a distributed Karabo system communicating via p2p links. Underlying, numpy [17] and scipy [18] are used for calculations on the CPU, ipcluster for concurrent processing and pyCuda [19] for GPU-accelerated computing. Further optimization using Numba JIT-technology [20] are being investigated and near-real time performance for calibration application has been demonstrated [21].

Figure 6 shows an example of pipelined processing within Karabo. Data acquired by an LPD prototype has been offset- and common mode-corrected on the GPU. In the figure the raw and corrected data are contrasted, making apparent that proper calibration is required to reveal the full diffraction patterns produced by the ferrous solution sampled with monochromatic 18 keV X-rays at the ESRF facility. The required calibration and correction constants were retrieved from the calibration database.

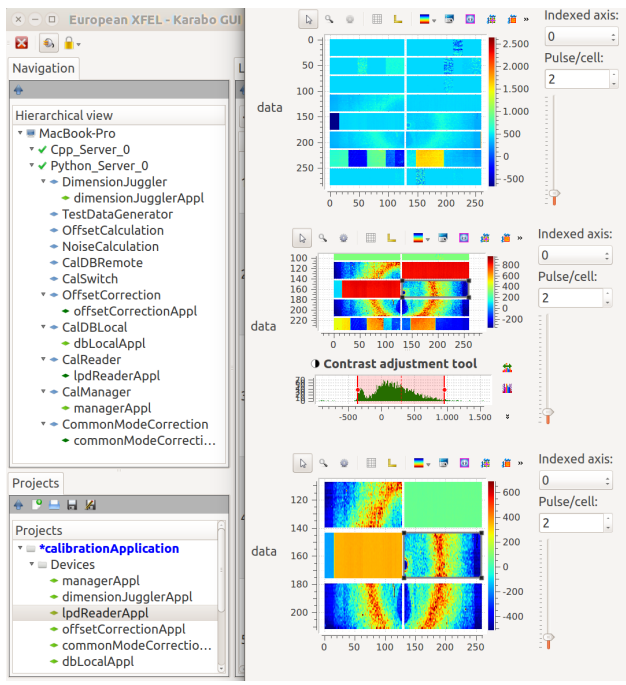


Figure 6: Example of calibration processing within Karabo. On the left the device instances interacting via p2p interfaces are visible. The superimposed panel shows preview images of uncalibrated (top), offset-corrected (center) and offset- and common mode-corrected data showing diffraction by a ferrous solution images by an LPD prototype.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONCLUSION

We have presented examples of the integrated control and processing within the Karabo software framework, needed to make optimal use of the capabilities of European XFEL detector systems. The examples range from control and monitoring tasks, an integrated DAQ and control system for an SDD detector, to high-performance calibration processing and associated interaction with a calibration database.

ACKNOWLEDGMENT

The authors acknowledge the support of the Control and Scientific computing (CAS), Advanced Electronics (AE) and IT and Datamanagement (ITDM) groups at XFEL.EU.

REFERENCES

[1] M. Altarelli, R. Brinkmann, M. Chergui, W. Decking, B. Dobson, S. Düsterer, G. Grübel, W. Graeff, H. Graafsma, J. Hajdu, *et al.*, "The european x-ray free-electron laser," *Technical Design Report, DESY*, vol. 97, pp. 1–26, 2006.

[2] E. Schneidmiller and M. Yurkov, *Photon beam properties at the European XFEL*. DESY, 2011.

[3] B. Henrich, J. Becker, R. Dinapoli, P. Goettlicher, H. Graafsma, H. Hirsemann, R. Klanner, H. Krueger, R. Mazzocco, and A. Mozzanica, "The adaptive gain integrating pixel detector agipd a detector for the european xfel," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 633, pp. S11–S14, 2011.

[4] M. Porro, L. Andricek, L. Bombelli, G. De Vita, C. Fiorini, P. Fischer, K. Hansen, P. Lechner, G. Lutz, and L. Strüder, "Expected performance of the depfet sensor with signal compression: A large format x-ray imager with mega-frame readout capability for the european xfel," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 624, no. 2, pp. 509–519, 2010.

[5] M. Porro, L. Andricek, S. Aschauer, M. Bayer, J. Becker, L. Bombelli, A. Castoldi, G. De Vita, I. Diehl, and F. Erdinger, "Development of the depfet sensor with signal compression: A large format x-ray imager with mega-frame readout capability for the european xfel," *Nuclear Science, IEEE Transactions on*, vol. 59, no. 6, pp. 3339–3351, 2012.

[6] M. Hart, C. Angelsen, S. Burge, J. Coughlan, R. Halsall, A. Koch, M. Kuster, T. Nicholls, M. Prydderch, and P. Seller, "Development of the lpd, a high dynamic range pixel detector for the european xfel," in *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE*, pp. 534–537, IEEE, 2012.

[7] N. Meidinger, R. Andritschke, R. Hartmann, S. Herrmann, P. Holl, G. Lutz, and L. Strüder, "pnccd for photon detection from near-infrared to x-rays," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 565, no. 1, pp. 251–257, 2006.

[8] S. Send, A. Abboud, R. Hartmann, M. Huth, W. Leitenberger, N. Pashniak, J. Schmidt, L. Strüder, and U. Pietsch, "Characterization of a pnccd for applications with synchrotron radiation," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 711, pp. 132–142, 2013.

[9] P. Denes, D. Doering, H. Padmore, J.-P. Walder, and J. Weizeorick, "A fast, direct x-ray detection charge-coupled device," *Review of Scientific Instruments*, vol. 80, no. 8, p. 083302, 2009.

[10] D. Doering, Y.-D. Chuang, N. Andresen, K. Chow, D. Contarato, C. Cummings, E. Domning, J. Joseph, J. Pepper, and B. Smith, "Development of a compact fast ccd camera and resonant soft x-ray scattering endstation for time-resolved pump-probe experiments," *Review of Scientific Instruments*, vol. 82, no. 7, p. 073303, 2011.

[11] A. Mozzanica, A. Bergamaschi, R. Dinapoli, H. Graafsma, B. Henrich, P. Kraft, I. Johnson, M. Lohmann, B. Schmitt, and X. Shi, "A single photon resolution integrating chip for microstrip detectors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 633, pp. S29–S32, 2011.

[12] B. Heisen, D. Boukhelef, S. Esenov, S. Hauf, I. Kozlova, L. Maia, A. Parenti, and J. Szuba, "Karabo: an integrated software framework combining control, data management and scientific computing tasks," *FRCOAAB02, ICALEPCS-2013*, 2013.

[13] Amptek, "AmpTek: Fast SDD," 2015–. [Online; accessed 2015-10-07].

[14] M. Kuster, D. Boukhelef, M. Donato, J.-S. Dambietz, S. Hauf, L. Maia, N. Raab, J. Szuba, M. Turcato, K. Wrona, *et al.*, "Detectors and calibration concept for the european xfel," *Synchrotron Radiation News*, vol. 27, no. 4, pp. 35–38, 2014.

[15] G. Weidenspointner *et al.*, "Calibration of the non-linear system response of a prototype set-up of the dssc detector for the european xfel," in *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, 2012.

[16] F. Pérez and B. E. Granger, "IPython: a system for interactive scientific computing," *Computing in Science and Engineering*, vol. 9, pp. 21–29, May 2007.

[17] S. van der Walt and G. Colbert, S.C. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, pp. 22–30, 2011.

[18] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online; accessed 2015-05-18].

[19] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, and A. Fasih, "PyCUDA and PyOpenCL: A Scripting-Based Approach to GPU Run-Time Code Generation," *Parallel Computing*, vol. 38, no. 3, pp. 157–174, 2012.

[20] T. Oliphant, "Numba python bytecode to llvm translator," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2012.

[21] S. Hauf *et al.*, "Calibration and calibration data processing concepts at the european xfel," in *IEEE Nuclear Science Symposium & Medical Imaging Conference*, no. PUBDB-2015-01571, The European X-Ray Laser Project XFEL, 2014.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

DEVELOPMENT OF THE J-PARC TIME-SERIES DATA ARCHIVER USING A DISTRIBUTED DATABASE SYSTEM, II

N. Kikuzawa#, H. Ikeda, Y. Kato J-PARC, Tokai-mura, Naka-gun, Ibaraki, Japan
A. Yoshii, NS Solutions Corporation, Shinkawa, Chuo-ku, Tokyo, Japan

Abstract

J-PARC (Japan Proton Accelerator Research Complex) consists of Linac, 3 GeV rapid cycling synchrotron ring (RCS) and Main Ring (MR). In the Linac and the RCS, data of about 64,000 EPICS records have been acquired for control of these equipment. The data volume is about 2 TB every year, and the total data volume stored has reached over 12 TB. The data have been being stored by a Relational Database (RDB) system using PostgreSQL since 2006 in PostgreSQL, but it is becoming that PostgreSQL is not enough in availability, performance, and flexibility for our increasing data volume. In order to deal with these problems, we proposed a next-generation archive system using Apache Hadoop, a distributed processing framework, and Apache HBase, a distributed database.

In this paper we are reporting that we have re-designed and re-constructed the cluster with resolving some issues, including enhancing hardware of master nodes, creating scripts to automatically construct nodes, and introducing monitoring tools for nodes. Having adjusted the configurations of HBase/Hadoop and measured the performance of our new system, we are also reporting its results and considerations.

INTRODUCTION

J-PARC is controlled with a lot of equipment, and we have been archiving a time series of operation data provided from about 64,000 EPICS records for the Linac and the RCS since 2006 [1]. PostgreSQL has been used in the present data archiving system, but it has some problems of capacity, extensibility, and data migration. In order to deal with these problems, we proposed a next-generation archive system [2][3] using Apache Hadoop [4], a distributed processing framework, and Apache HBase [5], a distributed database.

In the previous paper we reported that we updated the versions of HBase/Hadoop composing our test system [6], to conquer a single point of failure by making redundant a master node, and we showed issues to fix our tools in the new system. Having adjusted the configurations of HBase/Hadoop and measured the performance of our new system, we are also reporting its results and considerations.

RECONSTRUCTION OF THE CLUSTER

Replacement Master Node

We have updated Hadoop to the version 2.2.0 now. Hadoop 2.x provides a hot standby NameNode, which can take over the state that the previous active NameNode has provided, with no downtime. At least three master nodes are needed for this function to deploy ZooKeeper [7] and JournalNode. ZooKeeper is a high-performance coordination service for distributed applications, and both Hadoop and HBase depend on. JournalNode is one of the components of Hadoop. ZooKeeper and JournalNode are based on a majority decision among nodes, and it is meaningful to deploy them on an odd number of machines. We replaced the master node 2 and 3. The layout of the system is illustrated in Figure 1 and the spec of our system is listed in Table 1.

When the master nodes has been replaced, processes of Hadoop has been relocated, and memory allocation has been reconsidered. The allocation of memory shows in Table 2.

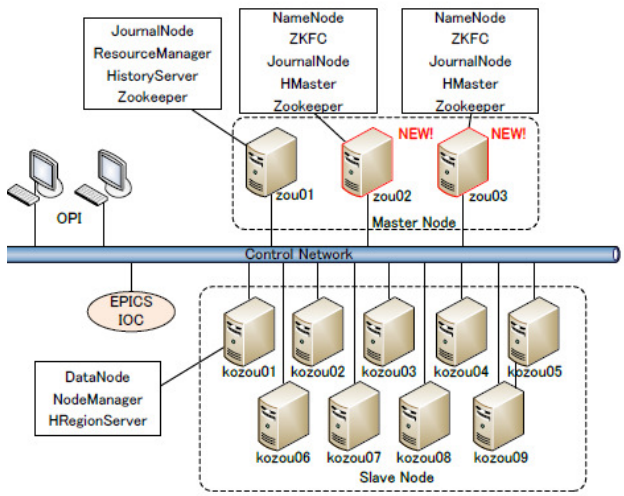


Figure 1: The archiving system configuration.

Table 1: Spec of Hadoop and HBase System

Master Node 1	DELL PowerEdge R610 CPU: Intel Xeon E5620 (4Core, 2.4GHz) MEM: 24GB HDD: 600GB x4 (RAID10)
Master Node 2, 3	DELL PowerEdge R320 CPU: Intel Xeon E5-1410v2 (4Core, 2.8GHz) MEM: 24GB HDD: 600GB x4 (RAID10)
Slave Node	DELL PowerEdge R410 CPU: Intel Xeon E5620 (4Core, 2.4GHz) MEM: 24GB HDD: 2TB x4
Software	OS: CentOS6.6 Hadoop: 2.2.0 HBase: 0.96..1.1 ZooKeeper: 3.4.5

#kikuzawa.nobuhiro@jaea.go.jp

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Table 2: Memory Allocation (GB)

Master Node	1	2,3	Slave Node	
ZooKeeper	1	1		
Name Node		8	Data Node	1
ZKFC		1		
Journal Node	1	1		
Resource Manager	1		Node Manager	1
History Server	1		YARN container	6
HBase Master		4	Region Server	12
total	4	15	total	20

RAID and Mount Option

It is common sense to use non-RAID for slave nodes in Hadoop. The redundancy of data in Hadoop is made by holding the same data in several slave nodes, and RAID cannot be an alternative for the Hadoop's redundancy because RAID just makes HDD redundant, not for the whole node. In place of the hot-swap function we can use automatic scripts to construct from scratch. It is also said that, a Hadoop cluster constructed with non-RAID is faster than a one with RAID-0 because RAID-0 makes physical disks keep pace with the slowest one of them.

In the previous presentation, we constructed the slave nodes with RAID-5 with 4 physical disks. We have decided to follow the Hadoop common sense and have changed RAID-5 to non-RAID. To be exact, our RAID controller cannot handle non-RAID, and we have constructed RAID-0 for each one disk. Ironically, that gives us an advantage that we can still use battery backed-up memory in the RAID controller.

We use ext4 for the file system on the OS, which ext4 is a standard format in Linux these days. As to mount options, although Hadoop makes data redundant, we have chosen options to ensure persistence of data for a rainy day, for example, in the case that all nodes lose power supply all at once, with the exception that we invalidate I/O barrier because of battery backed-up memory in the RAID controller, as described above. We use *ordered* for the journal mode in order to shut out the possibility to see invalid data when it crashes, which setting is enough because Hadoop doesn't support random access and consequently it doesn't overwrite existing data. Our versions of ZooKeeper, Hadoop and HBase support Java6, which means *dirsync* is required because Java6 doesn't support fsync on a directory. To tell the truth, we overlooked that the slave nodes in Hadoop don't invoke fsync unless we set the Hadoop configuration property *dfs.datanode.synconclose* to be true, which property is hidden from the document. The performance test described later was done before we found it out.

Attribute of HBase

Data compression is a presupposition of HBase. There are two compression layers. The one is Data Block Encoding, which has been introduced since HBase 0.94, and it compresses sequential records by just storing the difference between records. The other is generally used compression algorithm like Snappy, which is applied after Data Block Encoding. Data Block Encoding is effective especially for the keys are relatively much larger than the corresponding values, and that compression matches our table design. We are showing what combination of Data Block Encoding and compression algorithm is best, in the performance test described later.

HBase can apply a bloom filter to search for data. HBase should search several files, and applying a bloom filter beforehand narrows down the files, at the cost of a little increasing the resource usage. This setting is a table attribute, and by default it is enabled for record keys. Bloom filters are not applied to extract records with a range condition while such a way is the only one we assume to search records, and we have decided to invalidate a bloom filter and avoid wasting its resource usage.

HBase prepares Block Cache for millisecond order latency. When Block Cache is enabled, data blocks once read from a disk are cached in memory. The cache itself is shared per region server, while it is per table whether Block Cache is used or not for the table, and this setting is a table attribute. As to our purpose, we assume to always get much data at once, and such a large data would be cached in client-side. The clients are not so many, and the data is rarely requested again. And moreover, by default, Block Cache can use 40% of the maximum heap and it might reduce the performance by consuming a lot of heap with frequently triggering GC. Because of these reasons we have decided to invalidate Block Cache.

Cluster Management

In order to monitoring servers we use Nagios, which is a mature tool and widely used in Linux servers. In Nagios anomaly detectors are introduced as plugins, and Nagios itself manages to schedule to trigger the detectors and notify users by e-mail if any.

Nagios is designed to use e-mail for notification, but our cluster is on the LAN which is basically separated from the outer network, and sending an e-mail to the mail server on the outer network is blocked off. For now we just explicitly login the node the Nagios is deployed on, and check by parsing the file that Nagios generates at fixed intervals for a web application, or run an automatic script to do the same things.

Some standard Nagios plugins, including pinging to servers, are formally distributed as a set. For other plugins you should write a script or something by yourself, or download from trusted web sites. We use two such convenient external plugins, check_ipmi_sensor and check_openmanage. Both is for collecting remote hardware status, but check_ipmi_sensor does via IPMI using a tool freeipmi, which comes from the installation disks of Linux OS, and check_openmanage does via SNMP using Dell OpenManage. They are similar but each has its merits and demerits. In order to monitor modules in ZooKeeper, Hadoop and HBase, we have created scripts as Nagios plugins, using the command *jps* in JDK for checking existence of Java processes, and invoking a query via socket communication if any. They just emulate the

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

procedures to check the modules by hand, and we would easily grasp what happens.

In addition, we have installed Ganglia [8], which was also installed in the old cluster. Ganglia collects numerical data, which differs from Nagios, and Ganglia would be suitable to detect unusual CPU or resources usages, or to tune the performance. Ganglia has an attractive feature of using multicast so that a lot of monitoring targets don't inflict a heavy load on the network.

Automatic Scripts to Construct Nodes

We have created the kickstat file to install and configure OS and required software packages coming from the installation disks, and have created a set of installation files to install and configure the rest of the required packages, in line with the views we have described above.

It would be preferable to construct a node automatically from start to end, but we should manually set up SSH keys or security concerns, and set up configurations to get hardware status via IPMI or about local environment dependent concerns.

PERFORMANCE TESTS

We measured the performance of registration and acquisition of records based on the assumption of actual usage in J-PARC, which is not intended for the general situation. We measured both the old cluster and the reconstructed new cluster.

Conditions

For the measurement of registration of records, we assume that there would be 10,000 EPICS records and we should regularly get a record at one second intervals from each EPICS record. We put records for one day into our cluster without stops and we measure its consumption time, and we repeat the same things 5 times in a row. Because each record is very small, in actual usage in J-PARC we should use a buffer to send many records together. According to this assumption we invalidate auto-flush and explicitly invoke flush at the end, and we count the consumption time till the flush is complete. We create random names as EPICS records, which is 30 characters randomly selected from 52 uppercase and lowercase alphabetical letters. We generate data from random double precision values, selected from 0.0 (inclusive) to 1.0 (exclusive). Splitting regions in advance is generally preferable for HBase, although it is not trivial in our actual usage, and we expect that the balancer wakes up regularity and automatically splits and moves regions between nodes with distributing loads. As to the performance measurement, because we execute it in a short period and we should not expect the balancer, we pre-split regions and distribute loads from the beginning. To put it concretely, we assume that the randomly generated names are equally distributed in the name space, and equally divide the space by the first character of the names into regions whose number is same as that of our region servers.

For the measurement of acquisition of records, we make use of the environment after the measurement of registration. Assuming that there are 5 clients simultaneously connecting to the cluster, we use 5 threads and make each thread retrieve records for one random day for 10 random channels, and we measure its consumption time for each thread. HBase would not be designed for such acquisition; HBase gives priority to low latency, which is against to Hadoop which gives priority to throughput. Moreover, for data acquisition many clients' random accessing is suitable for HBase because it well distributes loads to multiple regions servers, improving its overall performance advantage. Despite these features, we are still interested in the performance of a few clients accessing large sequential data, along the assumption of actual usage in J-PARC.

As to buffers in client-side, we use the default size of the write buffer, which is 2MB, and we use 20,000 records for the read cache, which corresponds to 2MB if we count 100 bytes for one record.

As described before, we examine what combination of Data Encoding Block and compression algorithm is best for our table design. To put it concretely, we examine {none(non-use), *diff*, *fast_diff*} for Data Encoding Block, and examine {none(non-use), GZ, Snappy, LZ4} for the compression algorithm.

Results

The results of the performance measurement are shown in the following tables and figure. Table 3 shows the consumption time to register data for one day, averaging the times measured repeatedly 5 times in a row. Figure 2 shows its transition instead of averaging in the case of the compression combination none-none. Table 4 shows the consumption time to retrieve data, averaging the times for threads. Table 5 shows the total amounts of data written in the Hadoop distributed file system (HDFS), which are measured in the new cluster but theoretically they rather depend on the combination of Data Block Encoding and compression algorithm.

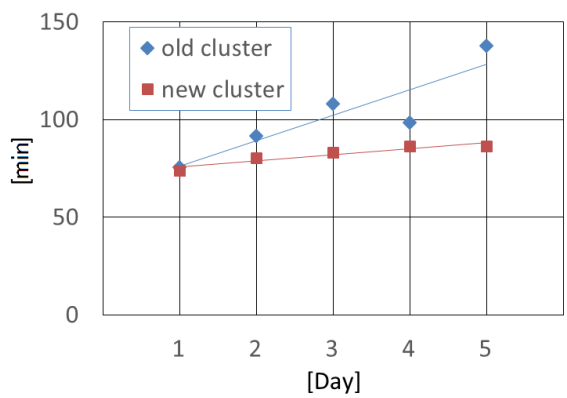


Figure 2: Write time in the none-none Case (min).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Table 3: Write Test (min)

	none	GZ	Snappy	LZ4
old cluster				
none	102.3	59.0	64.3	64.8
diff	57.4	54.0	58.9	59.4
fast_diff	56.7	55.9	58.5	58.8
new cluster				
none	81.9	77.6	74.9	80.8
diff	67.6	68.6	75.4	87.3
fast_diff	69.0	71.7	76.4	71.4

Table 4: Read Test (sec)

	none	GZ	Snappy	LZ4
old cluster				
none	7.3	8.3	8.8	7.1
diff	8.9	10.0	11.0	9.4
fast_diff	7.5	8.9	8.2	9.4
new cluster				
none	7.2	8.2	8.0	8.5
diff	9.8	8.5	10.0	11.3
fast_diff	8.6	8.6	9.0	8.3

Table 5: Sum of Stored Files (GB)

	none	GZ	Snappy	LZ4
none	286.2	64.1	80.5	76.7
Diff	53.1	47.3	52.9	53.1
fast_diff	53.8	48.7	53.8	53.9

Consideration

For the measurement of registration of records, the old cluster writes data 20-40% faster except the case of none-none, where the total amount of data written in HDFS is extremely larger than the other cases and the old cluster degrades its performance down according as written data in HDFS is increased. That indicates, RAID-5 supported by a RAID controller is superior on a lower load, and parallel accessing physical disks is superior on a higher load. We should have used much larger amount of data to point out specifically the advantage of the new cluster.

As to the combination of Data Block Encoding and compression algorithm, *diff* and *fast_diff* in Data Block Encoding have superior compression ratios and also have superior writing speed. Because we generate data from random values and the data almost always changes, it results in the compression ratio of diff being a little higher than that of *fast_diff*, although we expect *fast_diff* is more effective in the actual usage in J-PARC, where there would be many unchanged data to be recorded. Snappy and LZ4 prefer compression speed to compression ratios, but when applying diff or *fast_diff* we just find they have no effect to compress and waste time. On the other hand, GZ is said that it has the same compression speed as a fraction of that of Snappy and LZ4, but we don't find out such a disadvantage in our result. That would be because our hardware has much faster CPU in comparison with I/O.

In summary, our measurement result suggests that the new cluster has more scalability to the amount of data than the old cluster, but we should have used much larger data in order to point out specifically. Applying *diff* or *fast_diff* in Data Blocking Encoding is quite effective for our table design. Under applying either of them, only GZ is meaningful as compression algorithm.

CONCLUSION

We have archived to establish the procedure to construct a cluster of the practical use level. Now we are planning update the version of HBase, Hadoop and ZooKeeper. Because we had experience of being involved in troubles of their version compatibility, before everything we have just place the focus about making clear the procedure. Having achieved the goal, we are ready to update their versions, and adjust and fix.

We have made the data store redundant, but we also have to make the data capture tool redundant, otherwise it becomes a single point of failure from the viewpoint of data capture. It will be natural to use ZooKeeper currently working with Hadoop and HBase to select active/standby states of the redundant data capture tool.

REFERENCES

[1] S. Fukuta, et al., “Development Status of Database for J-PARC RCS Control System (1)”, Proceedings of the 4th Annual Meeting of Particle Accelerator Society of Japan, August 2007.
[2] A. Yoshii et al., “J-PARC operation data archiving using Hadoop and HBase” Proceedings of the 9th Annual Meeting of Particle Accelerator Society of Japan.
[3] N. Kikuzawa et al., “Development of J-PARC TimeSeries Data Archiver using Distributed Database System”, Proceedings of ICALEPCS2013.
[4] http://hadoop.apache.org/
[5] http://hbase.apache.org/
[6] N. Kikuzawa et al., “Status of Operation Data Archiving System using Hadoop/HBase for J-PARC”, Proceedings of PCaPAC2014.
[7] http://zookeeper.apache.org/
[8] http://ganglia.sourceforge.net/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MONITORING AND CATALOGUING THE PROGRESS OF SYNCHROTRON EXPERIMENTS, DATA REDUCTION, AND DATA ANALYSIS AT DIAMOND LIGHT SOURCE FROM A USER'S PERSPECTIVE

J. Aishima, A. W. Ashton, S. J. Fisher, K. E. Levik, G. Winter, Diamond Light Source Ltd, Didcot, Oxfordshire, UK

Abstract

The high data rates produced by the latest generation of detectors, more efficient sample handling hardware and ever more remote users of the beamlines at Diamond Light Source require advanced data reduction and data analysis tools to maximize the potential benefit to scientists. Here we describe some of our experiment data reduction and analysis steps, including real time image analysis with DIALS, our Fast DP and xia2-based data reduction pipelines, and Fast EP phasing and DIMPLE difference map calculation pipelines that aim to rapidly provide feedback about the recently completed measurements. SynchWeb, an interface to an open source laboratory information management system called ISPyB (co-developed at Diamond and the ESRF), provides a modern, flexible framework for managing samples and visualizing the data from all of these experiments and analyses, including plots, images, and tables of the analysed and reduced data, as well as showing experimental metadata, sample information.

INTRODUCTION

The availability of fast, continuous readout detectors (e.g. Pilatus3, Dectris[1]), highly automated sample handling systems such as BART robots[2], and intense X-ray beams at synchrotron sources have been crucial in increasing the rate of sample throughput and diffraction data collection at modern macromolecular crystallography (MX) beamlines. Non-automated evaluation of the many thousands of diffraction images that are generated per measurement in such experiments is no longer feasible, necessitating automated data reduction and analysis techniques combined with a laboratory information management system (LIMS) (SynchWeb [3]) that provides the user with the information necessary to evaluate the progress of their experiment while at the beamline.

SAMPLE EVALUATION

A typical macromolecular crystallography experiment on a beamline begins with visually locating a single crystal or using grid scanning[4] to locate samples or subsamples whose diffracting ability are assessed directly. For experiments where heavy atoms are incorporated for phasing purposes[5], a fluorescence scan may be performed to determine the optimal parameters for anomalous data collections.

STRATEGY

To determine the data required for an experiment, a screening data collection is typically performed with images at 0, 45, and 90 degrees. EDNA[6], a Python pipeline wrapping underlying software, is used on a multi-core computing cluster to calculate optimal data collections for several situations including native, anomalous, gentle (for radiation-sensitive samples), single anomalous diffraction (SAD), and considering measured flux. If applicable to the beamline, kappa alignment parameters (using XOAlign[7]) may also be calculated. Results of the strategy calculations are available in SynchWeb (Figure 1) and the desired EDNA strategy may also be retrieved when setting up a data collection using the Generic Data Acquisition software used at Diamond (GDA) [8].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Strategies Mosflm: ✔ EDNA: ✔

XOAlign

Axes	Kappa	Phi
[(c*, b*), (c*, a*)]	40.059	72.737

EDNA

Space Group	A	B	C	α	β	γ
P3	92.85	92.85	127.90	90.00	90.00	120.00

Lookup Cell

Strategy	Description	Ω Start	Ω Osc	Res (Å)	Rel Trn (%)	Abs Trn (%)	Exposure (s)	No. Images
Strategy1	Standard Native Dataset Multiplicity=3 I/sig=2 Maxlifespan=200 s	114	0.10	1.22	100.0	100	0.142	990
Strategy2	Standard Anomalous Dataset Multiplicity=3 I/sig=2 Maxlifespan=200 s	81	0.10	1.25	100.0	100	0.088	1950
Strategy3	strategy with target multiplicity=16, target I/sig=2 Maxlifespan=200 s	0	0.10	1.22	100.0	100	0.040	3600
Strategy4	Gentle: Target Multiplicity=2 and target I/Sig 2 and Maxlifespan=20 s	135	0.10	1.36	62.0	62	0.040	780
Strategy5	UnderDEV Anomalous Dataset, RadDamage of standard protein	81	0.10	1.25	100.0	100	0.088	1950

Mosflm

Space Group	A	B	C	α	β	γ
P3	92.89	92.89	127.91	90.00	90.00	120.00

Lookup Cell

Strategy	Description	Ω Start	Ω Osc	Res (Å)	Rel Trn (%)	Abs Trn (%)	Exposure (s)	No. Images
anomalous		39	0.20	1.18	0.0	0	0.000	600
native		54	0.20	1.18	0.0	0	0.000	600

Figure 1: SynchWeb display of alignment (XOAlign[7]) and strategies (EDNA[6] and Mosflm[9]) of thermolysin.

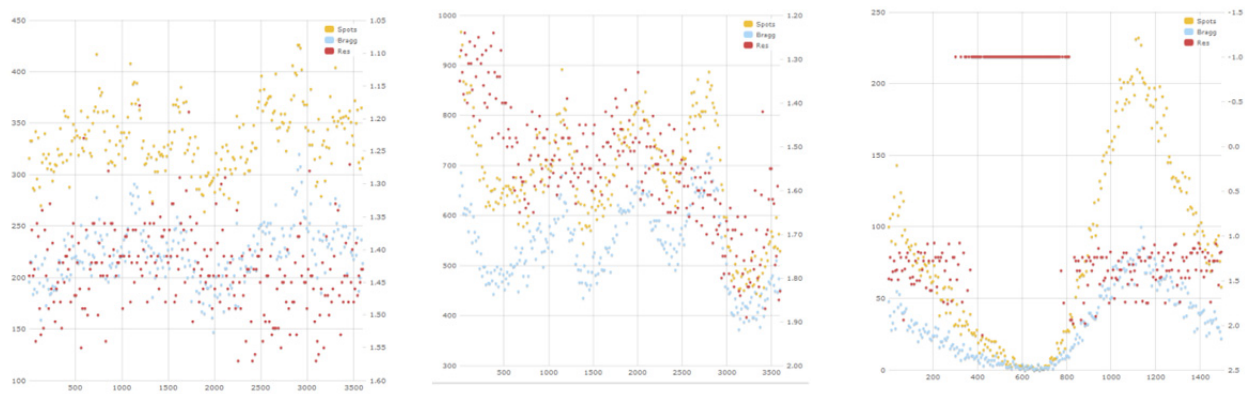


Figure 2: A normal crystal has no decrease in number of spots or worsening resolution after thousands of collected images (left), radiation damage causes a reduction in spots over time (center), and mis-centered crystals cause a region with no diffraction (right). In all images, numbers of diffraction spots in an image are indicated in yellow (all spots) or blue (Bragg diffraction spots), while red indicates resolution of the image.

PER-IMAGE DATA ANALYSIS

While most data analysis occurs after the data collection has finished and the data has been reduced, a running plot of the observed diffraction of the current sample can be monitored by analysing a subset of the diffraction images. By applying spot finding algorithms in DIALS [10] to 200-500 diffraction images distributed uniformly throughout the data set, data collection pathologies may be detected by the user by monitoring the trends during the data collection. Examples of a plot from a normal crystal and some pathological cases are shown in Figure 2, and include radiation damage (decrease in the number of spots visible after e.g. 360 degrees of exposure to the X-ray beam) and crystal miscentring (there is diffraction at the start and end of an experiment, but diffraction completely disappears during the experiment as the sample is no longer illuminated).

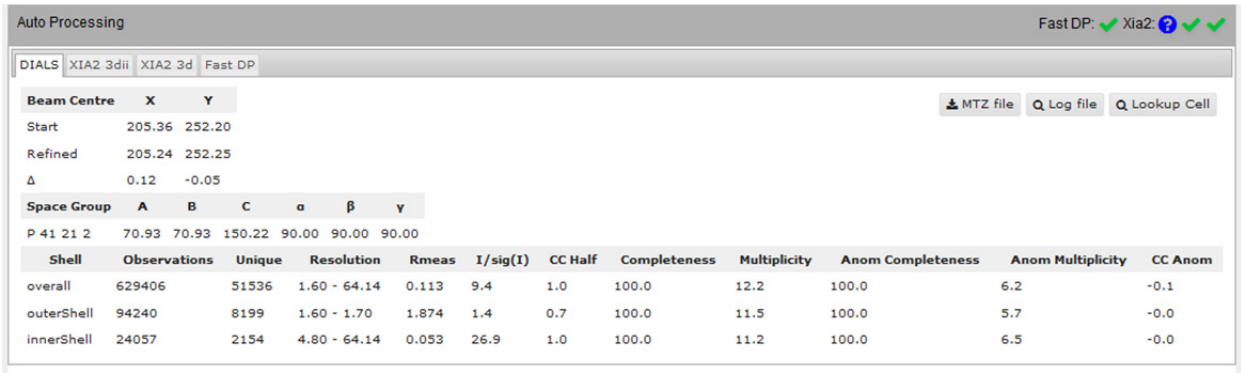
DATA REDUCTION

As the raw diffraction images are too numerous to manually inspect, data quality measures derived from data reduction are typically the better method of assessing the quality of the resulting data. As soon as the data collection has completed, data reduction is triggered to run the software packages Fast DP [11] and xia2 [12] on multi-core computing clusters.

Fast DP runs XDS [13] for the majority of its initial indexing and multiple integration steps. Pointless [14] and Aimless [15] are used to determine the correct point group and merge the data. Fast DP is designed to run

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors



Beam Centre	X	Y
Start	205.36	252.20
Refined	205.24	252.25
Δ	0.12	-0.05

Space Group	A	B	C	α	β	γ
P 41 21 2	70.93	70.93	150.22	90.00	90.00	90.00

Shell	Observations	Unique	Resolution	Rmeas	I/sig(I)	CC Half	Completeness	Multiplicity	Anom Completeness	Anom Multiplicity	CC Anom
overall	629406	51536	1.60 - 64.14	0.113	9.4	1.0	100.0	12.2	100.0	6.2	-0.1
outerShell	94240	8199	1.60 - 1.70	1.874	1.4	0.7	100.0	11.5	100.0	5.7	-0.0
innerShell	24057	2154	4.80 - 64.14	0.053	26.9	1.0	100.0	11.2	100.0	6.5	-0.0

Figure 3: SynchWeb displaying a typical set of DIALS data reduction results including beam centre, unit cell parameters, and statistics. Xia2 and Fast DP reduction results are also available by clicking on the appropriate tab.

quickly and can complete reduction of a single dataset of up to 3600 images from Pilatus3-6M detectors within a minute depending on data quality.

xia2 has been developed as the more thorough data reduction system. xia2 runs multiple data reduction processes with the possibility to run several different software packages simultaneously. xia2 can be run utilizing Labelit [16] to index using 3 images (3d mode) or XDS and all frames for indexing (3dii mode). xia2 may also be run with DIALS being used for indexing (dials mode). Results from xia2 are typically available within 10 minutes of starting the data reduction process.

Results from all of the data reduction processes that complete successfully, including space group, unit cell parameters, completeness, resolution, signal-to-noise ratio and the reduced datasets, are stored in the ISPyB database and may be viewed by using the SynchWeb page.data analysis - phasing

The production of reduced data from Fast DP can sometimes be followed by data analysis for determining the potential for experimental phasing.

Initial conditions to performing this analysis include high data completeness (> 80%), significant differences in anomalous pairs (dI/sigma(dI) >= 1 if data extend to equal to or worse than 2A resolution, or dI/sigma(dI) >= 0.8 if data extend to better than 2A resolution). If these conditions are met, Fast EP is run [11]. This software pipeline consists of SHELXC, D, and E [17] run on a computing cluster with multiple possible values of spacegroup and number of sites are tested first in SHELXD, with the best atom positions used for phasing and solvent flattening with multiple solvent fractions in SHELXE. The best results are then used to calculate electron density maps. Putative heavy atom locations and occupancies, figure of merit, correlation coefficients, and electron density maps may all be viewed in the SynchWeb interface.

DATA ANALYSIS – DIFFERENCE MAPS

To determine whether ligands are bound to proteins, users may provide an atomic coordinate file (PDB format[18]) that can be used for difference map calculation performed by DIMPLE [11]. The coordinates that best match the reduced data point group and unit cell parameters undergoes REFMAC5 [19] rigid body refinement. Further Phaser [20] molecular replacement may be performed if the initial model was not similar enough in orientation to the correct structure. An image of the largest region of difference map density with the underlying molecular model is rendered and is shown in SynchWeb. (Figure 4)

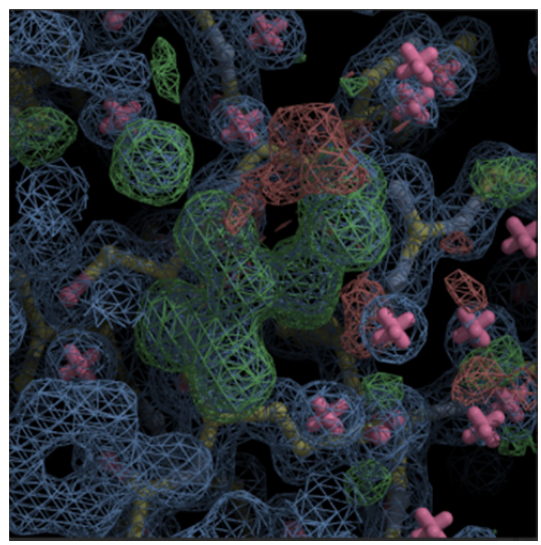


Figure 4: DIMPLE-calculated difference map.

CONCLUSION AND EXTENSION TO OTHER SCIENTIFIC METHODS

Improvements in all phases of Sample Handling, Data Collection, Reduction, and Analysis for MX beamlines have resulted in doubling of samples handled per shift between 2010 and 2015 on beamline I03 at Diamond (Katherine McAuley, personal communication).

Other domains of science are experiencing similar advances that will require better and more automated methods of analysis so that users are able to interpret

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

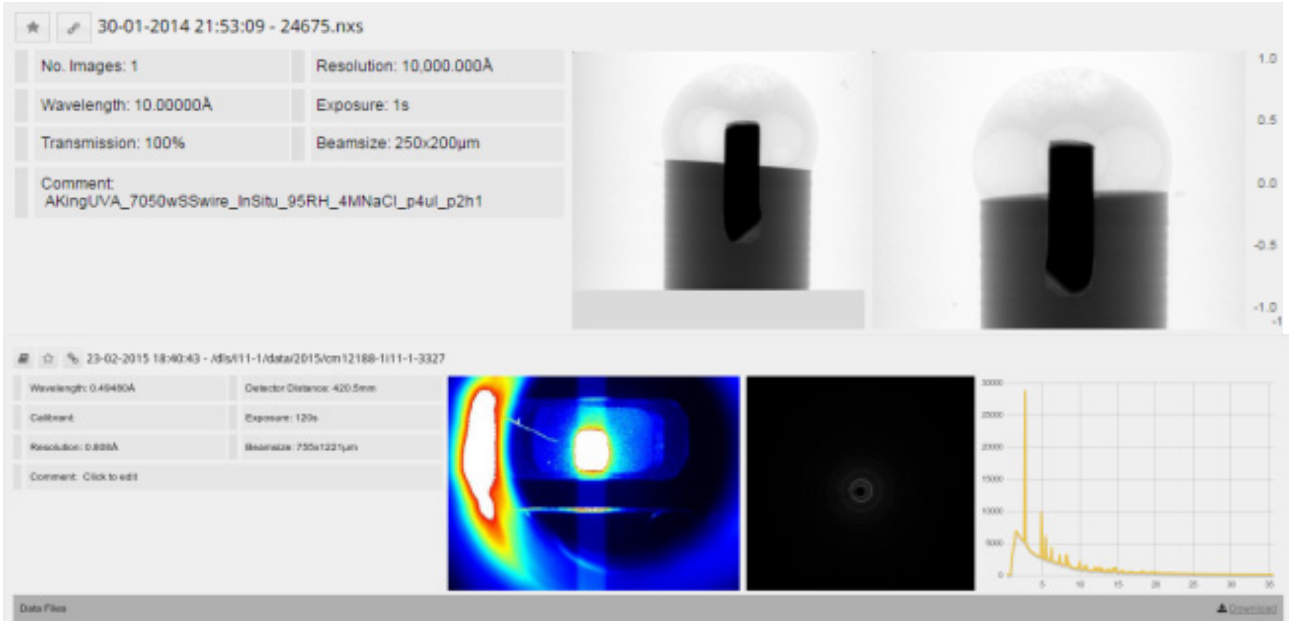


Figure 5: SynchWeb pages for tomography (top) and powder diffraction (bottom).

experiments. The underlying ISPyB database schema used by SynchWeb has been made more generic to allow sensible storage of data from these scientific domains.

Two particular sciences where SynchWeb is currently used at Diamond for experiment evaluation are tomography and powder diffraction (Figure 5).

ACKNOWLEDGMENT

Thanks to Patrick Collins for the miscentered crystal per-image-analysis image and Data Reduction SynchWeb image, Scientific Software for implementing and improving the software pipelines, IT Support for maintaining the cluster computing systems, and CCP4 for providing the effort for supporting and improving DIMPLE.

REFERENCES

[1] http://www.dectris.com
[2] K. McAuley, et al. in Proc. of SRI2015, New York City, NY, USA (2015).
[3] S. J. Fisher, et al., "SynchWeb: a modern interface for ISPyB". J. Appl Cryst. (2015) 48, 927-932.
[4] J. Aishima, et al. "High-speed crystal detection and characterization using a fast-readout detector". Acta Cryst. (2010) D66, 1032-1035.
[5] G. L. Taylor. "Introduction to Phasing." Acta Cryst. (2010) D66, 325-338.
[6] M.-F. Incardona, et al. "EDNA: a framework for plugin-based applications applied to X-ray experiment online data analysis." J Synch Rad (2009) 16, 872-879.
[7] P. Legrand, (2009). xdsme, http://code.google.com/p/xdsme/.
[8] http://www.opengda.org
[9] A. G. W. Leslie and H. R. Powell. (2007) "Processing Diffraction Data with Mosflm." *Evolving Methods for Macromolecular Crystallography*. 245, 41-51.
[10] http://dials.sf.net/.
[11] G. Winter, K. E. McAuley. "Automated data collection for macromolecular crystallography." Methods (2011) 55, 81-93.
[12] G. Winter. "xia2: an expert system for macromolecular crystallography data reduction" J. App. Cryst. (2010) 43, 186-190.
[13] W. Kabsch. "XDS." Acta Cryst. (2010) D66, 125-132.
[14] P. Evans. "An introduction to data reduction: space-group determination, scaling and intensity statistics." Acta Cryst. (2011) D67 282-292.
[15] P. R. Evans, G. N. Murshudov. "How good are my data and what is the resolution?" Acta Cryst. (2013) D69, 1204-1214.
[16] N. K. Sauter, et al. "Robust indexing for automatic data collection." J. Appl. Cryst. (2004) 37 399-409.
[17] G. M. Sheldrick. "Experimental phasing with SHELXC/D/E: combining chain tracing with density modification." Acta Cryst. (2010) D66, 479-485.
[18] http://www.wwpdb.org/documentation/file-format
[19] G. N. Murshudov, et al. "REFMAC5 for the refinement of macromolecular crystal structures".
[20] A. J. McCoy, et al. "Phaser crystallographic software." J. Appl. Cryst. (2007) 658-674.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

FLYSCAN: A FAST AND MULTI-TECHNIQUE DATA ACQUISITION PLATFORM FOR THE SOLEIL BEAMLINES

N. Leclercq, J. Berthault, F. Langlois, S. Le, S. Poirier, Control and Data Acquisition Group, Synchrotron SOLEIL, France
J. Bisou, F. Blache, Electronics Group, Synchrotron SOLEIL, France
K. Medjoubi, C. Mocuta, Scientific Division, Synchrotron SOLEIL, France

Abstract

Synchrotron SOLEIL [1] is continuously optimizing its 29 beamlines in order to propose state of the art synchrotron radiation based experimental techniques to its users. Among the topics addressed by the related transversal projects, the enhancement of the computing tools is identified as a high priority task. In this area, the aim is to optimize the beam time usage providing the users with a fast, simultaneous and multi-technique scanning platform. The concrete implementation of this general concept allows the users not only to acquire more data in the same amount of beam time, but also to address specific phenomena otherwise difficult to tackle (like processes involving rapid transformation of the sample).

This paper gives the reader a detailed overview of the so called 'Flyscan' platform. It notably details the solution retained to generalize a prototype previously developed as a proof of concept [2][3]. Some application examples are also reported.

MOTIVATION AND SPECIFICATIONS

The main motivation for launching the Flyscan project was beam time optimization. Acquiring data faster obviously means obtaining more exploitable data at constant beam time. But it also means that one will be able to quickly discover that experimental conditions will not provide usable data – a benefit which can potentially save hours of expensive and highly sought out beam time.

From a scientific point of view, fast data acquisition also gives access to dynamics of processes involving rapid transformation of the sample (e.g. thermal annealing, continuous mechanical deformation, etc.).

The technical translation of the requirement above led to the specification of an N-dimensional scanning platform whose first dimension is "continuous". Here continuous is opposed to the step-by-step approach – *i.e.* the classical scanning technique known to generate a huge amount of dead time (acceleration / deceleration times of the motors, steady state return information, etc.). In order to reinforce the concept of beam time optimization, the ambition is also to combine the continuous acquisition approach with a multi-technique solution on the sensors (*i.e.* detectors) side. Clearly speaking, the idea is to simultaneously collect – spatially or temporally – correlated data from different sensors along the continuous trajectory of one (or more) actuator(s).

DESIGN OVERVIEW

From a technical point of view, the Flyscan backbone relies on 3 buses: the timing bus, the communication bus and the data bus.

The timing bus is in charge of the synchronization signals (*i.e.* trigger) distribution. Timing events are polymorphic and can be dispatched in the form of hardware (*e.g.* TTL) or software (*e.g.* UDP) notifications. The communication bus routes the control oriented exchanges between the scan actors. Since the latter are implemented as Tango devices [4], these exchanges are mainly composed of Tango queries and events emitted over an Ethernet network. Finally, the data bus transports the experimental data from their respective source to a central point where they are merged into a common repository. The choice has been made to use HDF5 [5] files for both data transport and storage. Figure 1 illustrates the Flyscan architecture.

It results that the Flyscan system can be seen as an aggregation of individual components sharing a common timebase to guarantee temporally – and consequently, spatial - correlation of data produced by heterogeneous sensors distributed over a network.

IMPLEMENTATION DETAILS

Timing

In the Flyscan scheme, everything starts with a software or hardware master clock. The master clock source selection is guided by application requirements. For continuous motion driven experiments – such as tomography or X-ray microscopy – a hardware solution, named *Spietbox*, has been developed to generate a spatially periodic pattern (*e.g.* one clock pulse every tenth of a degree of a rotation) [6]. The *Spietbox* can be coupled with a counting board in order to sample and record the exact axis positions at which data is acquired on each data source while the axis is moving. Solutions also exist to generate slave clocks.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

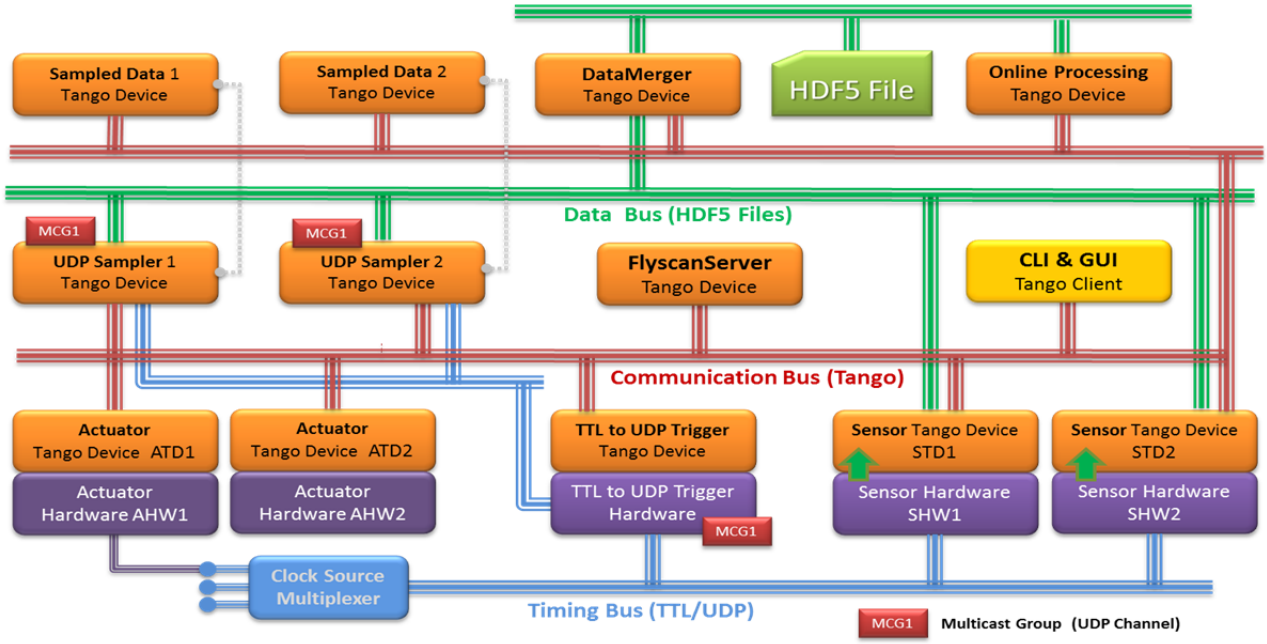


Figure 1: The Flyscan Architecture

Data Production and Transport

The timing signal is then distributed to any data source (or sensor) involved in the scan process. In most cases, a TTL signal in simply injected into the trigger input of the associated hardware. The latter is itself bound to a Tango device whose one of the main tasks is to save the acquired data into a temporary HDF5 file. The produced file is then – directly or indirectly – placed in one of repositories monitored by the data merging process. The Flyscan deployment scheme is mainly governed by technical and/or performance constraints. It is usually chosen so that the "biggest data" – such as 2D detector images – is not moved over the network. In cases where data displacement is inevitable, high speed network links can be setup to deploy our optimized FTP service.

It could seem technically archaic to use files as a data transport service. In fact, this approach offers numerous advantages. First of all, files act as data buffers whose size might not be reached by a RAM based implementation. Moreover, by their nature, files provide a way to asynchronously process gigabytes of data by relaxing the performance constraint on the consumer side. Once stored into a file, the data can safely wait for being processed, consumed or moved. Finally, technologies such as 10 Gb/s Ethernet, SSD, tmpfs… offer OTS solutions to implement file based exchanges with the expected performances.

However, the Flyscan has been globally designed as a general purpose solution and implemented using OTS technologies. Consequently, it could reach its limits when confronted with use cases producing very high data throughputs. Today, such applications – like ultrafast tomography – are specifically addressed by optimized and specially designed hardware and software infrastructures [7]. The Flyscan is not designed to compete with these kinds of applications. See the PISCHE beamline use case example for some Flyscan performance figures related to file transfers.

Online Processing and Visualization

Among the Flyscan specifications, online processing and visualization was one of the most challenging demands. In some applications – *e.g.* long scanning process, such as 2D raster scans – providing the user with a live feedback is mandatory. Here, the idea is to propose a generic mechanism for visualizing both raw and preprocessed data in order to validate the experimental context while the data acquisition is in progress.

Since the whole data scan is merged into a single or multiple files it is directly exploitable by any process with proper read access to the file(s). The "one writer, several readers" problem has been easily solved by pre-allocating the data into the HDF5 file(s) produced by the merging process. Our experience proves that space pre-allocation is a safe and reliable cunning to process the data while the acquisition is in progress (*i.e.* while data is continuously merged and written to the file).

This online processing and visualization requirement has been implemented in the form of a Python [8] Tango device supporting a plugin mechanism. It then becomes straightforward to process the raw data and to make the resulting information available on the network as Tango dynamic attributes. Any Tango aware client can then display this information in order to provide the expected feedback.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Scan Configuration, Sequencing and Monitoring

The core component of the Flyscan implementation is a Tango device whose aim is to orchestrate and monitor the scan process: the FlyscanServer (FSS). From a client point of view, the FlyscanServer is the unique entry point of the system. The main FSS activity is to delegate the actions constituting the scan sequence to abstract entities called "actors". The FlyscanServer currently support the following actors: continuous-actuators, actuators, sensors, timebases, hooks and monitors.

Continuous-actuators are associated with the physical parameters whose continuous variation represents the first dimension of the scan. Actuators incarnate the same concept for a step-by-step variation in any other dimension of the scan. Sensors are the experimental data sources producing the temporary files consumed by merging process. As its name might suggest, a timebase represent a timing signal generator. A hook is a user defined action that can be performed at several "key moments" of the scanning sequence. Finally, a monitor represents a required experimental condition for the scan to be performed.

The system allows attaching an unlimited number of actors to each scan action. For instance, two or more continuous-actuators can be attached to the continuous dimension of the scan; similarly several timebases can be managed in the same scanning sequence. In order to fulfil the initial specifications – *i.e.* the multi-technique constraint – the number of sensors is obviously unlimited.

For the FlyscanServer, an actor – whatever its type – is an abstract entity encapsulating the implementation details of the subsystem its represents. The FSS simply delegates the execution of a specific action to a specific plugin. The latter can encapsulate a very simple – or, on the contrary – quite complex logic. In the average case, the plugin acts as a proxy to a Tango device representing the actual actor (motion axis, counting board, 2D detector …). In order to offer the highest level of flexibility, plugins can be written in either C++ or Python.

A FlyscanServer plugin comes with all the details about the configuration parameters of its associated subsystem. Information such as parameter unit, data type, data format, default value, documentation or specific constraints is provided. This feature allows implementing smart Flyscan clients using an introspection of the plugins parameters description. Thus, the Flyscan platform gives the user access to any actor specific configuration parameters in a generic manner and allows these actors to be finely tuned and controlled.

Despite the flexibility of the approach, handling actors' configuration at this level of detail can lead to complexity. Beyond the fact of being obliged to valuate each parameter – which is error-prone – it also requires each user to have a deep knowledge of each actor's behaviour. In order to reach the expected user-friendliness, a so called "easy-configuration" mechanism has been implemented in order to propose a way to configure the scan with a small set of "master parameters". The latter are defined by an application expert and are supposed to allow computation of the whole set of scan parameters. The easy-configuration mechanism is implemented in the form of a python script executed by the FlyscanServer as the first step of the scan sequence. This approach encapsulates (*i.e.* hides) the technical details and the specificities of a given experimental setup into the expert's python code. It is important to note that the easy-configuration feature also provides the end-user with an "application oriented" view of our generic scanning platform. Thus, the Flyscan deployment can be fully customized to hide all the details in order to fit with the users' experimental habits.

The Flyscan Client Interfaces

The Flyscan comes with a Python command line and a Java GUI. The former is an *ipython* profile offering access to the Flyscan python client API. Beyond its scan configuration capabilities, the GUI also provides a visual feedback on actors' activity.

APPLICATION EXAMPLES

Tomography on PSICHE

One of the PSICHE beamline [9] experimental platforms is dedicated to tomography. The current setup integrates a 2D detector generating 8MB/image at up to 90 Hz. A local technical constraint requires the images to be transferred from the detector host to the one hosting the merging process. Despite this constraint, an average data throughput of 5.4 Gb/s (with 6.1 Gb/s peak) has been achieved on a dedicated 10 Gb/s network link using the Flyscan FTP service (which is itself implemented by 2 Tango devices implementing a subset of the FTP protocol). The 4000 images acquired during the 360° sample stage rotation are finally obtained in an average time of 50s.

X-ray Microscopy on NANOSCOPIUM

The NANOSCOPIUM [9] beamline, currently under commissioning, is aimed to high-resolution fast scanning multi-modal imaging. It will provide high sensitivity elemental, and structural mapping with down to 30 nm spatial resolutions by fast scanning XRF (X-ray fluorescence) imaging combined with absorption, differential phase contrast and dark field imaging. The Flyscan architecture, including the FSS, has been deployed on a microprobe prototype station. A 2D preliminary fast scanning test experiment has been performed. A nanostructure calibration chart – which includes a 250 µm x 75 µm SOLEIL logo – was used as a test sample. A scan of 500 x 1000 pixels with 4ms of dwell time took 35 minutes to complete and produced a

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

HF5 file of approximately 100 Go of raw. Thanks to the Flyscan online data processing mechanism, live reconstructions have been performed to produce some information-rich feedback to the users. Such an experiment could not have been achieved without a continuous scanning approach.

Raster Scans on DIFFABS

On the DIFFABS [9] beamline a performance gain of nearly 60 was measured using the Flyscan. A 121 x 121 raster scan with a 10ms integration time per point has been obtained in 4h28' in step-by-step mode. In the same conditions, a 121 x 225 raster scan completed in 8'40'' when managed with the Flyscan. Large area (20 x 20 mm) fluorescence raster maps were also performed with a 200 µm resolution (~ 10^6 points) and 10ms dwell time in slightly less than 3 hours.

Figure 2 presents results based on real performance measurements and linear extrapolation.

FUTURE AND PERSPECTIVES

Deployments

We already identified and fully specified 30 potential Flyscan applications on 14 of the 29 SOLEIL beamlines. As of this paper, the Flyscan platform is currently in production on 3 beamlines. The deployment schedule is under discussion – taking into account the scientific priorities and resources availabilities.

Developments

On the hardware side, we just started a collaboration on the so called "*PandA*" project [10] with the DIAMOND Light Source [11]. The main objective of *PandA* is the development of a powerful and feature rich hardware solution dedicated to continuous motion oriented applications. It means that *PandA* will replace the *Spietbox* in the near future.

On software side, the refactoring and enhancement of the merging process will constitute the major part of the activities in the next months. Compressed data support is one of the main topics that will be addressed by this work.

In parallel, we are also working on advanced features – such as continuous scan in the "reciprocal space" (aka hkl scans) for X-ray diffraction measurements. A solution already exists at SOLEIL and we are now on our way to make it available to any beamline with an experimental setup dedicated to crystallography.

CONCLUSION

The Flyscan project led to a highly scalable and tunable solution for continuous and multi-technique scans. The obtained performances and the provided features make the solution fulfill the initial requirements and specifications.

The Flyscan already saved tens of hours of beam time and allowed to analyze a huge quantity of samples. The platform extensions currently under development will offer even more experimental capabilities at medium term.

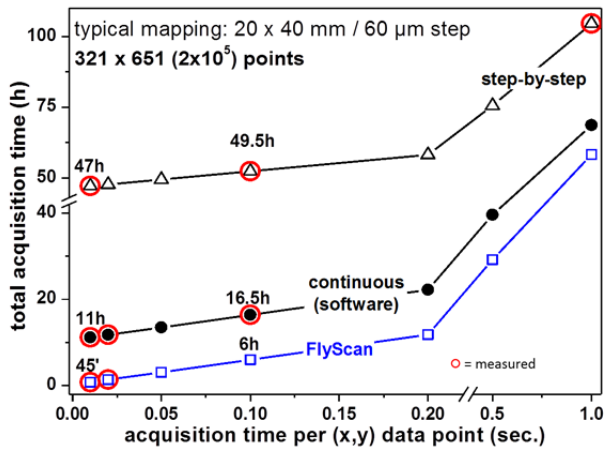


Figure 2: Performance comparison between several scanning techniques: step-by-step, user's implementation and the Flyscan. Thanks to the optimization work done on some actors, the results are now even better than the ones reported here (a supplementary gain of approx. 15% has been obtained).

ACKNOWLEDGEMENTS

The authors are grateful to the SOLEIL management team for is constant support.

As the project leader, I, Nicolas Leclercq, express my deep gratitude to my SOLEIL colleagues involved in this transversal project. I particularly thank the coauthors from their respective contributions.

REFERENCES

[1] Synchrotron SOLEIL: http://www.synchrotron-soleil.fr
[2] Medjoubi et al., Journal of Synchrotron Radiation
[3] Langlois et al., ICALEPCS 2011, Grenoble, France, WEPKN003
[4] Tango Controls: http://www.tango-controls.org
[5] HDF5: http://www.hdfgroup.org
[6] Y.M. Abiven et al., ICALEPCS 2011, Grenoble, France, WEMMU004
[7] Swiss Light Source giga-Frost project: https://www.psi.ch/sls/tomcat/detectors#Detectors_and_Optics
[8] Python: https://www.python.org
[9] http://www.synchrotron/soleil.fr/Recherche/LignesLumiere
[10] I. Uzun et al. - PandA Motion Project - Proc. of ICALEPCS'2015 - Melbourne – Australia
[11] The DIAMOND Light Source: www.diamond.ac.uk

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE AUSTRALIAN STORE.SYNCHROTRON DATA MANAGEMENT SERVICE FOR MACROMOLECULAR CRYSTALLOGRAPHY

Grischa R Meyer, Steve Androulakis, Philip Bertling, Ashley Buckle,
Wojtek James Goscinski, David Groenewegen, Chris Hines, Anitha Kannan,
Sheena McGowan, Stevan Quenette, Jason Rigby, Patrick Splawa-Neyman,
James M Wettenhall, Monash University, Clayton, Australia
David Aragao, Tom Caradoc-Davies, Nathan Mudie, Synchrotron Light Source Australia, Clayton
Charles Bond, University of Western Australia, Crawley, Australia

Abstract

Store.Synchrotron is a service for management and publication of diffraction data from the macromolecular crystallography (MX) beamlines of the Australian Synchrotron. Since the start of the development, in 2013, the service has handled over 51.8 TB of raw data (~ 4.1 million files). Raw data and autoprocessing results are made available securely via the web and SFTP so experimenters can sync it to their labs for further analysis. With the goal of becoming a large public repository of raw diffraction data, a guided publishing workflow which optionally captures discipline specific information was built. The MX-specific workflow links PDB coordinates from the PDB to raw data. An optionally embargoed DOI is created for convenient citation. This repository will be a valuable tool for crystallography software developers. To support complex projects, integration of other instruments such as microscopes is underway. We developed an application that captures any data from instrument computers, enabling centralised data management without the need for custom ingestion workflows. The next step is to integrate the hosted data with interactive processing and analysis tools on virtual desktops.

INTRODUCTION

The deposition and the wide availability of raw diffraction data have far-reaching benefits for the structural biology community [1–4]. We recently created TARDIS, a suite of tools for the deposition of X-ray diffraction images in an open access repository to facilitate their deposition using federated institutional repositories [3]. Subsequent engagement with IUCR and closer relationships with the Australian Synchrotron prompted us to develop a new framework for raw X-ray data archival and dissemination — the *Store.Synchrotron* service, which is described in this manuscript.

An earlier but more detailed description of this service is available at [5].

THE OPERATION OF STORE.SYNCHROTRON FOR THE AUSTRALIAN SYNCHROTRON

The *Store.Synchrotron* service (`https://store.synchrotron.org.au`) has been deployed to receive diffraction data from the macromolecular crystallography (MX) beamlines at the Australian Synchrotron (AS) automatically.

When a user starts collecting data at the beamline, all raw diffraction data and automated data processing results are transferred in real-time and on the next day respectively to *Store.Synchrotron* without user intervention.

This system is accessible through any web browser. Data is available immediately at time of collection to all authorised users via the internet. If desired, the data can be opened to the public.

Figure 1 shows a schematic overview of the service.

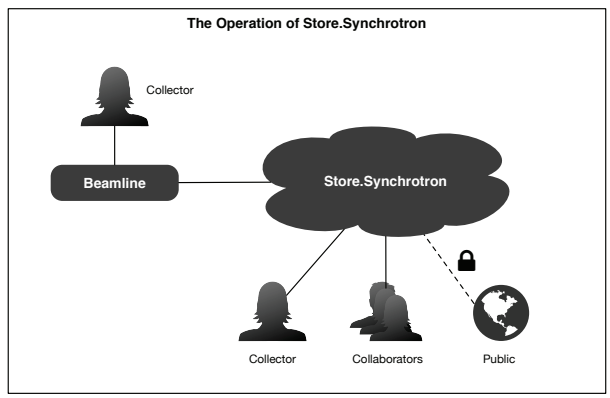


Figure 1: Schematic overview of *Store.Synchrotron*

The Need for an Automated Data Management System

At the Australian Synchrotron (AS) up until recently the most practicable solution for data management has been to take portable hard drives to the beamlines. The use of portable drives is a problem, however, because these drives may remain the lab's sole resource for raw data. After researchers have left, it may be difficult to correlate raw frames with datasets and structures, and a single disk is very prone to failure and data loss.

With *Store.Synchrotron* users now have for the first time a long-term, secure archive of their raw data, including metadata, which can be used to relate raw frames to projects and visits to the Australian Synchrotron.

In addition to the practical need for a robust data archive, new requirements by funding agencies for researchers to have a data management plan are conveniently solved by a

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

centralised solution that takes care of the data from collection to archival and/or publishing.

The End-to-End User Experience

The default way of accessing data on the *Store.Synchrotron* service is via the web through a web browser. As a user collects a diffraction image it is available on the *Store.Synchrotron* website within minutes (Fig. 2) including instrument specific metadata.

Data is organised into three tiers. An Experiment is related to an allocation on the beamline (Fig. 2A,B). The data is then separated into Datasets (Fig. 2C). Finally, the individual diffraction images are Datafiles in their respective datasets (Fig. 2D). All or selective elements of each tier can be downloaded via the web or via SFTP.

In addition to diffraction datasets, *Store.Synchrotron* makes accessible the results of preliminary autoprocessing. Collection of the images triggers automatic indexing and full integration and scaling for complete datasets. All autoprocessing output is downloadable as well.

Design and Implementation

The *Store.Synchrotron* service is an implementation of MyTardis (`http://mytardis.org`); a web-based open-source data management platform. MyTardis began as the TARDIS [3] (`http://tardis.edu.au`) diffraction image repository but has since expanded to suit the storage and organisation of a wide range of scientific data. The interface allows browsing, data access, data sharing and publication.

The beamline registers its the raw images with *Store.Synchrotron* in real-time, triggered by the instrument control system. Different processes extract information and preview images, compute SHA-1 checksums, and copy the diffraction data to the final *Store.Synchrotron*-managed storage location. By the end of an allocation slot both raw data and auto-processing results are archived and made available on *Store.Synchrotron*. For resilience all these processes are run using a message queue. All networked communication is encrypted.

The integrity of the files stored within the *Store.Synchrotron* service is of utmost importance. As the data are the results of often non repeatable experiments, data is verified via checksums at several steps in the pipeline, starting with ingestion. If a checksum does not match, it can be re-sent if possible. All archived files are re-verified regularly to ensure the ongoing integrity of the data.

In a 12 month period since *Store.Synchrotron* began operation, only 8 out of more than 1,700,000 files had mismatching checksums. This means the vast majority of data (99.9995%) was transferred successfully in the first attempt. It also demonstrates the value of integrity checking as these 8 files could be re-transferred before the original was lost.

The *Store.Synchrotron* service receives metadata via its RESTful API. By following the REST standard we made it trivial for developers to interact with *Store.Synchrotron* programmatically, e.g. automatically or manually via different front ends.

Another important technology we use that allows space and time efficient archiving and retrieval of large datasets is SquashFS (`http://squashfs.sourceforge.net/`). SquashFS is an archive file standard that supports compression and direct random file access. *Store.Synchrotron* can transparently provide access to individual files inside SquashFS archives.

Store.Synchrotron was developed based on the web framework Django. It is running on a Ubuntu 14.04. It is hosted on the NeCTAR Project's cloud computing service (`http://www.nectar.org.au/research-cloud`). To assist the maintenance and coordination of this service, we have used an orchestration and configuration management system (SaltStack `http://www.saltstack.com/`).

Benefits from User Perspective

Protein crystallography has benefited for many years by an open-access community based approach to the constant development and improvement of data processing, structure solution and refinement tools. As these tools evolve, it is often beneficial to revisit projects where data collected has proven intractable at the time, as has been done for example in [6].

A broadly systematic approach can be taken to processing old data for new results. Data processing programs are continuously evolving. As the *Store.Synchrotron* public repository grows we can envisage a system where as new methods to analyse raw data become available they are used against old data automatically.

OPEN DATA FOR THE CRYSTALLOGRAPHIC COMMUNITY

The Australian Synchrotron is the only facility of its type in Australia, and is in constant use. As such, the organisation has actively encouraged opening up of data produced there, to encourage efficiencies and reuse. This requires services that enable effective long term storage, curation and citation.

Provision of services related to making data more easily available are timely, as Australian government research funders have been strengthening their language in this area [7, 8], bringing Australia more in-line with the global research environment and that of international funding agencies such as the US National Science Foundation (NSF) and the UK Medical Research Council (MRC).

Store.Synchrotron also represents the logical extension of a long-standing effort in the macromolecular crystallography community to ensure that satisfactory evidence is provided to support the interpretation of structural experiments. This effort has included unrivalled requirements for validation of data interpretation processes [9–12].

Other projects exist to deposit and host openly-accessible diffraction data on the web, such as DIMER (`https://dimer.uq.edu.au/`), the registration-required JCSG Repository of Crystallography Datasets (`http://www.`

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

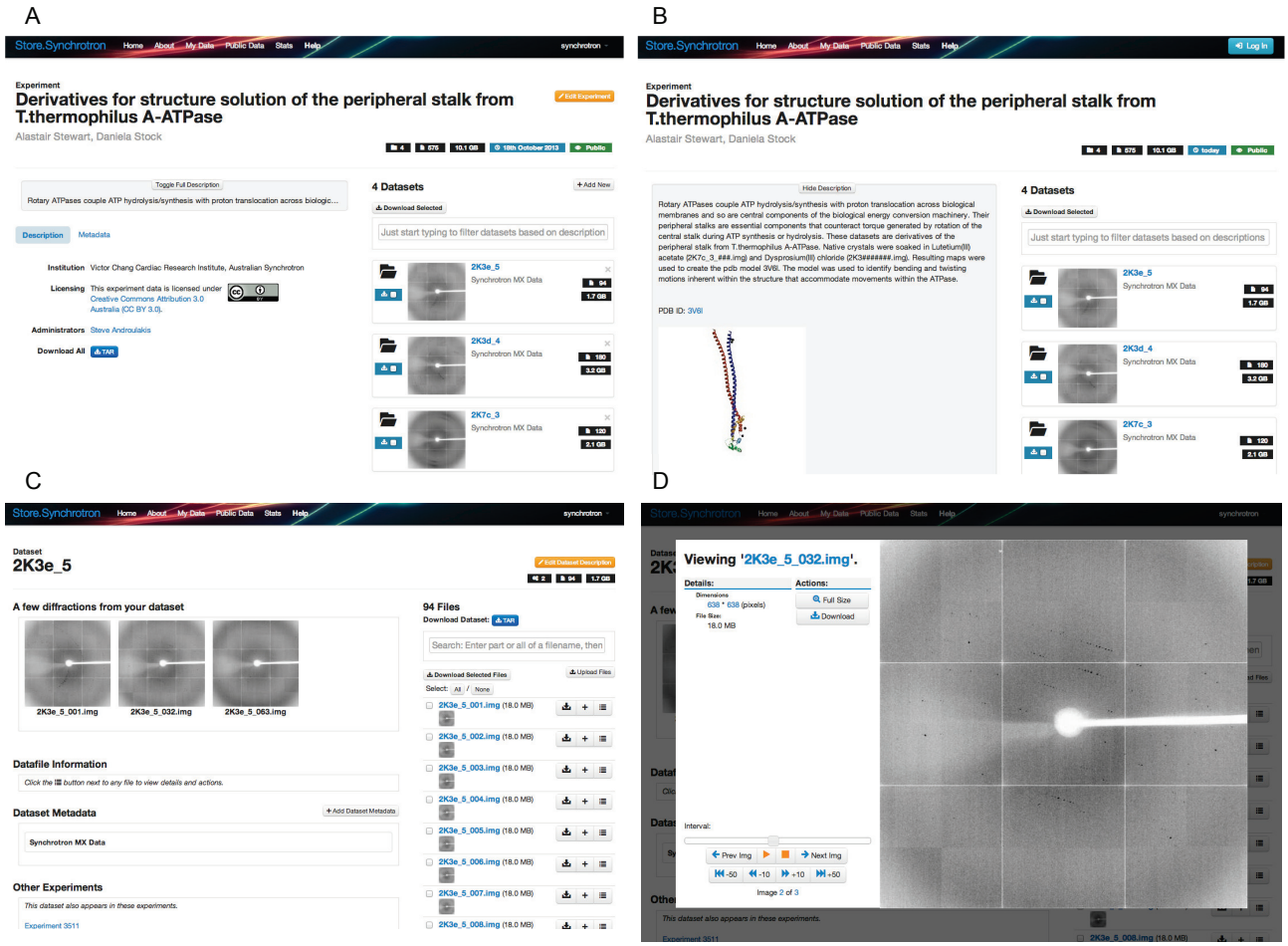


Figure 2: Screenshots of the web interface to the *Store.Synchrotron* service. **A**: A public experiment. **B**: A detailed description with images and links. **C**: One dataset highlighting a selection of images from the set. **D**: Image file preview.

jcsg.org/), and TARDIS (http://tardis.edu.au/). *Store.Synchrotron* differs from these services by automatically archiving all diffraction data produced from beamlines at time of data collection. This act eliminates the need for manual diffraction data deposition by the user as a step toward open crystallography data.

A European project, PaN-Data (http://pan-data.eu/) has also created infrastructure for the automatic archival and dissemination of raw diffraction data from facilities such as the Diamond Light Source and the ESRF. This is achieved through a combination of an information management and real-time data monitoring system ISPyB [13], metadata store ICAT (http://icatproject.org/) and its data-browsing and access web front-end TopCat (https://code.google.com/p/topcat/). To date, access to data has been private to users who collect it. *Store.Synchrotron* extends this functionality by providing both an automatic archival mechanism from the time of data collection and a publishing interface for open access to collected data.

To facilitate the publication process the *Store.Synchrotron* has developed a publication form to assist researchers in identifying and describing the raw data associated with their experiment and adding metadata such as related PDB entries. After a final check by Synchrotron staff the Experiment link is made publicly available and discoverable via the *Store.Synchrotron* site, with a Digital Object Identifier (DOI) minted for citing the data.

Considerations for Use of the Store.Synchrotron Model in Other Facilities

If this system is to serve as a model for evaluation by other synchrotrons and instrument facilities around the world, it is important to consider potential points of difficulty that may arise with the application of our approach. Certainly, the most difficult infrastructure requirement for *Store.Synchrotron* is the expense of storage and compute servers. When operating a data intensive service, a significant challenge arises: Who will pay to keep this data alive?

Monash University is a partner operator of VicNode (http://www.vicnode.org.au), the local node of the Research Data Service (RDS, http://www.rds.edu.au). VicNode provides a multi-location, tape-archived, large volume cloud data storage service that is utilised for all the data stored at the *Store.Synchrotron* service. Storage initially

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

provided to *Store.Synchrotron* by the VicNode is over 200 terabytes in size and is a combination of disk and tape, with an off-site backup.

The minting and ongoing maintenance of Digital Object Identifiers (DOIs) represent an ongoing cost. *Store.Synchrotron* uses the federally-run Australian National Data Service (ANDS) who provide a service that allows the free minting and maintenence of DOIs resolving to Australian research data.

As discussed above, the MyTardis data management and instrument integration application is free, open-source, and able to be deployed on a compute cloud or web server in an automated fashion. We envision that our service could offer an attractive model elsewhere for raw data deposition, access, archival and dissemination.

Integration with Current and Future Instrumentation

Structural biology increasingly harnesses and integrates multiple approaches to attack challenging problems. In addition to protein crystallography, these include electron microscopy, X-ray scattering, and multiscale computational modelling. This creates a real need for integrated, end-to-end data management solutions. Similar services are also being operated that manage data from over 25 instruments at other facilities, including Monash University Micro Imaging platform (Electron Microscopy), the Monash University Medical Proteomics Facility (Mass Spectroscopy) as well as instruments at the Australian neutron source ANSTO. All this work is built upon the MyTardis data management platform and thus forms a common interface and data organisation strategy.

ACKNOWLEDGEMENTS

Auto-processing and the *Store.Synchrotron* was partially supported by the National eResearch Collaboration Tools and Resources (NeCTAR - grant "Monash_Synchrotron_-Storage_Service") project (www.nectar.org.au), and the VicNode/RDSI (grant 2013R2.1).

D.A. would like to thank the Australian National Health and Medical Research Council (APP1047004) for partial funding.

Development of *Store.Synchrotron* was supported by staff at the Australian Synchrotron MX beamline and Scientific Computing groups.

The publication workflow was funded by the Australian National Data Service (ANDS).

S.M. is an Australian Research Council Future Fellow (FT100100690) and she thanks both the Australian Research Council and National Health and Medical Research (Project grants 1025150; 1063786; 1062216) for funding support.

AMB holds a research fellowship from the National Health and Medical Research Council (NHMRC), Australia.

Aspects of this work were funded by the Australian National Health and Medical Research Council (DA, APP1047004; CSB, APP513880 and APP1050585).

REFERENCES

[1] Robbie P Joosten and Gert Vriend. "PDB improvement starts with data deposition." In: *Science (New York, NY)* 317.5835 (2007), p. 195.

[2] T Alwyn Jones and Gerard J Kleywegt. "Experimental data for structure papers." In: *Science (New York, NY)* 317.5835 (2007), pp. 194–195.

[3] Steve Androulakis et al. "Federated repositories of X-ray diffraction images." In: *Acta Crystallographica Section D: Biological Crystallography* 64.7 (2008), pp. 810–814.

[4] J. Mitchell Guss and Brian McMahon. "How to make deposition of images a reality." In: *Acta Crystallographica Section D* 70.10 (Oct. 2014), pp. 2520–2532. DOI: 10.1107/S1399004714005185. http://dx.doi.org/10.1107/S1399004714005185

[5] Grischa R. Meyer et al. "Operation of the Australian Store.Synchrotron for macromolecular crystallography." In: *Acta Crystallographica Section D* 70.10 (Oct. 2014), pp. 2510–2519. DOI: 10.1107/S1399004714016174. http://dx.doi.org/10.1107/S1399004714016174

[6] Sheena McGowan et al. "X-ray crystal structure of the streptococcal specific phage lysin PlyC." In: *Proceedings of the National Academy of Sciences* 109.31 (2012), pp. 12752–12757.

[7] ARC. *Discovery Projects Instructions to Applicants for funding commencing in 2015*. http://www.arc.gov.au/pdf/DP15/DP15_ITA.pdf. 2014.

[8] NHMRC. *Australian Code for the Responsible Conduct of Research*. https://www.nhmrc.gov.au/guidelines/publications/r39. 2007.

[9] Axel T Brünger. "Free R value: a novel statistical quantity for assessing the accuracy of crystal structures." In: *Nature* 355 (1992), pp. 472–475.

[10] Gerard J. Kleywegt and T. Alwyn Jones. "[11] Model building and refinement practice." In: *Macromolecular Crystallography Part B*. Ed. by Robert M. Sweet Charles W. Carter Jr. Vol. 277. Methods in Enzymology. Academic Press, 1997, pp. 208–230. DOI: http://dx.doi.org/10.1016/S0076-6879(97)77013-7. http://www.sciencedirect.com/science/article/pii/S0076687997770137

[11] EN Baker, TL Blundell, and JL Sussman. "Deposition of macromolecular data." In: *Acta Crystallographica Section D* (1996).

[12] Editorial. "Data's shameful neglect." In: *Nature* 461 (2009), p. 145.

[13] Solange Delagenière et al. "ISPyB: an information management system for synchrotron macromolecular crystallography." In: *Bioinformatics* 27.22 (2011), pp. 3186–3192.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A DATA MANAGEMENT INFRASTRUCTURE FOR NEUTRON SCATTERING EXPERIMENTS IN J-PARC/MLF

K.Moriyama# and T.Nakatani, J-PARC, Ibaraki, Japan

Abstract

The role of data management is one of the greatest contributions in the research workflow for scientific experiments such as neutron scattering. The facility is required to safely and efficiently manage a huge amount of data over the long duration, and provide an effective data access for facility users promoting the creation of scientific results. In order to meet these requirements, we are operating and updating a data management infrastructure in J-PAPC/MLF, which consists of the web-based integrated data management system called the MLF Experimental Database (MLF EXP-DB), the hierarchical raw data repository composed of distributed storages, and the integrated authentication system. The MLF EXP-DB creates experimental data catalogues in which raw data, measurement logs, and other contextual information on sample, experimental proposal, investigator, etc. are interrelated. This system conducts the reposition, archive and on-demand retrieve of raw data in the repository. Facility users are able to access the experimental data via a web portal. This contribution presents the overview of our data management infrastructure, and the recent updated features for high availability, scaling-out, and flexible data retrieval in the MLF EXP-DB.

INTRODUCTION

The Materials and Life Science Experimental Facility (MLF) at the Japan Proton Accelerator Research Complex (J-PARC) is the experimental facility for neutron scattering, providing domestic and international users in a wide variety of research fields with one of the highest intensity pulsed neutron beam in the world. Twenty neutron experimental instruments equipped with large-area neutron detectors and various sample environmental apparatuses are currently in operation. MLF annually supplies beamtime of two hundreds days for about a thousand users and about several hundreds of experimental proposals are performed using these neutron instruments.

Neutron instruments create a huge amount of experimental data. A typical volume of data produced in one experiment is from several tens MB to several hundreds GB. Since a lot of experiments under various experimental conditions are carried out in a short period of time using high intensity beam, the total amount of data generated annually in a whole facility at full performance is on the order of PB.

In those situations, it is required to safely and efficiently manage a huge amount of data over the long duration. In particular, an appropriate protection and preservation of the experimental data as intellectual property produced at a large experimental facility such as MLF is required recently. Furthermore, it is required to provide facility users with a data access service enabling effective utilization of a lot of experimental data. To meet these requirements, we are operating an infrastructure for a central data management and access in the whole facility. In addition, we are updating the MLF EXP-DB, which is the core system of our infrastructure, responding to the expansion and growth of experimental environment based on an improvement plan in the facility. Since the neutron beam currently provided with bombarding of 500kW proton beam is finally increasing up to the intensity equivalent to the power of 1 MW proton beam, therefore the infrastructure should respond the enhancement of data production rate due to the intensity enhancement of neutron beam.

DATA MANAGEMENT INFRASTRUCTURE

A data management infrastructure in J-PARC/MLF is comprised of several database systems and the data repository. The schematic overview of the infrastructure is shown in Fig. 1. The MLF EXP-DB is the core system of the infrastructure, and responsible for managing data and providing data access. This system catalogues experimental data and manages measurement raw data centrally in the whole facility. Measurement raw data is preserved in the data repository. The MLF Business Database (MLF BIZ-DB) collects application information such as experimental proposal, primary investigator, and sample from the front end system of facility users. Also, there are an integrated authentication system (User DB), which manages user credentials of systems comprising infrastructure, and a sample database (Sample-DB) for a safety management of handling samples. The MLF EXP-DB collects automatically every variety of information related to experiment by interacting with these database systems.

In local system of neutron instrument at MLF, the MLF control software framework "IROHA" [1], which is adopted as standard software for measurement, coordinates measurement with controlling data acquisition and sample environmental devices. In response to the message from IROHA, the data acquisition system (DAQ) produces raw data files into the data storage. Also, measurement log file including measurement conditions such as sample device configuration and results is created as "experimental meta-data" in XML-format by IROHA. This experimental

#kentaro.moriyama@j-parc.jp

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

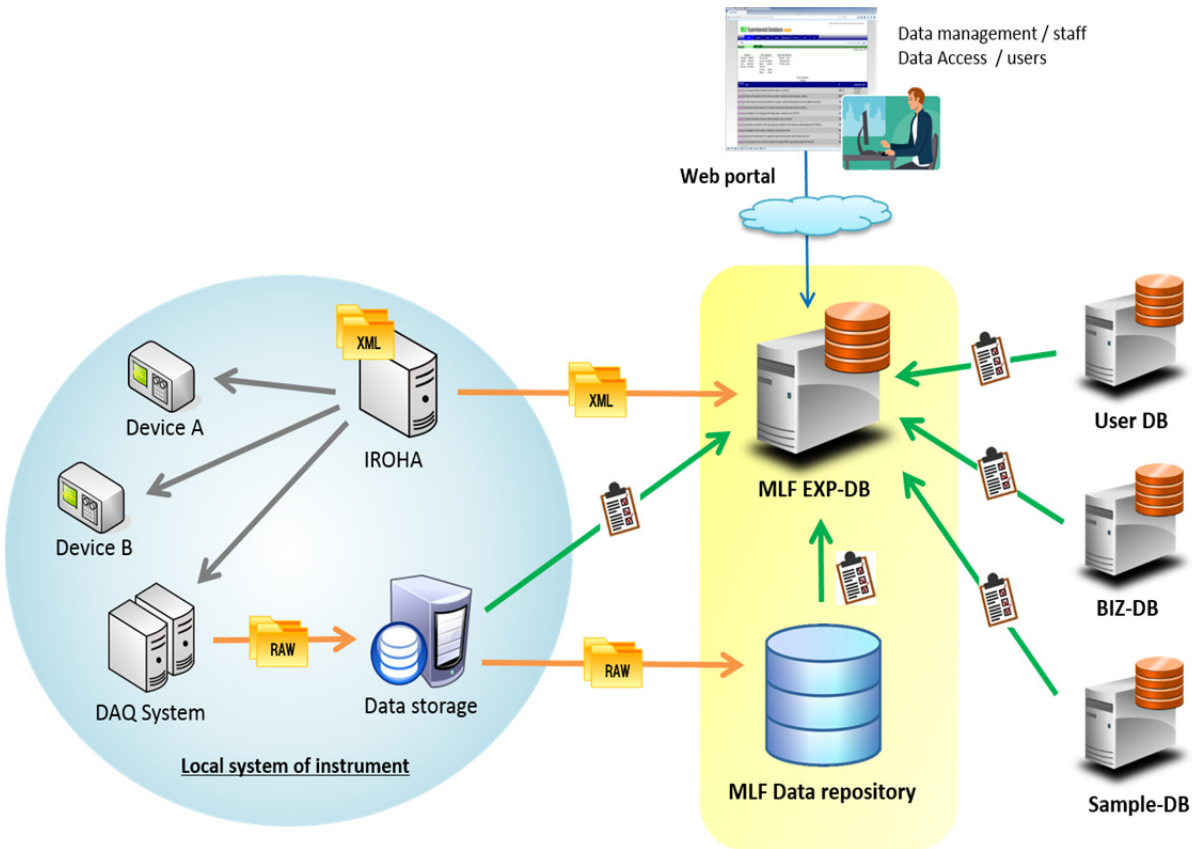


Figure 1: The schematic overview of the data management infrastructure in J-PARC/MLF.

meta-data is crucial information for the management and utilization of the experimental data.

The MLF EXP-DB collects the experimental meta-data and meta-information for raw data such as file name, location path on the storage, and size from local systems of the neutron instruments, and registers them on the experimental data catalogue with associated information collected from other database systems. Raw data registered on the catalogue is transferred from local data storage at instruments to the data repository by the MLF EXP-DB. The MLF EXP-DB provides web portals with the facility staff and users. These web portals are a front-end of the infrastructure. All of data management and access is performed via web portals.

MLF EXPERIMENTAL DATABASE

System Architecture

The system employs a Java-based commercial database software package, named "R&D Chain Management System Software (RCM) [2], based on the XML-DB architecture suitable for the handling data with flexible structure such as our experimental meta-data described in XML-format. The flexibility of the XML-DB is quite suitable for the handling of the experimental meta-data.

The RCM serves an integrated web database system designed on a three-layer model consisting of three components; RCM-Web, RCM-Controller and RCM-DB. These components run on the single physical server. The RCM-Controller has a XML-based workflow engine, and executes any control of database and remote servers according to workflow templates in which settings of job sequence are described in XML-format. Therefore, all of procedures for data cataloguing, raw data management, search for data access, etc. is implemented in the workflow templates. Result of each procedure is converted from XML to HTML using XSLT [3] that is a language for transforming XML documents into other XML documents and then web page is offered to system users via the RCM-Web. The experimental data collected from remote server in local systems of the neutron instruments are registered to the XML tree held on RCM-DB.

Access control is quite important for the data management and access. A disclosure level of data such as public, private, and share should be closely managed. In RCM, information on owner and authority is imparted as attribute to the every XML element. Therefore, access to the XML tree on RCM-DB is controlled in unit of XML element by referring this attribute. Allocating group such as the instrument group or user group to the owner of the XML subset corresponding to the experimental data, the access privileges and disclosure level of data are controlled. A login authentication to the system is performed using LDAP running on User DB.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

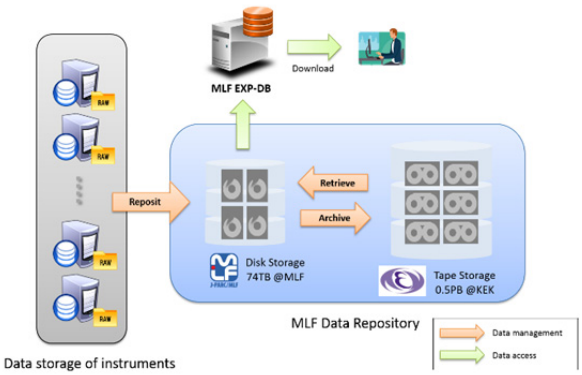


Figure 2: The schematic of the raw data management.

Experimental Data Catalogue

The experimental data catalogue can facilitate the creation of scientific results by supporting data utilization of facility users. It is useful to search experimental data necessary for analysis among a lot of data created under various experimental conditions and samples. Furthermore, it is also possible to refer information on experimental condition instead of conventional log notebook. The data catalogue is a list of measurement indicating various information related to experiment. It contains information as follows:

- **Experimental proposal:** Proposal ID, Beam line (Instrument), Title, Primary investigator (PI) and Period.
- **Measurement:** Run No., Time and date, Measuring time, Number of counts, Comment, etc.
- **Sample:** Sample ID, Name, Chemical formula, Composition, quantity, etc.
- **Device condition:** Device type, Setting parameter, Physical value, etc.
- **Data File:** Name, Size, Location path, Hash value, etc.

These kinds of the information are valuable especially for searching and analysing the experimental data taken in the time-sharing measurement varying experimental condition dynamically for transient phenomenon. A file location path enables the downloading of data stored in the data repository.

Raw Data Management

It is required to effectively and safely store a huge amount of raw data in the data repository with limited space. The data repository is composed of two data storages separately located by a distance of over 50 km. One is the 74TB disk storage at MLF campus and the other is the 0.5PB tape storage at KEK Tsukuba campus. Raw data is handled by the MLF EXP-DB within this reposit in a hierarchical way. The schematic of the raw data management is shown in Fig. 2. Raw data, produced initially in the local storage at each neutron instrument, is transferred to the primary disk storage (this process is called a "reposit") and archived into the secondary tape storage after certain period of time. The archived data is retrieved to the primary disk storage on demand. Meta-information of raw data such as the file location on the repository is updated in every data handling. Facility users are able to request and download their raw data through the MLF EXP-DB.

Web Portals

All of data management and access is performed via web portals. There are two kinds of web portals. One is a data management portal for the facility staff, and the other is a data access portal for users. The experimental data catalogue is available in both portals. Fig.3 shows screenshots of portals. The data management portal enables a various management of experimental data: a curation of data automatically registered, off-line registration after the measurement, management of raw data, quality determination of the measurement, and annotation providing. The data access portal allows the users to browse the experimental data catalogue, search with various conditions and download datasets necessary for analysis, and add annotations. It is also to execute standard data reduction and visualization, and share them among the research group.

Currently, the data management web portal is practically used at several instruments in MLF. On the other hand, the data access portal is still under trial use and development for opening service available to facility users.

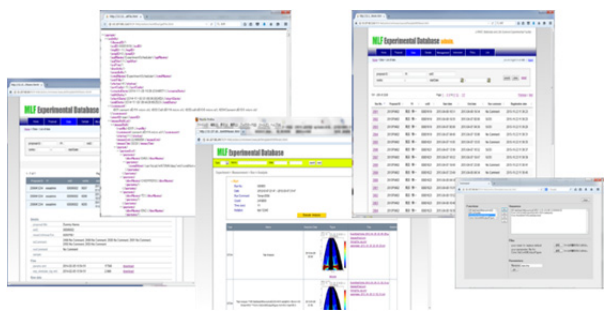


Figure 3: The screenshot of the web portal.

RECENT UPDATED FEATURES

Recently, we have redesigned the MLF EXP-DB to enhance availability and scalability, and improved a function of the experimental data retrieval in the web portal.

For the core system of the data management infrastructure, high availability is required. A service outage owing to the system failure and maintenance applying patches should be avoided as much as possible. A redundancy of system allows avoidance of such service outage, so that availability is improved. We have made the system to a redundant distributed system.

It is also necessary to consider scalability of the system especially in the data cataloguing. The data production rate in the facility is continuously rising. It depends on the beam intensity and the performance of neutron

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

instruments. In MLF, the beam intensity has been increasing in phases according to the development plan for the accelerators and neutron sources. The scale of the experimental data catalogue will be the orders of tens million tags at full power operation in MLF. Also, additional installation of detector and improvement of various devices has been keeping in the instrument. Considering these situations, we have improved the MLF EXP-DB into the system capable of scaling-out responding to the scale of data quantity produced in the facility.

Furthermore, an improvement of experimental data retrieval can promote the data utilization. We have improved the experimental data retrieval with a flexible search condition according to the experimental conditions.

High Availability

The improved redundant system is composed of two servers in a switch over relationship. Figure 4 shows the overview of the system operation in this configuration. When one server is operated as active servers, which is responsible for the collect of the experimental data from instrument local systems and data access, the other is operated as hot-standby server. The database replication is regularly performed from active server to hot-standby server. In the occurrence of system failure or maintenance, the operation is switched. The data access is provided via a front-end web server, so that the user need not to take into account the access destinations.

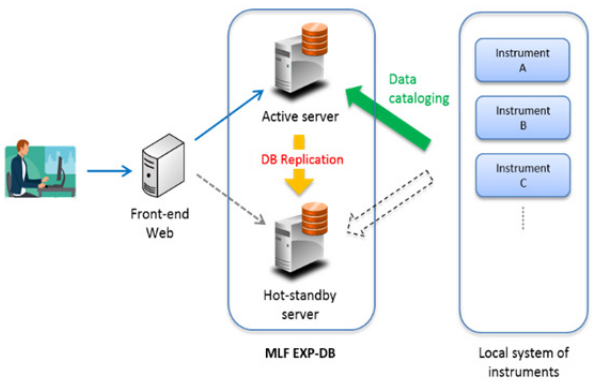


Figure 4: The schematic of the system operation in the redundant distributed configuration.

Scaling-out

The system has been scaled out to improve scalability for the data cataloguing. It is composed of two redundant system mentioned in the previous subsection. There are two experimental halls in MLF, each hall has about 10 instruments in operation. Currently, one system is allocated to the instruments in the 1st experimental hall and other system is allocated to the instruments in the 2nd experimental hall. It is capable to perform further scaling out of a load distribution by adding the redundant systems responding of the data production rate.

Flexible Data Retrieval

Since various sample environmental apparatuses in neutron instruments are used depending on the purpose of experiment, the structure of the experimental meta-data can be changed every experiment or instrument. Although every variety of information associated to experimental conditions as experimental meta-data in XML-format is registered in the experimental data catalogue, the retrieval condition is limited to some basic information on the experiment such as run No., date and time, experimental title, primary investigator in the conventional retrieval function.

In the improved retrieval, it is possible to execute a flexible data search more by specifying the search conditions for each tag of experimental metadata corresponding to the device condition. Figure 5 shows the screenshot of this retrieval.

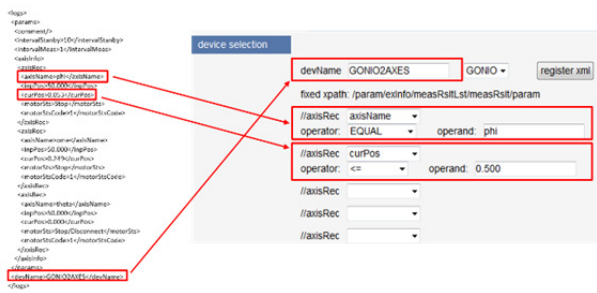


Figure 5: The screenshot of the flexible data retrieval.

CONCLUSION AND FUTURE PLAN

We are operating and updating the data management infrastructure in J-PARC/MLF. The infrastructure is mainly composed of the MLF EXP-DB, which is responsible for data management and access, and the data repository for the central preservation of experimental raw data. Currently, it manages the experimental data at four neutron instruments in MLF and plan on starting management at other instrument soon. The MLF EXP-DB provides the web portal as the interface for the data management and access. The web portal offering data access for the facility user is under trial use. We will start the data access service with this web portal including the flexible data retrieval next year.

We have also improved the MLF EXP-DB by redesigned to the redundant and scaling-out system. We have a plan for a performance evaluation in the near future.

REFERENCES

[1] T. Nakatani et al., "The Implementation of the Software Framework in J-PARC/MLF" Proceedings of ICALEPCS2009, Kobe, Japan, (2009) p.676.
[2] http://www.i4s.co.jp/rcm/rcmabs.html (in Japanese).
[3] http://www.w3.org/TR/xslt

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

BEAM TRAIL TRACKING AT FERMILAB

Dennis J. Nicklaus, Linden Ralph Carmichael, Richard Neswold, Zongwei Yuan.
Fermilab, Batavia, IL 60510, USA

Abstract

We present a system for acquiring and sorting data from select devices depending on the destination of each particular beam pulse in the Fermilab accelerator chain. The 15 Hz beam that begins in the Fermilab ion source can be directed to a variety of additional accelerators, beam lines, beam dumps, and experiments. We have implemented a data acquisition system that senses the destination of each pulse and reads the appropriate beam intensity devices so that profiles of the beam can be stored and analysed for each type of beam trail. We envision utilizing this data long term to identify trends in the performance of the accelerators.

INTRODUCTION

The Fermilab particle beam starts in its ion source, and from there it can be directed to a wide variety of beamlines and experiments. This paper describes the software created to track the accelerator efficiencies (amount of beam transported without loss) along the various beamlines and for the designated destinations. We use the term trail to designate each of the potential uses and destinations.

Trails can have a wide range of complexity. The simplest trails consist of only one 15Hz Linac beam pulse, and might go to a local beam dump, or be used to study Linac performance. The more complicated pulses involve multiple 15Hz pulses from the Linac, and involve the Linac, Booster, Recycler, and Main Injector accelerators as shown in Fig. 1. Final destinations include various beam dumps along the way, a neutron therapy target, short and long baseline neutrino beamlines, the muon campus experiments, or a test beam area.

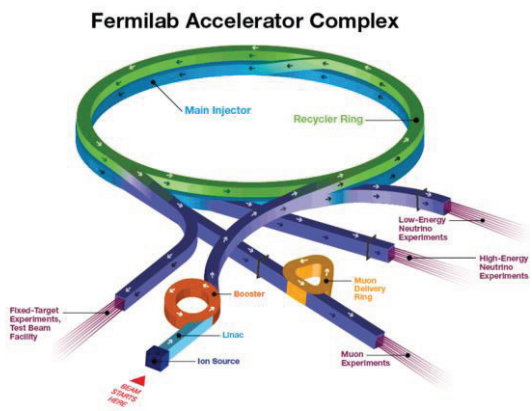


Figure 1: The Fermilab Accelerator Complex.

Collection of these more complex trails is also complicated by the fact that some of the simpler 15Hz pulse trails may occur while the more complicated trail is still underway in the downstream accelerators.

The process starts with the overall beam pulse planning controlled by Fermilab's Sequencer, which coordinates the production of timelines generated by the Time Line Generator (TLG) with other accelerator activities. At the implementation level, a timeline is a series of private bus events (TCLK). Various accelerator components are programmed to respond to these events to send the beam to the desired location. These systems have long been in routine use for accelerator operation.

For this beam trail tracking, we added or enhanced several software programs. To start the process, the TLG now encodes the trail number onto our MDAT data bus when it begins instructions for each beam pulse. The Beam Cycle Coordinator (BCC) decodes that trail number and makes it available to our Acnet control system, in addition to many digital status bits that indicate readiness of various beamlines, components, permits, and interlocks. These status indicators also show whether the accelerator chain was ready and beam pulse had the opportunity to follow the indicated trail, or whether it had to be aborted.

With the above groundwork laid, the main implementation for beam trail tracking begins. The tracking software has three main components:

- The Correlator, which collects data from Acnet-connected devices, groups together readings from the same pulse, and sorts them by trail number.
- The Acnet Formatter, which takes the data from the Correlator, aggregates and re-formats it and makes it available to Acnet.
- The Database Interface, which reads the data over Acnet and stores it into a relational database. Along the way, this also calculates various sums and means, and we have developed a user interface to help retrieve data of interest.

BACKGROUND

Main Injector Ramp Cycle

In order to understand some of the requirements of the trail tracking software, it is useful to have a basic understanding of our Main Injector ramp cycle. The Main Injector accelerates protons from 8 GeV to 120 GeV, taking a little over one second for this acceleration. While it is ramping up and down, multiple batches of protons from the Booster can be injected into the Recycler

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

and stacked into a more intense beam. When the Main Injector has finished the previous ramp cycle, it is injected with the new beam of 8 GeV protons from the Recycler. Overlapping the filling of the Recycler with the ramping of the Main Injector decreases the average full acceleration time for each beam pulse, and has enabled Fermilab to achieve record beam power levels.

Not every 120GeV trail uses the Recycler in this way. We can also inject directly from Booster to the Main Injector.

IMPLEMENTATION

Correlator

The initial data acquisition, the sorting by trail number, and the correlation into the appropriate 15Hz time slots is all done in the Correlator. Most of the data from the Linac and Booster is collected at 15Hz. Some of the other data is only collected upon a particular TCLK event, or on one of several different TCLK events, depending on the trail. We primarily collect beam intensity measurements from a variety of instrument types, although other readings such as the BCC status bits are also included. All data is collected through Fermilab's Acnet control system

Since the readings come from a variety of instruments connected to different front-ends, Acnet doesn't guarantee that the readings will always arrive in the same order, or even that a reading from one front-end will even arrive in the same 15Hz time slice as all the other readings from that same slice. The load on a particular node, or unforeseen network delays can all contribute to this. So in order to collect the maximal amount of data, with as few rejections for tardiness as possible, the Correlator maintains a sliding window of time slices it is currently collecting. When one slice is "full" (meaning all devices for that trail have arrived) or when that slice falls off the end of the sliding window, that 15Hz trail is collected for further processing and the time slice is closed. The Correlator follows the useful rule that collections can't go back in time, so when one slice is declared full, all previous unfilled slices in the window are also closed out for processing and any uncollected devices are noted. Thus the system is somewhat fault tolerant. If one response is dropped by the network, or if a particular front-end is temporarily unresponsive, data collection and processing can still proceed.

Another fault-tolerant adaptation that the Correlator performs relates to errors from devices. If one device replies with some error, then the Correlator will attempt to restart the collection process in hopes that the error was only temporary and has cleared. If a device is repeatedly in error, then the Correlator masks that device off the list of requested devices for all trails. Periodically, data collection is retried in hopes that the error condition has cleared and the device's reading can re-join the data set. Of course, faults in some crucial devices, such as the trail number, cannot be tolerated, and processing has to always wait on those errors to clear before proceeding.

Some of our most common trails will begin at the ion source, go through the Linac and Booster, then into the Recycler for beam stacking, then into the Main Injector for final acceleration, and then to another final destination, such as the long-baseline neutrino beamline. However, as noted above in the explanation of the Main Injector ramp cycle, one trail of beam may still be in the Main Injector while another trail begins assembling in the Recycler. Or while the Main Injector ramp is occurring, another trail through the Booster to the short baseline neutrino beamline may occur. The Correlator has to deal with these overlaps, making sure that the appropriate trails all stay sorted, and dealing with the fact that one Recycler and Main Injector fill will include multiple 15Hz pulses from the Linac-Booster chain. The software uses the term super-trail to describe any trail which includes multiple 15Hz batches.

When the Correlator knows it is finished with any trail or super-trail, they are sent off to the Acnet Formatter for further processing. The end of a super-trail is typically indicated by some TCLK event, which the Correlator has to monitor.

The devices to be read, acquisition frequencies or triggering events, devices needed per trail, and classifications of trails are all driven by a human-readable configuration file. Thus it is straight-forward to change the devices requested for any particular trail.

Acnet Formatter

The Acnet Formatter is much less complicated than the Correlator. As it receives trails from the Correlator, it builds them into a list of trails for each trail number. Periodically (typically on a 10 second interval), for each trail list, the Acnet Formatter reformats and packs the data into structures (arrays) that can be transmitted over Acnet to any requestor. A header is prepended onto the data structure and the structure is built into a documented format.

While performing these collection and reformatting tasks, the Acnet Formatter also creates several diagnostics for each trail, such as number of completely filled vs. unfilled trails, counts of the number of times a device is missing (uncollected) from a trail, which is device is most often missing, which device's reading most often returns with a timestamp outside of the sliding collection window, and the total number of device readings with such a bad arriving timestamp.

All the formatted trail data structures and the diagnostic values are made available as Acnet device readings.

Database Interface

The Database Interface periodically reads the packaged trail data from the Acnet Formatter and stores the readings in a Postgress database. It computes several averages and sums, for instance the sum of each intensity device in a trail over a day. These computed values are also written to database tables, so graphs and displays of more commonly used values can be created more quickly,

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

without having to retrieve all the raw data and compute everything at plotting time.

We have created a graphical interface into the database to enable users to create useful plots of the data. The Java program allows the user to apply multiple selection criteria, including date, device name, summed vs. raw data, and more. This and other Java classes enable us to create daily or weekly beam summaries and statistics.

Figure 2 shows a sample plot of the type that can be made with the database information using the Java interface program. Figure 2 shows the daily summed intensity for a variety of devices along the beam trail and accelerator chain.

PROGRAMMING

The Correlator and Acnet Formatter are implemented in Erlang in our Erlang-based ACSys front-end framework. The Acnet Formatter is one of the frameworks "device drivers", with custom code handling the receipt of trail messages from the correlator and the repackaging of that data into Acnet-accessible structures. The Correlator is implemented as a supporting process to that device driver, being started by the driver and reporting to it. Data collection is done using our standard Erlang-based data collection client, and this project was the heaviest test of that collection client software to date and suggested several refinements of it.

The Database Interface is written in Java and uses our standard Acnet libraries to read data from the beamtrail collection front end. A Java OAC performs the continuous job of reading the assembled trail data and exporting it to the database. Other standalone Java classes and programs provide a graphical user interface to the database data and can create plots with many options from the data.

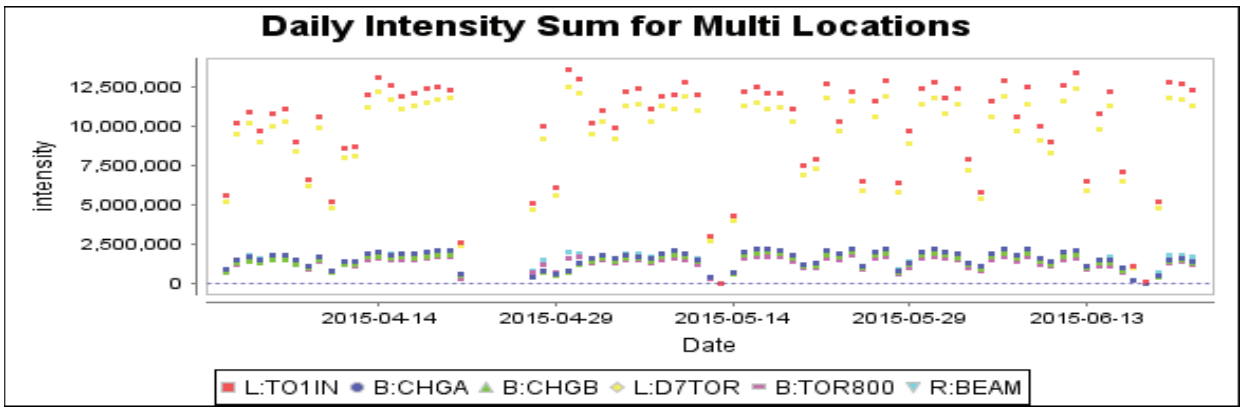


Figure 2: A sample plot showing summed intensities for a variety of device locations.

SUMMARY

We have implemented a set of programs that enhance Fermilab's ability to analyse accelerator performance. We collect beam intensity readings and sort them by the different trail each beam pulse follows through the accelerator and experimental target area chain. The software is able to handle temporary outages by individual reporting devices, and untangles overlapping beam cycles. With the database of information collected, we are able to produce summary plots and reports that provide information about short and long term performance of the accelerators.

ACKNOWLEDGEMENT

Operated by Fermi Research Alliance, LLC under Contract No. De-AC02-07CH11359 with the United States Department of Energy.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PROCESSING HIGH-BANDWIDTH BUNCH-BY-BUNCH OBSERVATION DATA FROM THE RF AND TRANSVERSE DAMPER SYSTEMS OF THE LHC

M. Ojeda, P. Baudrenghien, A.C. Butterworth, J. Galindo, W. Hofle,
T.E. Levens, J.C. Molendijk, D. Valuch, CERN, Geneva, Switzerland
F. Vaga, University of Pavia, Pavia, Italy

Abstract

The radiofrequency and transverse damper feedback systems of the Large Hadron Collider digitize beam phase and position measurements at the bunch repetition rate of 40 MHz. Embedded memory buffers allow a few milliseconds of full rate bunch-by-bunch data to be retrieved over the VME bus for diagnostic purposes, but experience during LHC Run I has shown that for beam studies much longer data records are desirable. A new "observation box" diagnostic system is being developed which parasitically captures data streamed directly out of the feedback hardware into a Linux server through an optical fiber link, and permits processing and buffering of full rate data for around one minute. The system will be connected to an LHC-wide trigger network for detection of beam instabilities, which allows efficient capture of signals from the onset of beam instability events. The data will be made available for analysis by client applications through interfaces which are exposed as standard equipment devices within CERN's controls framework. It is also foreseen to perform online Fourier analysis of transverse position data inside the observation box using GPUs with the aim of extracting betatron tune signals.

SYSTEM OVERVIEW

The "Observation Box" (ObsBox) is a custom acquisition system designed to overcome the bandwidth limits of the VME bus to capture diagnostic data. Figure 1 shows an overview of the system.

The transmitters shown in the figure are typically custom-built VME modules in the beam feedback systems with embedded FPGAs that sample different aspects of the accelerator's beam at a high frequency. In many cases, the acquisition of these data streams for diagnostic purposes can be done through the VME bus into the Linux host residing in the crate. However, some analyses of the beam data can really benefit from a substantially higher data bandwidth than the VME bus can provide.

Typically, the transmitters are close to the beam, in the underground caverns. For accessibility and maintenance reasons, the receiver, a Linux server, is installed on the surface, with a fiber link running between the two.

The new transmitters designed for the ObsBox system stream the data out through the fiber links. At the server end, for each fiber, a CERN-designed PCI Express v1 module is used to act as the receiver: the Simple PCI Express Carrier (SPEC) module [1].

The ObsBox server runs a customized Linux kernel intended for hosting real-time software. A tailored Linux kernel device driver is used to control the SPEC modules over the PCIe bus and is able to stream the data to a user-space process through syscalls and a *sysfs* interface.

The user-space process is responsible for reading the driver streamed data regularly and quickly enough and storing it into very large memory buffers, where the data waits to be requested eventually by clients. Clients connect to the ObsBox server over the CERN's technical network, a standard Ethernet network.

The user-space process is also foreseen to perform online data analysis if the network bandwidth is not enough to stream it out to clients. Since at this stage the data is local, using a coprocessor (like a GPU) to perform the analysis is a viable option, which in turn opens the possibility of running other kinds of analysis for which the CPU may not be performant enough.

TRANSMITTER (VME MODULE)

The Longitudinal Position Measurement VME module (LPM) comprises an analogue front-end and a digital signal processing FPGA platform generating digitized information that can be transmitted via fast digital serial links for further processing. The hardware is based on the beam phase measurement module originally developed for the LHC RF Beam Control [2].

Two signals are acquired in the RF front-end: the beam pickup and the 400.8 MHz RF voltage vector sum of eight cavities. These are demodulated into analog I/Q signals and bunch synchronously sampled by four 14-bit AD converters to give I/Q pairs at a data rate of 40.08 MHz. In the FPGA platform, the VME card is equipped with four gigabit links. The bunch synchronous data of the two I/Q signals (at a rate of 40.08 MHz) is "repackaged" into 50 MHz serial data frames, computing 1 Gbps per link. A FIFO synchronization approach has been used to handle the data crossing among clock domains.

The four data streams are sent over a copper to optical converter card towards the ObsBox. At the receiver side, the four optical links are resynchronized again with 40.08 MHz and the revolution frequency clock. Signal generators are also available within the FPGA platform enabling digital calibration and link debugging.

Similiarly, the LHC transverse damper system (ADT) uses two or four pickups per plane per beam to measure beam position at 40.08 Msamples/s, digitized in a VME beam

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

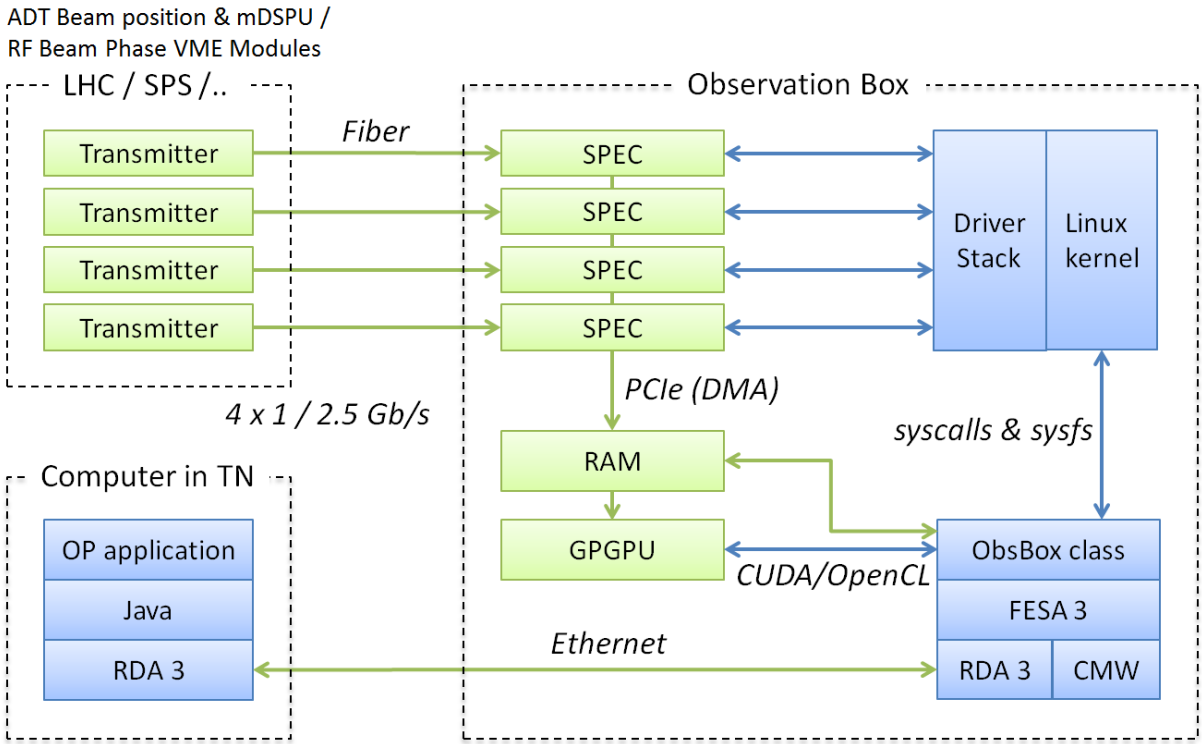


Figure 1: System overview.

position module (BPOS) and streamed to a processing VME module (mDSPU) [3]. Both modules have gigabit links allowing data transfer towards the ObBox using the same format as the LPM..

FIBER LINK

Table 1 describes the data structure transmitted to the receiver over the fiber link in the LHC case.

Since the communication is unidirectional and the receiver has no knowledge on when the transmitter started a block of data (representing one LHC turn), the software in the receiver has to have a way to synchronize with the blocks (frames) being transmitted. In order to do this, each block starts with a fixed constant. On startup or error recovery, the receiver scans the data stream until it recognizes the fixed constant in the stream. It can then acquire the rest of the block and validate its correctness using the CRC. If by chance the fixed constant appeared in the data stream in another position, the CRC will not match, with very high probability. Additionally, other fields can be checked to validate further the block (e.g. checking the block size or the source ID). In case of a CRC-matching failure, the receiver can continue scanning the stream.

The current production system runs each fiber link at 1 Gbps, while it has been designed with a final target speed of 2.5 Gbps, currently in development.

In the deployed LHC use case, the actual payload data rate (after discounting encoding, link synchronization and the data structure overhead), is of approximately 0.6 Gpbs per link. Currently, the production system is running with 4 fiber links per server.

Table 1: Link Data Structure (LHC)

Field	Size [bit]	Description
Fixed constant	32	Arbitrary constant used to identify the beginning of a turn within the data stream
Source ID	32	An identifier of the data being received
Block size	32	Size of the data structure
Turn counter	32	Monotonically increasing, wrapping counter
Tag bits	32	Flags to describe events associated to this turn, e.g. beam instability detected
Reserved	$3 \cdot 32$	Reserved for future use.
Data	$3564 \cdot 16$	Actual payload, which can be arbitrary. In the LHC case, the data is structured in 16-bit words, one per bunch
CRC	32	Used for error detection

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

RECEIVER (LINUX SERVER)

Server

The server to host the ObsBox software and receivers has to be carefully considered. There are several requirements that limit the selection of suitable models from manufacturers.

The primary goal of the ObsBox system is to increase the amount of data available to analysis for end users without a priori filtering or decimation, both in terms of data rate (bunch-by-bunch) and in terms of total time buffered (minutes in the new system compared with milliseconds in the old). Given the 0.6 Gbps of actual payload data per channel, and considering 4 channels per server, the server needs to hold a considerable amount of memory. The current servers used are able to hold up to 6 minutes of bunch-by-bunch data for 4 different channels in a single server, requiring 128 GiB of RAM.

It is desirable to minimize the number of servers due to constraints on the available rack space and other infrastructure considerations. Therefore, each server should be able to manage as many fiber links as possible. This limits the number of motherboards (number of PCI Express slots) and CPUs (processing power) that can be selected.

With these requirements in mind, the current system was deployed using SuperMicro's SuperServer 6028U-TR4+ servers (shown in Figure 2), which have 6 available PCIe slots including one x16 graphics card port.

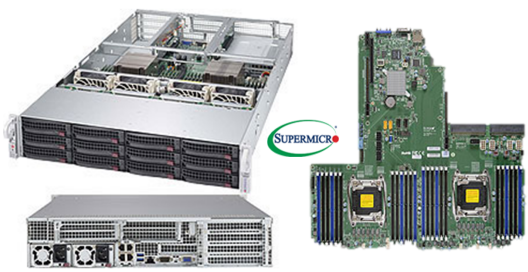


Figure 2: SuperMicro's chassis and motherboard.

SPEC PCIe Module

The module responsible to be the bridge between the optical fiber link and the memory of the Linux host is a codeveloped project between CERN BE-CO and BE-RF groups.

The Simple PCIe FMC carrier module [1] (shown in Figure 3) is a project from the Open Hardware Repository [4], developed by the CERN BE-CO group. It is a simple 4-lane PCIe carrier for a low pin count FPGA Mezzanine Card (VITA 57).

The SPEC's small form-factor pluggable (SFP) receptacle is used to house a fiber optic receiver and a customized FPGA firmware was developed by the CERN RF and Beam Instrumentation groups.

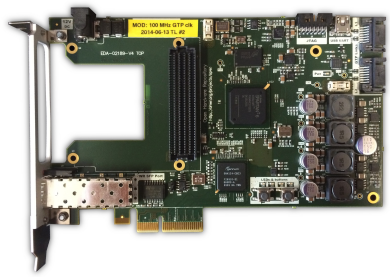


Figure 3: Simple PCIe FMC carrier (SPEC).

Driver

The kernel driver is based on the ZIO framework [5, 6]. ZIO provides a solid framework that simplifies the development of data acquisition drivers. The framework defines the input/output data flows and an user-space interface to access the data. The framework provides buffers, triggers, Linux kernel DMA configuration and it eases the declaration of *sysfs* attributes.

Data access from user-space is completely handled by the framework through char-devices with *mmap* support. Within the ZIO framework, a driver has only to provide support for the hardware access since most of the software logic is done by the framework. In the ObsBox case, the driver is in charge of setting the DMA registers each time some data is ready to be transferred and providing access to the hardware registers for configuration purposes. Everything else is handled by the framework.

User-space Process

The user-space process is the bridge between the clients of the data which will perform the analysis on it and the rest of the system. There are many requirements that the user-space process has to cope with at the same time:

1. Clients may request any recent data at any moment, with different possible filtering criteria. Therefore, the server cannot do any filtering a priori and, thus has to hold all the data until it is considered to have expired. Typically, clients are interested in only a fraction of the streamed data (e.g. selection by bunch, or by time).
2. Depending on the use case, the buffers may be constantly recording new data, or they may start or stop on some events (e.g. beam instability). In addition, the process may have to react to events coming from the CERN timing network.
3. The fact that the driver can only hold a tiny fraction of data in kernel-space before it needs to be released also poses a real-time requirement on the user-space process. The process must regularly and within some deadline read the data from all the channels (devices) from the driver.
4. The process has to expose the data to end users through a typical Ethernet network. The CERN controls framework [7] establishes a standard way of defining, config-

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

uring and controlling all the devices in the accelerator. The communication layer is developed within CERN's Controls Middleware [8].

5. While the currently deployed system is not strictly critical for the beam in the accelerator, availability of the diagnostics may be critical for operators in some scenarios. Therefore, the system has to be robust with high availability.

In order to satisfy all these requirements while keeping the software as simple as possible, the user-space process takes advantage of the Front-End Software Architecture (FESA) framework [9], which provides a comprehensive environment for equipment specialists to design, develop, test and deploy real-time control software for front-end computers.

USE CASES

Longitudinal Diagnostics

The initial use case of the ObsBox for longitudinal diagnostics is the observation of the phase of individual bunches. This is essential to study coupled-bunch instability threshold, and to identify such instabilities if they appear. From the bunch phase, one can also compute the power lost by the bunch. At high beam current the LHC suffers from e-cloud build-up, which can be studied and monitored from the bunch phase measurements [10]. Compared to the previous system which was able to acquire only 73 turns worth of data, the precision of both these measurements will be largely improved due to the very large number of turns now available.

Transverse Dampers

For the transverse dampers in SPS and LHC, habitually called "ADT" in the LHC, it is foreseen to use the ObsBox to record and be able to make triggered acquisitions both of processed pick-up signals and the raw beam position signals.

Both VME modules in the ADT system, the BPOS and the mDSPU, have internal memory which is insufficient to store large amounts of data as needed for injection damping analysis, instability or tune diagnostics. Eventually beam pick-up position, feedback signals and headtail oscillation index [11] can be made available. Due to the large amount of data generated by the system, an intelligent trigger system [12] is also foreseen to freeze long records of data, either on demand, at injection, or automatically, in case of instability detected on the beam. Online data processing is also foreseen to extract machine parameters such as the tune [13], with the possibility to add a GPU co-processor. In a first stage, position information from four pick-up signals (BPMCQ7 in point 4 of LHC [3] of both transverse planes) is being recorded by a single server unit. Following experience with this setup a decision will be taken on adding more signals from the set of currently 40 envisaged. With the recently completed upgrade of the SPS damper to the digital VME standard of LHC, it is also foreseen to implement an ObsBox for the transverse plane in the SPS.

CONCLUSION

The new custom acquisition system is already delivering improved resolution in diagnostics for the LHC accelerator compared to the previous system and is on its way to be used in other accelerators and projects as well.

There are two major milestones in the near future for the project. The first one is to achieve the 2.5 Gpbs target speed on each fiber link, which will make the system even more useful for several projects and with less infrastructure requirements (rack space, fiber installations, etc.).

The second milestone is to compute some expensive analysis in the server itself, using a coprocessor (e.g. GPU). This will open a broad new set of possibilities regarding online analysis which may prove useful for the Operations and Physics teams in the department.

REFERENCES

[1] Simple PCIe FMC carrier (SPEC) website: `http://www.ohwr.org/projects/spec`

[2] P. Baudrenghien et al., "The LHC Low Level RF", EPAC'06, Edinburgh, Scotland, June 2006, TUPCH195 (2006), `http://www.JACoW.org`

[3] D. Valuch et al., "LHC transverse feedback", 5th Evian Workshop on LHC beam operation, Evian-les-Bains, France, 2014

[4] Open Hardware Repository (OHWR) website: `http://www.ohwr.org`

[5] A. Rubini et al., "ZIO: The Ultimate Linux Kernel I/O Framework", ICALEPCS'13, San Francisco, October 2013, MOPB026 (2013), `http://www.JACoW.org`

[6] ZIO Linux device drivers framework website: `http://www.ohwr.org/projects/zio`

[7] Z. Zaharieva et al., "Database Foundation for the Configuration Management of the CERN Accelerator Controls Systems", ICALEPCS'11, Grenoble, France, October 2011, MOMAU004 (2011), `http://www.JACoW.org`

[8] M. Arruat et al., "Middleware Trends and Market Leaders 2011", ICALEPCS'11, Grenoble, France, October 2011, FRBHMULT05 (2011), `http://www.JACoW.org`

[9] M. Arruat et al., "Front-End Software Architecture", ICALEPCS'07, Tennessee, USA, October 2007, WOPA04 (2007), `http://www.JACoW.org`

[10] J. F. Esteban Müller et al., "High-accuracy diagnostic tool for electron cloud observation in the LHC based on synchronous phase measurements", PRST-AB, accepted for publication

[11] W. Hofle et al., "Transverse rigid dipole and intra-bunch oscillation detection using the transverse feedback beam position detection scheme in SPS and LHC", IBIC2015, Melbourne, Australia, 2015

[12] T. Wlostowski et al., "Trigger and RF Distribution Using White Rabbit", WEC3O01, these proceedings, ICALEPCS'15, Melbourne, Australia, 2015

[13] F. Dubouchet et al., "Tune measurement from transverse feedback signals in LHC", IBIC2013, Oxford, UK, 2013

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DEVELOPING HDF5 FOR THE SYNCHROTRON COMMUNITY

N. Rees, Diamond Light Source, Oxfordshire, UK
H. Billich, PSI, Villigen, Switzerland
A. Götz, ESRF, Grenoble, France
Q. Koziol & E. Pourmal, The HDF Group, Champaign, IL, USA
M. Rissi, Dectris AG, Baden, Switzerland
E. Wintersberger DESY, Hamburg, Germany

Abstract

HDF5[1] and NeXus[2] (which normally uses HDF5 as its underlying format) have been widely touted as a standard for storing Photon and Neutron data. They offer many advantages to other common formats and are widely used at many facilities. However, it has been found that the existing implementations of these standards have limited the performance of some recent detector systems. This paper describes how the synchrotron light source community has worked closely with The HDF Group to drive changes to the HDF5 software to make it more suitable for their environment. This includes developments managed by a detector manufacturer (Dectris - for direct chunk writes) as well as synchrotrons (DESY, ESRF and Diamond - for pluggable filters, Single Writer/Multiple Reader and Virtual Data Sets).

INTRODUCTION

The original initiative for this work came from a workshop held at Paul Scherrer Institute (PSI) in late May 2012. This was jointly organised by PSI and Dectris and brought together representatives from the synchrotron community and employees of The HDF Group. A number of issues were identified at that workshop, including:

1. The single threaded nature of the HDF5 library made it difficult to benefit from the multicore architecture of modern CPU's.
2. Compression was an important mechanism to improve data throughput in many synchrotron use cases, but including any form of compression limited the throughput of the library and also could not be used when writing in parallel with the Parallel HDF5 library (pHDF5).
3. Use of a hierarchical format like HDF5 led to the need to store many detector frames in one file. However, this increases the latency of any processing because HDF5 files could not be open for read and write simultaneously.

Since this meeting the authors have collaborated to fund a number HDF5 developments that have eliminated many of these problems for practical synchrotron use cases.

DIRECT CHUNK WRITES

This development was sponsored by Dectris and PSI and introduced a new function `H5DOwrite_chunk` in the high-level HDF5 library which bypasses the data gathering and scattering, data conversion, filter pipeline, and chunk cache and writes the data chunk to the file directly (see Fig. 1). This allows the user program to compress the data outside the library – potentially using parallel algorithms or hardware accelerators. It also avoids a number of data copies, which limits any dataflow through the filter pipeline to ~500 MB/sec on typical processors.

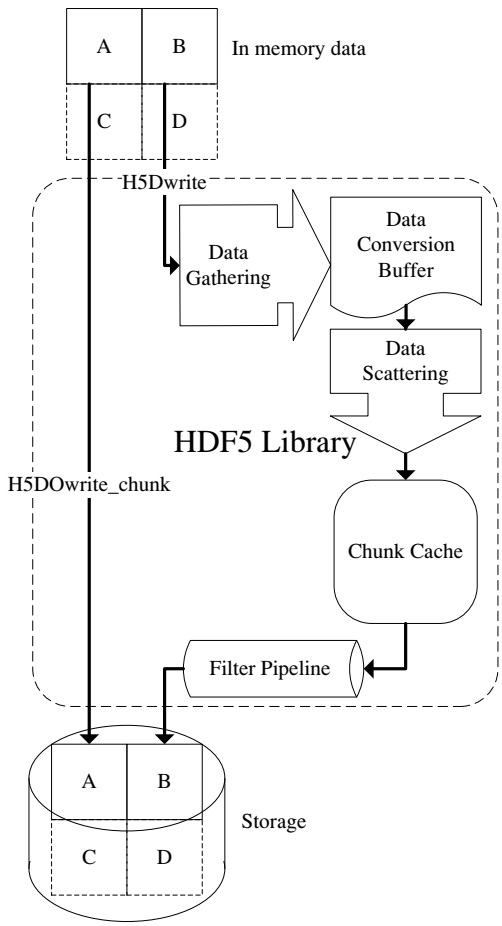


Figure 1: HDF5 data writing, showing the elements that `H5DOwrite_chunk` bypasses.

The feature was released in HDF5 version 1.8.11 in May 2013. The implementation and testing required about 5 months of work and, while the function did not present any technical difficulties, fitting it into the HDF5 library required some rethinking of the HDF5 source code architecture. The latest versions of the library are set for

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

expansion with other optimized functions if the need arises.

DYNAMICALLY LOADED FILTERS

This development was sponsored by DESY. As mentioned above, HDF5 implements data compression by means of a filter pipeline. The library comes with several predefined filters including GZIP and SZIP compression algorithms [3] but SZIP, for example, has license terms that may restrict data distribution. Those algorithms are also very general and so may not achieve optimal performance in terms of speed and/or compression ratio for a specific experimental application. Also prior to this development custom compression methods had to be incorporated into the main HDF5 library at link time and this further limited the way filters could be used.

With the new feature, the filters can be dynamically loaded at run-time which allows application domains to develop specialist filters suited to their requirements without the need for integrating them within the HDF5 code base or passing support to The HDF Group. When a filter DLL or a shared library is available on the system, HDF5 tools and user applications can use an off-the-shelf HDF5 library to read and modify data transparently to a user's application. Each filter has a unique identifier and filter developers are encouraged to register their filters with The HDF Group to avoid identifier clashes and allow a list of available filters to be published (see https://www.hdfgroup.org/services/contributions.html).

Writing, testing and distributing custom HDF5 filters is now a community effort with The HDF Group just providing guidance. The HDF Group, DESY and Dectris are working on a repository of the dynamically loaded filters (https://svn.hdfgroup.uiuc.edu/hdf5_plugins/), so users can download code and binaries for filters they need.

SINGLE WRITER/MULTIPLE READER

With the transition from less structured data to a structured data format that is related to the goals of an experiment comes the requirement to store all data for an experimental scan (or even experiment) in a single file. This groups like data together and makes data storage more efficient at high frame rates – where in the past each frame and its metadata had been stored in separate files. However, since HDF5 did not allow simultaneous reading and writing by separate threads or processes, this either meant that any processing of the file had to wait until the scan completed, which in extreme cases can be several hours, or that data servers had to be written to handle all the data reading and writing, creating a data bottleneck.

Single Writer/Multiple Reader (SWMR) allows a single process to update datasets in an HDF5 file whilst multiple other processes are reading data from the same datasets. To solve the difficulties involved with allowing concurrent access to write and read processes, HDF5 developers used an approach that is lock-free and that does not require any communication between processes. All communications are done through the HDF5 file itself.

In previous versions of HDF5 the file organisation on disk was only guaranteed to be consistent if it wasn't open for writing because data storage metadata (that describes how the data is laid out on disk) was kept in the writers cache until the file was closed. The approach taken for the SWMR development was to modify the writer's metadata cache and file data structure code to ensure that metadata that is referred to by another metadata object is always flushed from the cache first. This ensures that a reader will never attempt to resolve an invalid offset and has the secondary benefit that the state of the file on disk is always consistent and cannot be corrupted by an application crash. The parent-child relationship between the metadata objects is called a flush dependency.

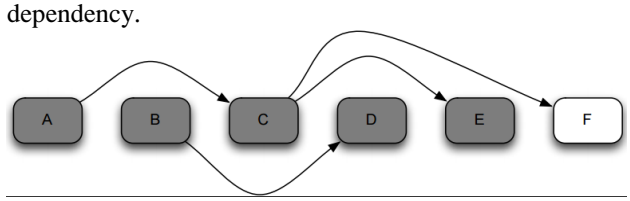


Figure 2: Representative flush dependencies between metadata cache objects.

An example of the flush dependency algorithm is shown in Fig. 2 where entry A is the parent of C, B is the parent of D, and C is the parent of both E and F. Dirty entries are shaded (A-E), and clean entries are not shaded (F). With this configuration, A could not be written to the file until C is clean, and C could not be written to the file until both E and F are clean. Likewise, B could not be written to the file until D was clean. So, with this configuration, when the metadata cache is attempting to flush dirty entries to the file (perhaps when flushing all the metadata for the file), either entries D and E could be written to the file first (making them clean), then entries B and C, and finally entry A. If C was written before B, A would also be able to be flushed before B.

This algorithm is based on the assumption that the HDF5 file resides on a file system compliant with POSIX I/O semantics. There are two major features of POSIX I/O semantics that are critical for SWMR use:

- POSIX I/O writes must be performed in a sequentially consistent manner.
- Writes to the file must appear as atomic operations to any readers that access the file during the write operation.

These semantics apply to any processes that access the file from any location. The caveat is that some file-systems (notably NFSv3) do not abide by these semantics and they cannot be used since reading processes do not see changes to the data in the same order as it is written.

The feature presented significant technical difficulties and required a lot of rework of the HDF5 library architecture. It also triggered changes to the HDF5 File Format introducing new chunk indexing structures with enabled checksums on the metadata items.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The feature leveraged the work done by The HDF Group developers for another commercial customer who funded the effort for several years.

The completion of the SWMR development for "append-data" only writers [4] was sponsored by Diamond Light Source, ESRF and Dectris. The development was coordinated, but each organisation contributed to a different work package of the development and so funding was kept separate.

The development comprised about twelve months of the effort and was delivered as a specialized version of the HDF5 library to DLS, ESRF and Dectris. The HDF Group has been working on merging the feature with the main stream of HDF5 and delivering it in the HDF5 release version 1.10.0.

VIRTUAL DATA SETS

The virtual dataset concept has the most interesting history in that it was developed last and has evolved the most from the original ideas in the PSI workshop in 2012. The original driver was storage of data from high-speed detectors. Two approaches were considered to maximise the storage throughput – one was compression (for example, the Pilatus detector used the imgCIF packed compression) and the other was writing data in parallel. The problem was that pHDF5 was not able to write compressed data, and so a proposal called "Parallel Compressed Writer" was generated. As the name implied, one of the initial design ideas was to allow compression in pHDF5, but the complexities of this was a problem and it was also discovered that the performance per process of pHDF5 was not as good as single stream HDF5 because of the MPI overhead on the nodes used for writing.

After some discussion another approach was proposed which extended the concept of the HDF5 dataset soft-link to allow for a dataset to be comprised of a union of datasets from a number of other files. These files could then be written independently and in parallel and since the writers wouldn't use pHDF5 they could also be compressed. When The HDF Group saw this proposal they connected it with use cases from other domains far beyond just high-speed detectors. Consequently, the idea was generalised so that the parent dataset could be composed of an arbitrary mapping of child datasets, and the name of "Virtual Dataset" was coined [5].

Figure 3 shows a simple mapping of source datasets to a virtual dataset, but much more complex mappings are possible. The mappings can be any n-dimensional rectangle and gaps can be filled with user specified fill values. These gaps can be both gaps between the maps or, in a data acquisition application, gaps representing data that hasn't yet been written to the underlying datasets. Hence virtual datasets can be used for sparse data, or for combining data from different sources to be used by an application that wasn't considered when the original files were written. Virtual dataset capabilities are particularly useful in an environment where multiple data sources need to update an HDF5 dataset simultaneously. The child files can all be written by different SWMR processes and readers can still open and read the parent dataset without conflict, and periodically poll the dataset metadata to determine if additional data has appeared.

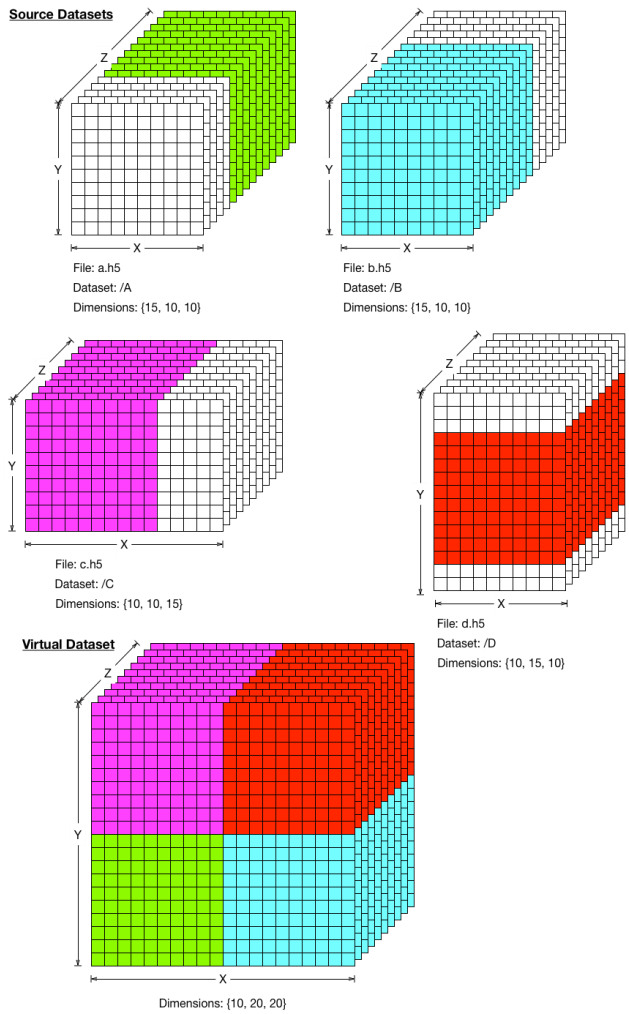


Figure 3: Simple mapping of source datasets to a VDS.

The feature implementation required adding a new storage layout to HDF5 and triggered improvements to the HDF5 library code. The total implementation effort took seven months of the developers' time. Another two months will be spent on merging VDS with the HDF5 mainstream library and preparing the feature for the HDF5 1.10.0 release.

The Virtual Data Set project was jointly funded by DESY, Diamond and the Percival detector project. Since the project could not be broken down into appropriate sized work packages, Diamond Light Source coordinated the work and all three parties contributed equal amounts.

CONCLUSIONS

The main point of this paper is to demonstrate how some of the major synchrotron light sources have collaborated to develop our common software for the benefit not just of the stakeholders, but for the entire

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

community. These developments were different to many of the collaborative software developments we have undertaken in the past (e.g. EPICS and TANGO) in that all the development was done commercially rather than in-house. This has created challenges because money rather than manpower had to be committed and this has required a strong motivation and champion for the development. However, it has also proved to be productive because the developments have been delivered largely to time and to the agreed budget.

The largest time delay has come for the latter two projects because they have involved a change in the underlying HDF5 data file format. This has meant that they cannot be released as an HDF5 point release, but have had to be delayed until the next major HDF5 release (HDF5 1.10). This has slowed the uptake of the developments and we look forward to the release of HDF5 1.10 late in the first quarter of 2016.

REFERENCES

[1] The HDF Group. Hierarchical Data Format, version 5, 1997-2015. http://www.hdfgroup.org/HDF5/

[2] NeXus. A coomon data format for neutrons, x-ray and muon sources. http://www.nexusformat.org

[3] The HDF Group. “Using compression in HDF5”, https://www.hdfgroup.org/HDF5/faq/compression.html

[4] The HDF Group. “HDF5 New Features”, https://www.hdfgroup.org/HDF5/docNewFeatures/

[5] The HDF Group. “HDF5 Virtual Dataset”, https://confluence.hdfgroup.org/display/UF/HDF5+Virtual+Dataset

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ILLUSTRATE THE FLOW OF MONITORING DATA THROUGH THE MeerKAT TELESCOPE CONTROL SOFTWARE

M. Slabber*, SKA SA, Cape Town, South Africa
M.T. Ockards†, SKA SA, Cape Town, South Africa

Abstract

The MeerKAT telescope [1], under construction in South Africa, is comprised of a large set of elements. The elements expose various sensors to the Control and Monitoring (CAM) system, and the sampling strategy set by CAM per sensor varies from several samples a second to infrequent updates. This creates a substantial volume of sensor data that needs to be stored and made available for analysis. We depict the flow of sensor data through the CAM system, showing the various memory buffers, temporary disk storage and mechanisms to permanently store the data in HDF5 format on the network attached storage (NAS).

ELEMENTS THAT MAKE UP CAM

Sensor

Sensors are either physical sensors on devices, aggregated/calculated or internal metrics from software components. Some examples of physical sensors are tilt, temperature, wind speed and motor rpm. It is estimated that once MeerKAT is completed there will be close to 80 000 sensors in the system.

Device

Devices are the hardware that constitute the telescope. Devices are engineered to have Ethernet interfaces and can communicate via Karoo Array Telescope Communication Protocol (KATCP). Where devices do not have KATCP interfaces, translators are created [2]. The devices expose sensors and requests on the KATCP interface [3].

Component

CAM consists of many distinct components. Components are the standalone building blocks of the CAM system. Interaction with components are done with the KATCP protocol. The components expose sensors and requests over KATCP.

Node

Nodes are virtual containers on the host hypervisor [4]. The MeerKAT nodes run Ubuntu 14.04 LTS Linux operating system. When MeerKAT is fully deployed, it is expected to have 13 nodes.

Proxy

Proxy components make sensors and requests on devices available to the CAM system. Sensors and requests from the devices can be discarded or even rolled up into new sensors and requests on the proxy component. The proxy component also exposes sensors and requests of its own.

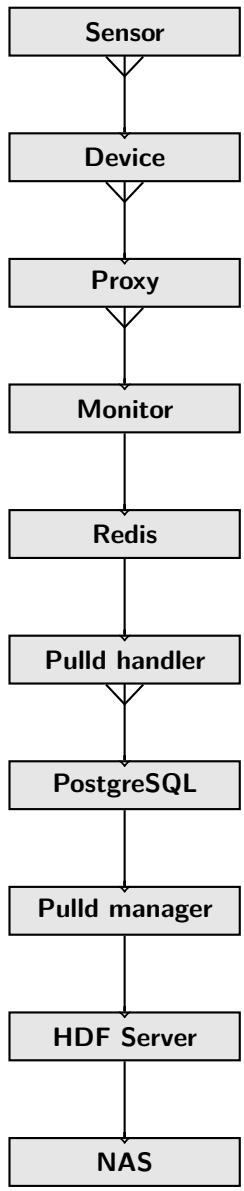


Figure 1: The progression of storing a sample, from sensor to archive.

Controller

Controller components connect to other subsystems. As with a proxy component, sensors and requests from the subsystem can be exposed or rolled up into new sensors and requests. Controllers always use KATCP to communicate to

* martin@ska.ac.za
† tockards@ska.ac.za

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

the subsystem. The subsystem is responsible for presenting a central interface to which CAM can connect in order to gather sensor samples and use requests to initiate action. Examples of subsystems are the Correlator Beam Former and Science Data Processing.

Monitor

On each node in the CAM system a monitor component gathers sensor samples at a configured sampling rate from the other components on the same node. The gathered samples are written to a memory buffer on the node.

Redis

Redis is used as a memory buffer on each of the nodes in the CAM system. Samples are written to the memory buffer by the monitoring component.

Pulld

Pulld component on the storage node that moves the samples from the memory buffers to the central database. A sub-process of Pulld, Pulld handler, is started for each of the nodes in the system. Pulld will archive samples from the storage node and send it to the HDF Server to be archived. The archiving is done in the Pulld manager on a scheduled basis.

PostgreSQL

PostgreSQL is used as the central database for sensor samples and sensor metadata [5]. Samples are stored in the central database until they are archived to files on the network attached storage (NAS). The archived data is presented in the database as a foreign table; a foreign data wrapper (FDW) was developed to achieve this. Queries to the archive table are translated by the FDW and sent to the HDF Server; the response from the HDF Server is translated for presentation in the database. The FDW was developed to provide a read only interface to the HDF Server.

HDF Server

This component is responsible for performing all the read and write operations on the archive files. The archive files are HDF5 formatted files stored on the NAS. The files are stored in a hierarchical directory structure. An archive file is created for each component of the system per day. e.g. For *subarray1* on 7 March 2015 the file `2015/03/07/2015-03-07_subarray1.h5` will be created.

NAS

The NAS is a standalone storage unit attached to the network. An export on the NAS is mounted on the storage node using network file system (NFS). The NAS is accessible over a 10Gbps Ethernet network.

Query Interface

Two query interfaces were developed to provide API's for accessing historical sensor data. The first uses KATCP as the access protocol and runs on the storage node, this interface is used by other CAM components. The second interface was developed as part of the portal system and provides an HTTP interface for the Graphical User Interface (GUI) [6] and external systems. Both query interfaces use the PostgreSQL database for all queries.

SENSOR SAMPLE

It is important to understand what a sample is in order to comprehend how the samples are transported, stored and made available for queries. Each sensor reading processed by the CAM system is called a sample. A sample always has a sensor name, sample timestamp, value timestamp, status and a value.

SENSOR_NAME A normalised form of the sensor's name. Non-alphanumeric characters are replaced by the '_' character; the case of alphabetic characters is maintained. The original KATCP name of the sensor is maintained as an attribute of the sensor with the sensor meta data.

SAMPLE_TS The timestamp of when the sample was first processed by the CAM system. It is a UNIX timestamp, the time in seconds since the epoch of 1 January 1970 00:00 UTC. SAMPLE_TS is always unique for a sensor and presented as a floating point number, the number of decimal places (fraction of a second) used is dependent on the host operating system.

VALUE_TS A UNIX timestamp when the acquisition was performed. VALUE_TS and SAMPLE_TS are the same or within milliseconds of one another if the acquisition was performed in sync with the CAM sampling strategy. When a sensor is over-sampled it is possible to have the same consecutive VALUE_TS, STATUS and VALUE.

STATUS An indication of the state of the sensor when the sample was taken. Can have a value of UNKNOWN, WARN, NOMINAL, FAILURE, INACTIVE, ERROR, UNREACHABLE as defined in the KATCP [3] specifications.

VALUE The value of the sensor at the time of acquisition (VALUE_TS). The data type of VALUE is typically one of float, integer, string, or boolean; additional types are allowed by the KATCP specifications but they are less commonly used. It was decided that the storage system will treat all values as strings until the archiving is performed. In the archive, data is stored as float, integer, boolean or string; all other types are stored as strings in the archived files.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SAMPLING STRATEGIES

CAM inherits the sampling strategies of the KATCP protocol. A client connected to the KATCP interface on a device or a component (the server) can set a sampling strategy for the sensors on the server; sensor samples will be sent by the server at the requested sampling rate. Several sampling strategies are available in KATCP. KATCP defines the strategies `none`, `auto`, `period`, `event`, `differential`, `event-rate` and `differential-rate`. Examples are `<period P>` the value is reported every P seconds, `<event>` the value is reported when it changes and `<event-rate SP LP>` the value is reported when it changes or every LP seconds. For the full details on sampling strategies see the KATCP specifications [3].

The sampling strategy used by the monitor component can be set per sensor in the central configuration. Sensors that do not have a configured sampling strategy are sampled every 10 seconds; for sensors of boolean and string types `event-rate 0 600` is used.

PROGRESSION THROUGH THE STORAGE SYSTEM

The following key words are used in Figures 2, 3 and 4.

- **Monitor** component has an instance running on every node.
- **Redis** memory database is installed on every node.
- **Pulld handler** has an instance on the storage node for every node in the system.
- **Pulld manager** has one instance on the storage node.
- **DB** is the central database on the storage node.
- **HDF Server** is running only on the storage node.
- **NAS** is the file storage system made available over the network.
- **Query** interface has instances on the storage node and the portal node.

Monitoring Component to Central Database

The monitor process connects to all the CAM services on the node on which it is running . Based on configuration, the monitor process subscribes to sensors and receives updates based on the requested sampling strategy. In Figure 2 (it is shown) for every new update the monitor gets from a sensor, the sample is written to the memory database (1). The memory database is a buffer, it allows for very fast writes and prevents blocking the monitor process. Because of the memory buffer, Pulld does not need any knowledge of the write rate of the monitor process. Pulld queries (2) the memory database and writes the received samples (3) to the database (4) on the storage node. When the write to the database on the storage node is successful (5) the samples are deleted (6) from the memory database on the relevant node.

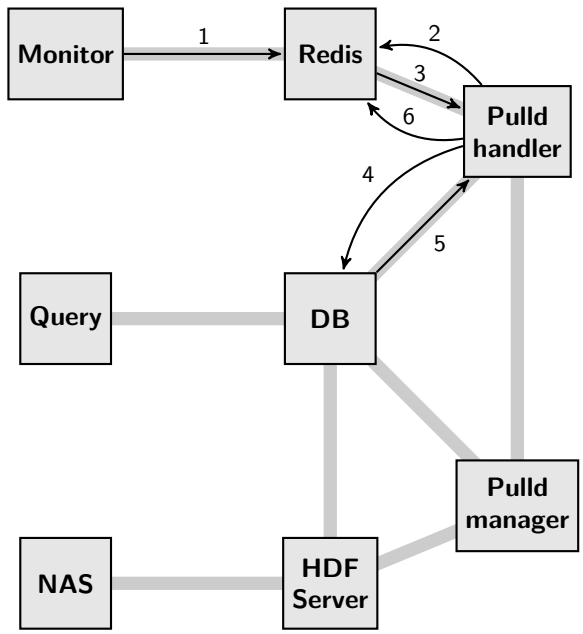


Figure 2: Moving samples from monitoring component to the central database.

Central Database to the Archive

Based on configuration, Pulld will start an archive task at a scheduled time (Fig. 3). The archive task will determine what data should be archived (1); based on a configurable sample age parameter. Pulld will retrieve data (2) from the database and write this data to the HDF Server (3). When the HDF Server acknowledges (5) that data has been successfully written to the NAS (4), the data will be deleted from the database (6).

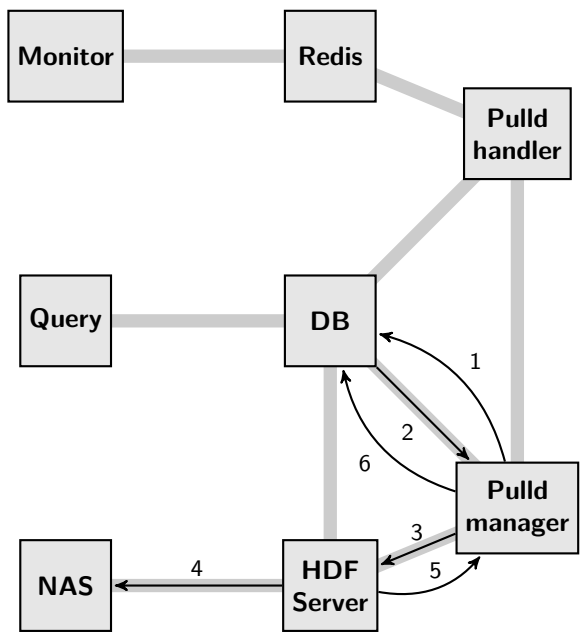


Figure 3: Archiving data from storage database to NAS.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Query the Samples

Requests from other components for historical sensor samples are done against the query interface (Fig. 4). The query interface is connected to the database and will create the appropriate SQL query and run the SQL against the database (1). If the query requires archived data, the database has a foreign table that contains no data, but provides an interface to the HDF server. It uses the FDW to achieve this. The FDW will rewrite the query into a syntax that can be interpreted by the HDF Server (2). The HDF Server will read the appropriate files from the NAS (3) and return the samples to the database (4). The database will add the returned samples to the response and return (5) to the query interface.

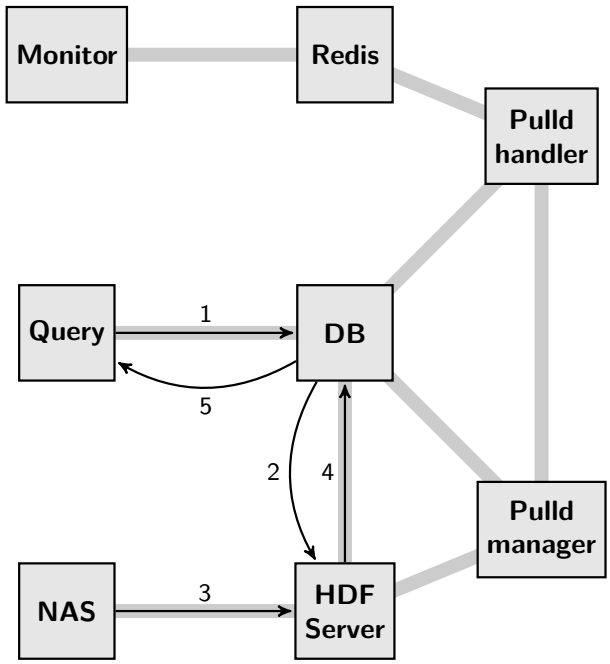


Figure 4: Query data across storage database and archive on NAS.

REFERENCES

[1] R.S. Booth, W.J.G. de Blok, J.L. Jonas, and B. Fanaroff, "MeerKAT Key Project Science, Specifications, and Proposals," *ArXiv e-prints*, pp. 1–16, 2009. [Online]. Available: http://arxiv.org/abs/0910.2935

[2] L. van den Heever, "MeerKAT Control And Monitoring - Design Concepts and Status," in *Proceedings of ICALEPC 2013*, paper MOCOAAB06, 2013.

[3] S. Cross, R. Crida, T. Bennett, M. Welz, and T. Kusel, "Guidelines for Communication with Devices," 2012. [Online]. Available: http://pythonhosted.org/katcp/_downloads/NRF-KAT7-6.0-IFCE-002-Rev5.pdf

[4] N. Marais, "Virtualization and deployment management for the KAT-7 / MeerKAT control and monitoring system," in *Proceedings of ICALEPC 2013*, paper THCOBA06, 2013.

[5] M. Slabber, "Overview of the monitoring data archive used on the MeerKAT telescope," presented at ICALEPCS'15, Melbourne, Australia, paper THHD3O06, *these proceedings*, 2015.

[6] M. Alberts and F. Joubert, "The MeerKAT graphical user interface technology stack," presented at ICALEPCS'15, Melbourne, Australia, paper THHC3O01, *these proceedings*, 2015.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A SYSTEMATIC MEASUREMENT ANALYZER FOR LHC OPERATIONAL DATA

G. Valentino*, X. Buffat, D. Kirchner, S. Redaelli, CERN, Geneva, Switzerland

Abstract

The CERN Accelerator Logging Service stores data from hundreds of thousands of parameters and measurements, mostly from the Large Hadron Collider (LHC). The systematic measurement analyzer is a Java-based tool that is used to visualize and analyze various beam measurement data over multiple fills and time intervals during the operational cycle, such as ramp or squeeze. Statistical analysis and various manipulations of data are possible, including correlation with several machine parameters such as β^* and energy. Examples of analyses performed include checks of collimator positions, beam losses throughout the cycle and tune stability during the squeeze which is then used for feed-forward purposes.

INTRODUCTION

The CERN Large Hadron Collider (LHC) was designed and built to accelerate two counter-rotating beams to an energy of 7 TeV and deliver high energy collisions of protons or heavy ions in the four experiments, namely ATLAS, ALICE, CMS and LHCb [1]. Data from thousands of devices in the CERN accelerator infrastructure, amounting to around 1 million signals and 50 TB/year are stored by the CERN accelerator logging service (CALS) [2].

An overview of the software architecture is shown in Fig. 1. Two Oracle databases are used for storage. The Measurement database (MDB) is used to store raw data for a short period of time, which is then filtered and transferred to the Logging database (LDB) for indefinite storage of filtered data. The databases subscribe to specific properties of each device via the Java API for Parameter Control (JAPC) [3] or PVSS for the industrial SCADA systems.

A Java GUI called TIMBER is provided to visualize and extract the logged data. However, in order to assess the stability of the machine and perform analyses over several fills and machine modes, a tool called Systematic Measurement Analyzer was developed. This paper describes the implementation of the tool as well as some typical use case scenarios where it has proved to be useful.

LHC MACHINE CYCLE

The operation of the LHC follows well-established stages, which together form a machine cycle. A typical LHC machine cycle is shown in Fig. 2. At the injection stage (1), the LHC receives two beams from the Super Proton Synchrotron (SPS) at an energy of 450 GeV per beam. The beams arrive in bunch trains, which may consist of 1-144 bunches, depending on whether the machine is to be filled for beam tests or physics. When the filling procedure is completed, both beams are ramped up (2) until the desired energy is reached. At flat top (3), the machine operators initiate the beam squeeze procedure (4), in which corrector magnets are used to shrink the beam size at the experimental insertion points (IPs) to achieve the desired β^* (the β-function at the experimental IPs).

Up to this point, the beams are separated by several σ (r.m.s. beam size) in all IPs. Hence, the final step is to collapse the separation orbit bumps and bring the beams into collisions. The operational state known as stable beams (5) is declared, and the experiments begin taking data. The beams may be extracted or dumped (6) at any point during the fill due to operational requirements, equipment failures, beam instabilities or operator mistakes. When this happens, the machine is ramped down (7) to injection energy in preparation for the next fill. The data presented in Fig. 2 are taken from fill number 3131, in which a total of 1374 bunches were present in the machine, and $130 \times 130 \mu b^{-1}$ of luminosity was delivered to the ATLAS and CMS experiments in 3.5 hours of stable beams.

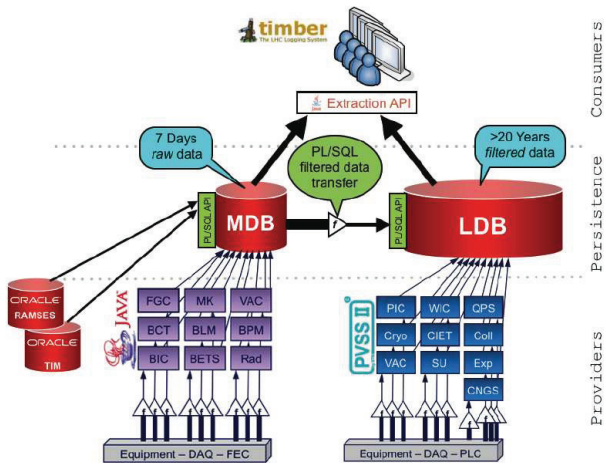


Figure 1: The CERN accelerator logging service architecture [2].

SOFTWARE ARCHITECTURE AND UI

The systematic measurement analyzer is written in Java, which is the standard for LHC operational applications run-

* gianluca.valentino@cern.ch

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

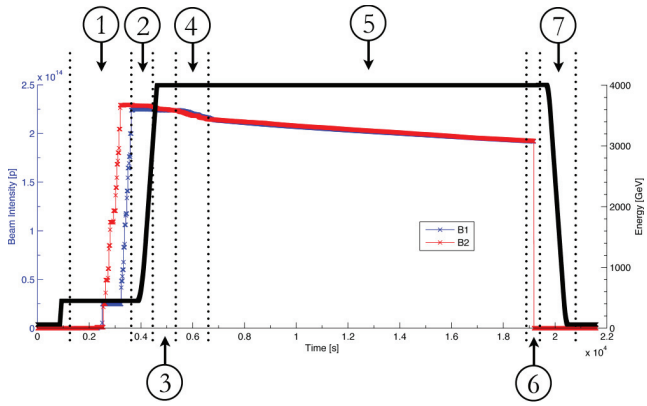


Figure 2: The LHC machine cycle, illustrated by the beam energy and beam intensities taken from fill number 3131 (t[0] = 05.10.2012 01:00:00). There are seven stages: 1) injection; 2) ramp; 3) flat top; 4) squeeze; 5) stable beams; 6) beam dump; 7) ramp down.

ning in the CERN Control Center. It was initially developed to optimize the squeeze and for feed-forward purposes [4]. A flowchart of the steps needed to extract and analyze the data is shown in Fig. 3. The user can input a set of predicates to filter which fills are to be used for the data extraction. These include the beam mode, beam energy, and fill duration. The latter is useful for instance if one needs to look at stable beams fills of sufficient length to make conclusions from the data [5]. The fills are selected based on a time window or by fill number.

The next step is to select the logging variables, which can be in numeric or vectornumeric format, in which case an index needs to be provided. The selection can be done via regular expressions, and simple operations (addition, subtraction, multiplication and division) can be applied on two or more variables. A list of pre-defined, frequently used variable names is available with human-readable names. The default abscissa for the subsequent charts is time, but one logging variable can also be plotted as a function of another. The data are extracted for the selected fills from the LDB via a Java API provided by CALS.

Plots can be generated for a specific interval, such as during the ramp or for the whole fill. It is possible to post-process the data, for example to synchronize all data sets to the start or end point of a beam mode, or perform scaling operations. Histograms can be made for a set of pre-defined analyses, such as the intensity transmission in ramp or squeeze, or the beam mode duration for one or both beams. The third category of plots provides an overview of the reproducibility of the data for a particular variable from one fill to another.

A screenshot of the user interface showing the main components is shown in Fig. 4. The predicate and fill selection panels are on the left hand side. The time spent in each machine mode can be viewed for each fill individually. The central part is used to display the plots, while the plotting operations can be accessed from the bottom panel. These

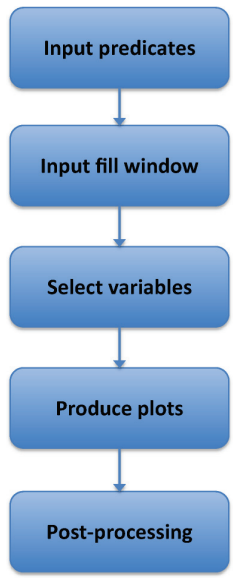


Figure 3: Analyzer usage flowchart to extract and visualize the data.

including adding a new chart or a new line to an existing chart. Post-processing and dataset synchronization with a given machine mode can be performed from here. The data can be exported from the chart to a text file for additional analyses.

USE CASES

Collimation System Performance in Stable Beams

The LHC collimation system [6] protects the machine agains damage due to beam losses. Almost 100 collimators, each having two jaws form a four-stage hierarchy to intercept, scatter and absorb highly energetic halo particles. Beam losses at specific locations in the LHC are measured by 3600 Beam Loss Monitors (BLMs). The cleaning inefficiency of the collimation system can be defined as the leakage of primary particles from the collimation hierarchy to the dispersion suppressor (DS) in insertion region (IR) 7, where the betatron collimation system is installed. It can be derived from the ratio of the BLM signal at one of the superconducting magnets in the start of the IR7 DS to the losses at the primary collimator, which is closest to the beam and normally has the highest losses. Figure 5 shows the ratio of the two signals over several fills during stable beams in 2015.

Collimator LVDT - Magnet Current Correlation

Linear Variable Differential Transformers (LVDTs) provide an independent read-out of the LHC collimator jaw positions. During Run 1, the LVDT readings at some collimators were found to be susceptible to electromagnetic interference from nearby magnets [7]. The plot in Fig. 6 shows the correlation of the difference in the collimator

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

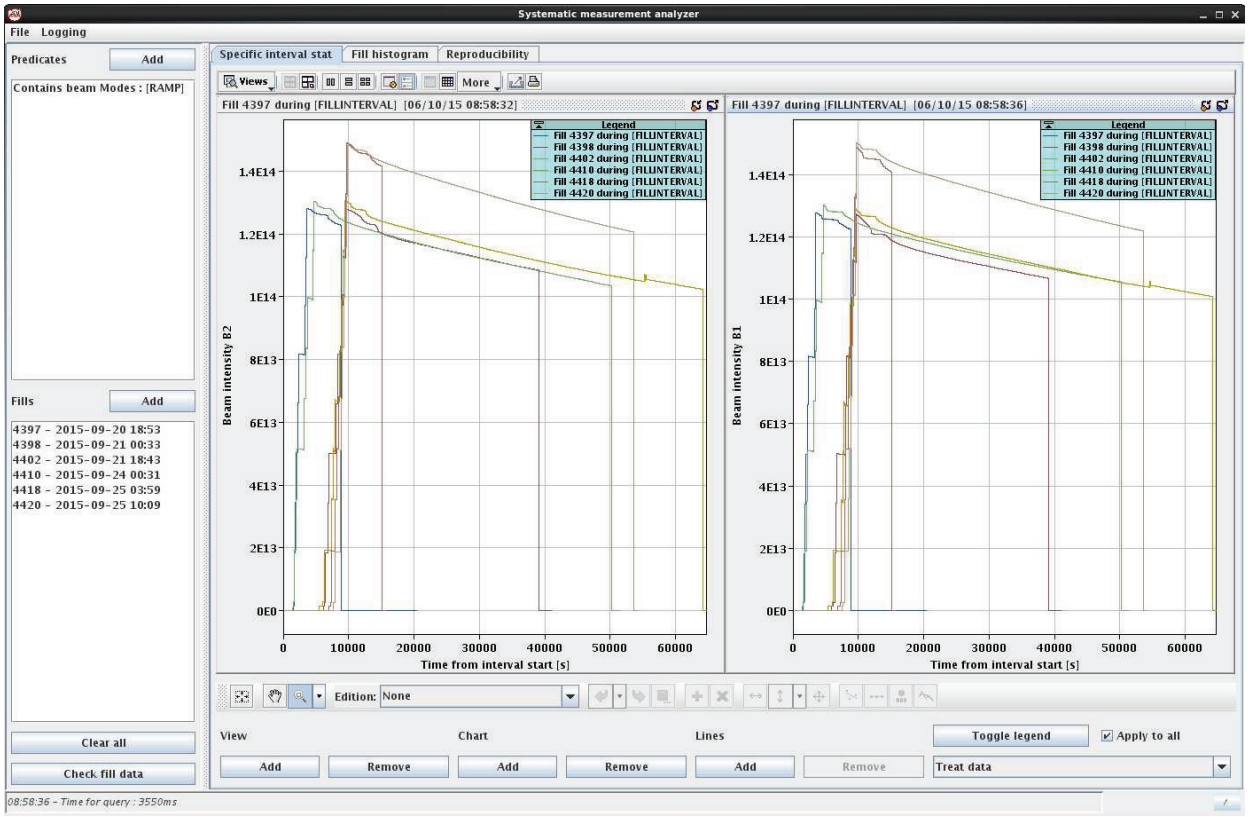


Figure 4: Screenshot of the analyzer user interface, showing the beam intensity evolution in several fills in B1 and B2.

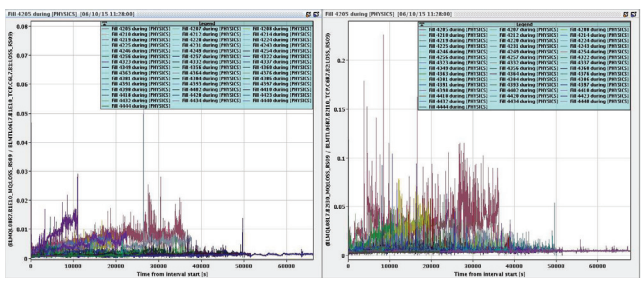


Figure 5: Collimation cleaning inefficiency in stable beams.

gap measured by the gap LVDT and calculated from the left and right motor positions, as a function of the magnet current during several LHC pre-cycles. An offset of 200 μm is reached at maximum current. This led to several of the worst-affected LVDTs to be replaced by a new design (I2PS [8]), which is not affected by this interference.

Beam-based Optimization of the Squeeze

The betatron squeeze is a critical operational phase of the LHC due to the reduced aperture margins in the superconducting triplets at top energy. In order to reduce the beam losses and therefore improve the intensity transmission during the squeeze, a fill-to-fill analysis was made to determine the reproducibility of the main beam parameters such

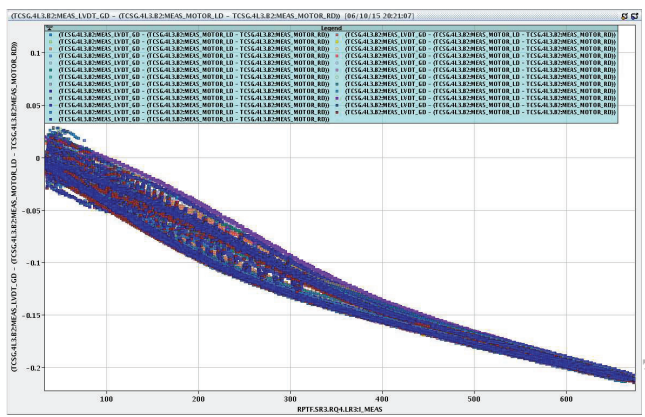


Figure 6: Correlation of the collimator jaw gap motor-LVDT offset and the current in a nearby magnet.

as the tune and orbit [9]. The very good reproducibility allowed the possibility to apply feed-forward corrections, hence minimizing the trims made by the real-time feedback systems for the tune and orbit during the squeeze [4]. This has the effect of reducing the possibility for the beams to be lost in case of trips of the real-time feedback systems.

Beam Mode Duration Statistics

The systematic measurement analyzer can produce bar charts to visualize the time spent in any or all of the 18 LHC

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

machine modes in each fill. This is useful to determine the operational efficiency and turnaround time of the LHC from one physics fill to the next. The plot in Fig. 7 is for all fills in September 2015.

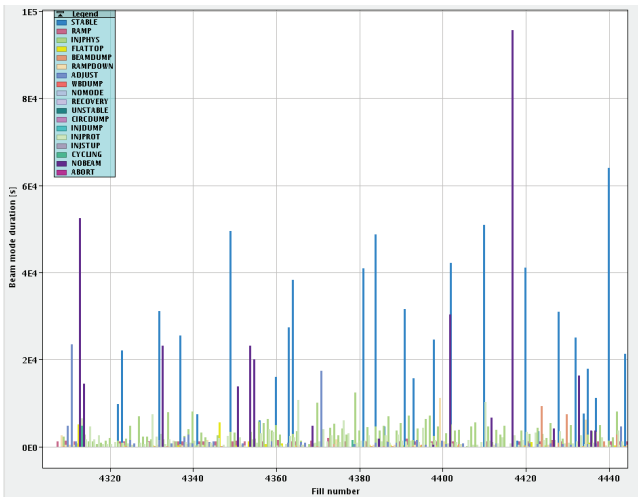


Figure 7: Beam mode duration statistics.

Luminosity-intensity Correlation in Stable Beams

Another example of the functionality of the tool is shown in Fig. 8, in which the ATLAS instantaneous luminosity is plotted as a function of the B1 intensity during stable beams. Although the luminosity and intensity decay depend on a number of factors, such as emittance blow-up and losses at collimators respectively [10], this type of analysis is still useful to obtain a quick overview of the situation over several fills online.

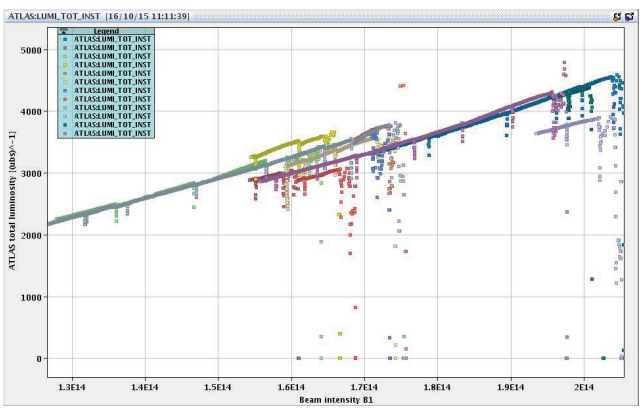


Figure 8: ATLAS instantaneous luminosity as a function of B1 intensity during several fills.

CONCLUSION AND OUTLOOK

This paper documents a tool which is useful for performing fill-to-fill analyses need to monitor the performance and behaviour of the LHC and its equipment. It has been successfully in several case studies which have been presented. Further enhancements of the tool include curve fitting, which could be used for example to calculate the beam lifetime evolution during stable beams.

REFERENCES

[1] Report No. CERN-2004-003-V1, edited by O. S. Brüning, P. Collier, P. Lebrun, S. Myers, R. Ostojic, J. Poole, P. Proudlock, 2004.

[2] C. Roderick, L. Burdzanowski, G. Kruk, "The CERN accelerator logging service - 10 years in operation: a look at the past, present and future". In Proceedings of ICALEPCS'13, San Francisco, CA, USA, pp. 612-614, 2013.

[3] V. Baggiolini et al., "JAPC - the Java API for Parameter Control". In Proceedings of ICALEPCS'05, Geneva, Switzerland, 2005.

[4] X. Buffat, M. Lamont, S. Redaelli, J. Wenninger, "Beam based optimization of the squeeze at the LHC". In Proceedings of IPAC'11, San Sebastian, Spain, pp. 1867-1869, 2011.

[5] D. Kirchner, "Feature set expansion for the SysteMA software", CERN-STUDENTS-Note-2014-152, (2014).

[6] R. W. Assmann et al., "Requirements for the LHC collimation system". In Proceedings of EPAC'02, Paris, France, pp. 197-199.

[7] G. Valentino, A. Danisi, "Follow-up of collimator LVDT drift analysis". Presented at the 167th LHC Collimation WG meeting, 04.11.2013.

[8] A. Danisi, A. Masi, R. Losito, Y. Perriard, "Design optimization of an ironless inductive position sensor for the LHC collimators".

[9] X. Buffat, "Betatron squeeze optimization at the Large Hadron Collider based on first year of operation data", Masters thesis, CERN-THESIS-2011-006 (2011).

[10] F. Antoniou, G. Arduini, Y. Papaphilippou, G. Papotti, "Building a luminosity model for the LHC and HL-LHC". In Proceedings of IPAC'15, Richmond, VA, USA, pp. 2042-2045, 2015.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

FORMALIZING EXPERT KNOWLEDGE IN ORDER TO ANALYSE CERN'S CONTROL SYSTEMS

A. Voitier, M. Gonzalez-Berges, F.M. Tilaro, CERN, Geneva, Switzerland
M. Roshchin, Siemens AG, Corporate Technology, München, Germany

Abstract

The automation infrastructure needed to reliably run CERN's accelerator complex and its experiments produces large and diverse amounts of data, besides physics data. Over 600 industrial control systems with about 45 million parameters store more than 100 terabytes of data per year. At the same time a large technical expertise in this domain is collected and formalized. The study is based on a set of use cases classified into three data analytics domains applicable to CERN's control systems: online monitoring, fault diagnosis and engineering support. A known root cause analysis concerning gas system alarms flooding was reproduced with Siemens' Smart Data technologies and its results were compared with a previous analysis. The new solution has been put in place as a tool supporting operators during breakdowns in a live production system. The effectiveness of this deployment suggests that these technologies can be applied to more cases. The intended goals would be to increase CERN's systems reliability and reduce analysis efforts from weeks to hours. It also ensures a more consistent approach for these analyses by harvesting a central expert knowledge base available at all times.

INTRODUCTION

CERN employs about 600 industrial Supervisory Control and Data Acquisition (SCADA) systems for the supervision and monitoring of its accelerators, detectors and infrastructure machines. While the day-to-day operations are running smoothly, a growing need appeared to exploit the data generated by these applications. Indeed, collectively they produce more than 100 terabytes of control data over 45 million parameters. The gathered data could be seen as a deep reflection of the current state of the processes under control. A lot of information about the performance, stability and overall behaviour of the machines resides within these data. Today, an expert can manually follow the signals deemed important and apply his/her knowledge to maintain a good level of service. However, current tools for industrial control systems are not properly designed for doing such dedicated analysis. The external analysis tools used today are not well integrated with the operator applications. Moreover the size and complexity of some systems does not allow running advanced data analytics methods in normal office computers. In addition, this way of working is a non-sense for analyses that have to be part of the operational tools themselves. Lastly, while the experts are knowledgeable about their domain, they are not properly skilled to perform these data analysis or computing problems tasks.

In this context, formalizing expert knowledge means capturing the methods and knowledge used by experts and transforming them into analyses that can be scaled out to all similar systems without burdening the users with a vast amount of manual operations to carry out.

From a computer science point of view, this problem is part of the Data Analytics, or Big Data, field which combines technologies for processing vast amount of data carrying out analytical tasks tailored in our case to industrial control system needs

Once an analysis can be applied to solve a control problem, it becomes part of the control system itself making possible to raise awareness of operators on specific issues from the supervision application they are accustomed to.

This paper will present the knowledge capture and the methods used to detect specific conditions traditionally done "by hand" (exporting data, importing them into a spreadsheet software or writing an adhoc script) or by looking at trends. Then, we discuss briefly their implementation as analytics tools that were then scaled out and applied automatically with the help of readily available software solutions.

CONTROL SYSTEM DATA ANALYSIS

Control systems play an essential role to run the CERN accelerators complex and generate a huge amount of data that can be used to analyse the behaviour of these systems and to find insights useful to operators and experts. The data analytics activities have been divided into three different categories:

- Online monitoring
- Fault diagnosis
- Engineering design

These three families of analysis focus on control data to offer analytical services as added value on top of the traditional industrial services.

Online Monitoring

Currently the monitoring and operation of CERN industrial systems is – for most parts – achieved through the deployment of specific applications based on the commercial SCADA software WinCC Open Architecture [1].

Therefore, these data analytics activities aim to add new features and services to this monitoring layer. This close integration allows reaching a high level of automation

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

necessary to provide both real-time and historical information on their performances.

Events Threshold Learning Analysis

Most events produced by the control systems are already filtered out and only the most relevant are checked by the operators in order to take appropriate actions. Nevertheless, the rejected events that cannot be handled manually may contain valuable insight on the status of the running systems [2].

The main objective here was to analyse these hidden data to provide operators with valuable information improving their understanding of the systems.

An online analysis system - based on Complex Event Processing (CEP [3]) engines - has been designed and developed to continuously collect events generated by each device. These messages have been clustered by type to avoid replicas and to produce structured data

Then, the generated data stream has been parsed by an online learning algorithm in order to discover behavioural patterns [4][5].

Specifically, the analysis goes through two different steps. Firstly, the stream of events is used to learn the amount of messages produced by single devices in normal conditions. Secondly, the algorithm uses the learnt behaviours to detect anomalies [6], with the assumption that faults in the system would result in the generation of an increasing number of messages (as shown in Fig. 1).

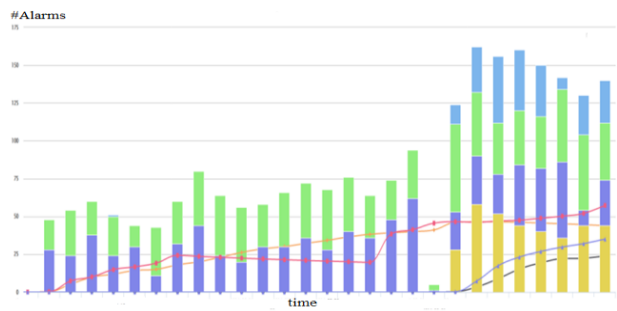


Figure 1: Cumulative alarms number and learnt threshold.

The nature of CERN systems, continuously updated both in hardware and software, imposes that the learning phase runs continuously, trying to detect new behaviours.

Signal Oscillation Analysis of the LHC Cryogenics System

The signal oscillation detection analysis [7] represents a second example of online monitoring activity [8]. Unlike the previous scenario where textual events were analysed, this time it makes use of numerical data (such as pressure values or valve position). The analysis can be applied to any control system where signal oscillation detection [9] is of interest.

For instance, in the cryogenics systems several abnormal behaviours have been identified by inconsistent sensor readings when the system was running smoothly. In this specific analysis the attention was focused on the detection of the oscillation of control valves. Under nominal conditions the process values oscillate, causing the valves to open or to close as expected. However, for various reasons these valves may start oscillating with an unexpected frequency or amplitude causing hardware damage.

The developed algorithm follows the analysis flow shown in Fig. 2. It consists first in a univariate signal analysis with a sliding time window. Then, the discrete Fourier transform is calculated to detect possible peaks in the spectrum [10][11][12] comparing each component against a given threshold. This threshold is calculated on the base of expert knowledge with a shape of a logarithmic function, which best fits the frequencies component amplitudes.

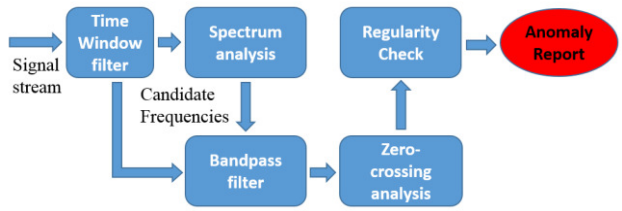


Figure 2: Analysis flow of the oscillation detection algorithm on a sample signal.

To verify it the initial signal is given as input to a band pass filter built on the previously discovered candidate frequencies. The frequencies that have passed the precedent conditions are further analysed. A zero-crossing analysis [13] is applied to the demeaned version of the filtered signal to check the presence of oscillations. This condition is verified if the number of zero-crossing is higher than a parameterised threshold, the zero-crossing rate. This rate is proportional to the frequency under analysis: lower frequencies will be associated to lower zero-crossing rates. As a final step the regularity of both the oscillation period and amplitude is checked by comparing the standard deviation of the period/amplitude with its mean. After an initial tuning the above algorithm has been able to detect the presence of multiple oscillations [14] and found their relative oscillation periods by recognizing regular patterns in the analysed data.

Parallelization of the Control Analysis

The huge amount of data produced by the CERN control systems will make necessary to run the developed algorithms against a large dataset of control signals and events. A cluster-based computing approach would be able to handle the enormous computation needs. This is the reason why a Docker-based cluster solution has been designed to parallelize the execution of such algorithms, showing the positive benefits of scaling the analysis across multiple nodes. Moreover, the lightweight portability of Docker containers minimizes the deployment issues linked to differences in execution environments.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Fault Diagnosis

This category includes the activities which perform an analysis posteriori to a fault occurrence. Therefore, it mainly aims at finding out the possible reasons for malfunctions looking into historical data. However, due to the complexity of the analysed systems it is generally not possible to identify the real initial cause factors. Nevertheless, it is necessary to reach a convenient degree of accuracy (diagnostic resolution) to which faults origin can be located to guide operators to relevant events/anomalies which need to be further investigated. As an example of this analysis category the GAS system alarms avalanche is presented.

Use Case: Gas System Alarms Avalanche

Supervision systems trigger alarms when process variables are out of their acceptable ranges. Alarms are the visible symptoms of an anomaly that may have occurred even several hours before, but they do not pinpoint to the exact root cause of the problem.

When a fault appears many alarms and events are raised rapidly by the system due to interaction between the different sub-systems. Moreover, the 1st raised alarm is not necessarily the most relevant to identify the root cause of the problem. Thus, alarm and event sequences must be analysed in depth by the operators to deduce a correct diagnosis before taking the appropriate actions [15]. This flood of information often overloads operators, generated by slowing down the diagnosis process.

Our analysis consists of a fault isolation method based on event pattern matching. As initial step all the events generated by specific faults are collected. These information are then processed to detect events that were always present in a specific fault list, even with different orders. As a last step, these fault signatures are injected into an expert knowledge database which can be used to detect occurrences of similar faults in the future (Fig. 3). It must also be pointed out that the method proposed here is quite generic and not dedicated to a specific process. For the purpose of this work a simulator has been used to replicate the faults and extract accordingly the generated event list.

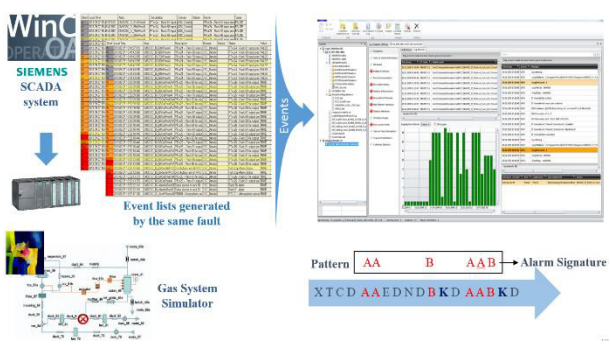


Figure 3: Analysis flow of the gas system alarms.

Engineering Design

The last family of activities is related to the analysis of historical data to draw conclusions about systems behaviour. The results can then help engineers to optimize specific system aspects like control regulation, parameterisation, or even drive the design of new parts of a system. Several use-cases, such as the analysis of historical data related to the CERN electric network which belong to this category, are under investigation and their results will be published later. Then, future consumption of different CERN areas can be predicted according to external factors like the accelerators schedule, weather conditions, technical interventions, and so on.

CONCLUSION

The control system of a facility produces a large amount of data and the regular extraction of insights from these data is a burdening task for experts, especially in the context of a very large and complex machine.

Capturing expert knowledge and formalizing how it can be used to deter issues is a first and essential step to provide assisting tools.

Once an abstract knowledge is converted into a tool it can be connected to the control system. This tool allows automation of advanced monitoring and pre-emptive maintenance, tasks that were previously triggered mostly by manual operators and experts actions. It helps them to look for abnormal conditions leveraging both their knowledge and scalable computing resources. It increases efficiency and availability of a machine, and lowers risks of disastrous events.

Knowledge capture and analysis is not always a straightforward task. Firstly, the expert needs a good understanding of the potential of an analysis tool in order to express problems that are addressable. Then, a mathematical approach is needed to transform a problem statement into a program able of discovering the actual issues from the data. Also, often an intermediate data science expert is needed as the machine experts are not fluent in the data analysis tools available at hands.

In addition, the set of available data analysis tools is limited by the interoperability capabilities of industrial control systems. These tools also have to be a good match for computing parallelisation frameworks. This includes the capacity of their algorithms to be parallelised, as well as for the software itself to be scheduled in a greater infrastructure.

Despite these limitations we have seen that these projects led to improved confidences into analyses on control systems. Involving machine experts for their knowledge is a prerequisite. Naïve approaches such as large-scale cross-correlations are not enough to extract meaningful insights. Further work is planned on different use cases of online monitoring, fault diagnosis and engineering design.

Furthermore, to address these limitations, we plan to provide a data analysis service (DAaaS) adapted to

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

control systems needs and their classical infrastructure, available to the experts directly.

ACKNOWLEDGEMENTS

A particular thanks to B. Bradu, P. Gayet, A. Rijllart, N. Bouchair, E. Blanco Vinuela and C.R. Vintila for their helpful contribution to the presented work.

REFERENCES

[1] ETM.at website: http://www.etm.at/index_e.asp
[2] R. de Haan, M. Roshchin, Detecting Temporally Related Arithmetical Patterns, 2012.
[3] E. Wu, U. Berkeley, High-performance complex event processing over streams, 2006.
[4] V.J. Hodge, J. Austin, A Survey of Outlier Detection Methodologies, 2004.
[5] S. Ramaswamy, R. Rastogi, K. Shim, Efficient Algorithms for Mining Outliers from Large Data Sets, 2000.
[6] Knorr, Algorithms for Mining Distance-Based Outliers in Large Datasets, 1998
[7] L. Wilhelm, M. Proetzsch, K. Berns, Oscillation Analysis in Behavior-Based Robot Architectures, 2010.
[8] T. Miao, D.E. Seborg, Automatic detection of excessively oscillatory feedback control loops, 1999
[9] L. Ettaleb, M.S. Davies, G.A. Dumont, E.E. Kwok, Monitoring oscillations in multi-loop systems, 1996.
[10] C. Pryor, Auto-covariance and power spectrum analysis, Control Engineering, 1982.
[11] G. Chowdhary, S. Srinivasan, E.N. Johnson, A Frequency, Domain Method for Real-Time Detection of Oscillations, 2010.
[12] N.F. Thornhilla, S.L. Shah, B.Huang, A.Vishnubhotla, Spectral principal component analysis of dynamic process data, 2002
[13] B. Kedam, Time series analysis by higher order zero -crossings, 1993.
[14] N.F. Thornhill, B. Huang, H. Zhang, Detection of multiple oscillations in control loops, 2003.
[15] S. Charbonnier, N. Bouchair, P. Gayet, Fault isolation by comparing alarm lists using a symbolic sequence matching algorithm, 2014.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INTEGRATING WEB-BASED USER INTERFACE WITHIN CERN'S INDUSTRIAL CONTROL SYSTEM INFRASTRUCTURE

A. Voitier, M. Gonzalez-Berges, P. Golonka, CERN, Geneva, Switzerland

Abstract

For decades the user interfaces of industrial control systems have been primarily based on native clients. However, the current IT trend is to have everything on the web. This can indeed bring some advantages such as easy deployment of applications, extending HMIs with turnkey web technologies, and apply to supervision interfaces the interaction model used on the web. However, this also brings its share of challenges: security management, ability to spread the load and scale out to many web clients, etc... In this paper, the architecture of the system that was devised at CERN to decouple the production WINCC-OA based supervision systems from the web frontend and the associated security implications are presented together with the transition strategy from legacy panels to full web pages using a stepwise replacement of widgets (e.g. visualization widgets) by their JavaScript counterpart. This evolution results in the on-going deployment of web-based supervision interfaces proposed to the operators as an alternative for comparison purposes.

INTRODUCTION

Supervisions for industrial control systems are applications that have to remains stable and shielded from any potential factor of disruption. As such, they are usually deployed on dedicated and isolated servers to minimize risks and control its access.

However, users of these systems have a pressing demand of accessing, or at list viewing, the state of their systems without the protective hurdles. Another limitation is the need to install specific software on the computer used for accessing remotely the supervision applications.

These requirements are following the tendency of the present computing era where we got accustomed to seamlessly access any remote services through a simple web-browser or a mobile application. A so-called Web UI (Web User Interface) for the SCADA tool used at CERN is implemented to answer these requests, provided security is strictly enforced.

This UI allows to view existing panels and synoptic used in the current native UI. The graphical engineering editor of the SCADA tool is also accessible with a web browser, suppressing the need to install any specific software. Another important advantage is to profit from standard and modern web technologies to enhance the usability of user interfaces as a wide range of versatile graphical widgets are available to web-enabled technologies [1].

For the end users it brings new paradigms of using supervision applications. It allows accessing them from a field intervention, or from a non-professional place for rapid remote diagnostics. Yet another usage is to broadcast the state of a particular machine to hundreds or thousands of users of this machine, which is a need for physics experiment collaborations.

This paper will present how this project went through several phases of iterative researches. First by prototyping a Web UI with classic web components. Then, facing important limitations, introducing an alternative approach similar to a remote desktop software embedded in a browser.

WEB ACCESS FOR INDUSTRIAL CONTROL SYSTEMS

At CERN, supervisions for industrial control applications amount to a total of 600 systems. All production applications are hosted on a dedicated network isolated from the intranet and internet by firewalls. To enable a web access, the following four main requirements have been postulated.

Security: Making control systems of very costly machines accessible to the entire world means security is paramount. A powerful access control has to be used while keeping the balance with usability. Indeed, with a very secure but hard to use system users have a tendency to circumvent the secure measures, putting the protected systems at risk. Another point is to stick to web security standards extensively reviewed and tested by experts and not forge custom solutions.

Legacy: In the past 15 years at CERN a lot of scripts and panels have been developed for the SCADA tool. This created an important base of legacy work that cannot be deprecated. The Web UI has to be compatible with existing developments. Yet, at the same time it should remain flexible and open to improve the existing panels with user experience paradigms the web can offer.

Scalability: The new Web UI infrastructure should support up to a thousand simultaneous clients in read-only mode. This is for the use case of broadcasting a machine state to the many users interested in these information. One related requirement is that the new UI system should minimize any extra load on the production data servers running SCADA systems. At best, a decoupling and/or buffering should exist between a SCADA system and its numerous clients.

Integration: Industrial control systems are about integrating off-the-shelves components. A Web UI for it should have the same philosophy of reusing readily available web servers and services. In particular, it should integrate with web-hosting services centrally supported within an organisation running industrial control systems. As such, it emphasises the need to use compatible web standards and be platform independent. In a similar way, it

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

should also integrate naturally on the client side. Users should not have to install any special plugin. The services should only make use of standard web technologies. Today those are HTML in its 5th version for structuring data, Javascript for scripting, and CSS for styling.

Four use cases have been initially intended:

1. One user running one panel. In that case, not only the user can see the information the panel display, such as process values. It should also let the user click on buttons, interacts with other widgets and open new panels. Such display can be used on mobile devices for field interventions or remote debugging (while an on-call person is away from a control room for instance).
2. Many users can view one panel. This is intended for broadcasting information in read-only mode. Therefore, the actions of one user should not affect the control system and also what the others can see. In that case, interactions are limited to zoom in and out of trends, and to navigate between different broadcasted panels.
3. Editing panels and scripts. This amount to replicate or enable the original graphical engineer editor of the SCADA tool to run in a browser.
4. Shift away from the classical SCADA paradigm of showing panels (merely the content of a framed window) to rather display web pages (where the content is flowing horizontally and/or vertically). In that case, usual web page generation methods apply. The control system only need to provide ways to get, set and subscribe to data, as well as call certain functions remotely.

FULLY WEB BASED SOLUTION

The first approach chosen was to replicate all UI functionalities using conventional dynamic web page generation. This meant two important capabilities: Firstly, being able to read the original description format of the SCADA tool panels. Conveniently, these can be exported to XML. Secondly, re-implementing all the editing features of the graphical engineering tool.

Two major limitations arose while implementing prototypes for this solution. The first was that panels contain code. This code is written in a proprietary language for the SCADA tool. This makes it difficult to execute it on client-side as the only commonly accepted mean of scripting in a browser is Javascript. The company of the tool attempted to write a translator. However, it resulted in a solution implying modification to the original codes. A more exotic approach was considered, compiling the code to LLVM [2] bytecode, and executing it in the browser through the use of the software package Emscripten [3]. The initial test has shown that it would require a lot of developments for the compiler. Panels can also contains code that is not purely doing user interface logic. Indeed, sometimes these panels contain business logic code that should be executed on server-side. Solving this problem would require a major refactoring of the existing code base.

The second major limitation was that interpreting the XML files to draw graphic primitives of the panel is a heavy process. There is a large combination of graphical features, themselves dependent on the way it is rendered today by the native Qt-based UI. This made it difficult to keep a compatibility to the pixels, a feature needed by the synoptic drawings. Similarly, the engineering tool would also had to be replicated in the same way. As this tool is quite complex and feature full, it made the task almost impossible to write it entirely in another technology (Fig. 1).

Figure 2 describes the architecture used. This full-web replication approach was dismissed based on all these limitations.

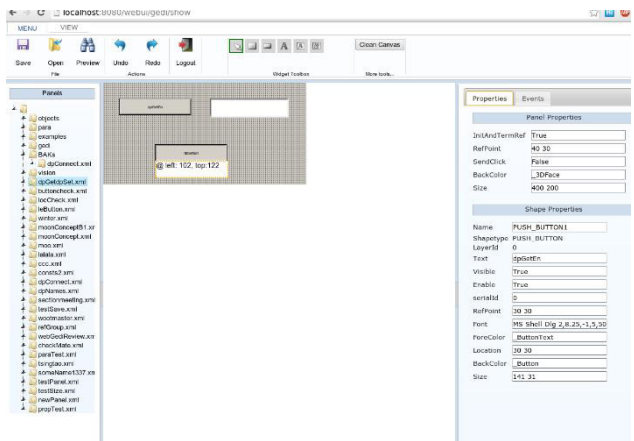


Figure 1: Screenshot of the engineering tool replicated in a prototype of the full web solution.

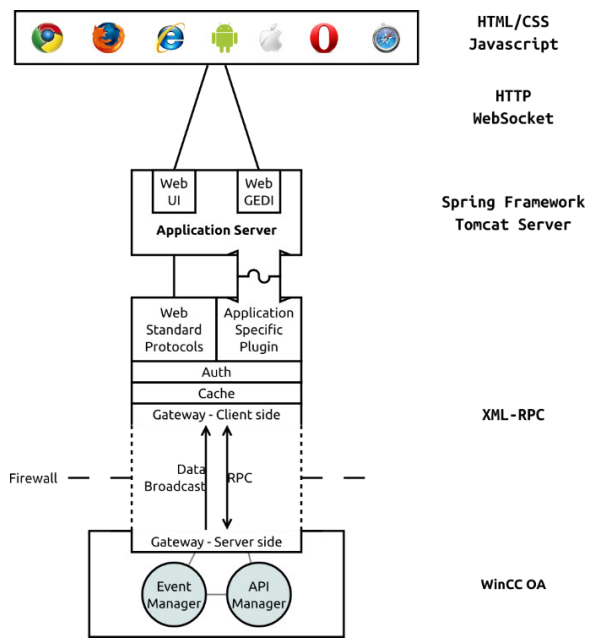


Figure 2: Architecture of the full web solution.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

INTERMEDIATE SOLUTION

As a full web solution is not yet possible, the company providing the SCADA tool proposed a more pragmatic solution. It consists of rendering native panels on the server-side, and send its representation to the browser through a special protocol (Fig. 3). The browser reconstruct the panel representation using HTML5 canvas. Initially, the solution was thought to send the graphical primitives as binary commands driving the Javascript drawing engine. In the end, a protocol similar to VNC, usually used for remote desktop applications, was chosen instead. A Javascript client-side program takes care of implementing the protocol and draws on the canvas.

This solution presents the advantages of keeping full compatibility with the legacy panels and to run the panel code directly on the server. Several performance evaluations were carried out until representative test panels could be displayed over bandwidth-restricted mediums such as copper landlines and mobile phone networks.

However, although this approach is meeting most of the requirement, when many users view the same panel, the interactions of one user are visible to all users. This prevent the inclusion of scrollable trends and navigation buttons. To work around this issue, Javascript widgets can be overlaid on top of the VNC canvas. To implement this functionality, a hook was provided to the panel's developer by reusing a previous CERN project integrating Javascript widgets and code into native panels [1]. In such case, instead of being rendered on the server-side, the Javascript insertion is sent to the client browser. All interactions with these client-side widgets are limited to a single user. Nevertheless, in case of widgets making use of process data (e.g. trends) it is needed to disassociate the data transfer from the VNC protocol within the same websocket.

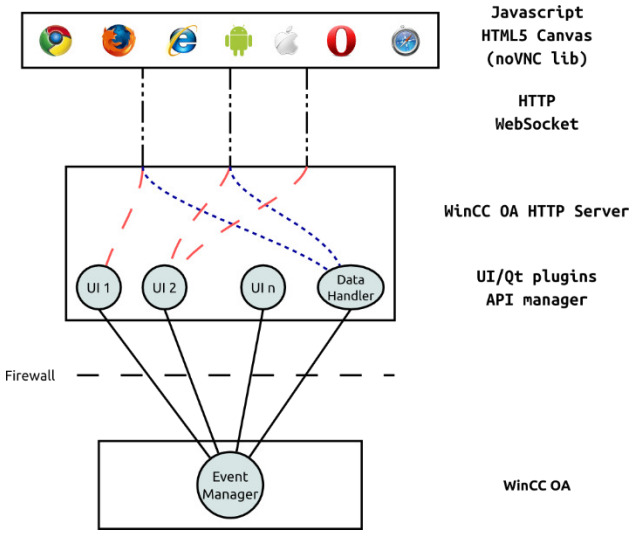


Figure 3: Architecture of the VNC-based solution. Red dashed lines are the communication of rendering primitives. Blue dotted lines are for the process data. Dashed and dotted lines are a mix of both types going through a single websocket.

SECURE IMPLEMENTATION

In the present state-of-the-art, two-factor authentication is often considered a secure and usable measure. It is understood by users that the additional minor hurdles are a necessary evil to benefit from the additional level of protection. Therefore, its usage is warranted to authenticate users accessing the Web UI. CERN IT service provides a single sign-on (SSO) implementation (based on Shibboleth) reusable by all web services across the organisation. This service provides two-factor authentication based on the user password plus several second factors: SmartCard, Yubikey, SMS OTP, and Google Authenticator (Fig. 4). The user's password is actually never shared with the Web UI application front-end. Once a user is identified, a central LDAP service provides the information about which groups the user belongs to. This is used to define the roles and allowed actions within the supervision application [4].

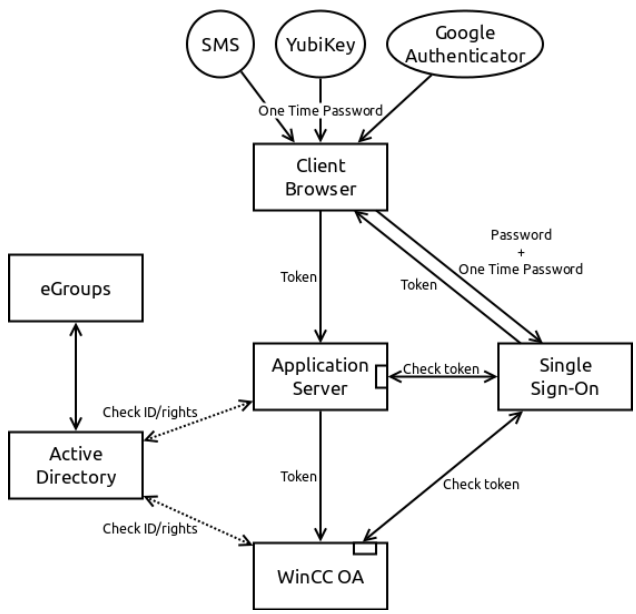


Figure 4: Single sign-on implementation for control system Web UIs.

CURRENT DEPLOYMENT

The VNC solution is expected to enter in production at CERN by the end of 2015. To accommodate the various requirements, the web server provided by the SCADA tool will not be the one facing incoming requests. Instead, the requests will first arrive into an application server managing the initial SSO negotiation, spawning the UI processes doing the rendering and connecting these processes to the original web clients through a websocket proxy. This implementation is thought to be scalable by spawning the UI processes on a farm of virtual machines managed by CERN's Openstack service. These machines will be part of a set of computer used to define firewall rules such that the production SCADA servers are exposed only to these machines. In turn, these machines will never be reached by public traffic as all their communications

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

with clients will transit through the application server proxy. Penetration tests organized by CERN's security team will also be performed to validate the overall solution.

CONCLUSION

Enabling web access to industrial control system is an expected feature by many CERN users. Control systems being critical pieces of the installations it is important to ensure a proper implementation. Considerations about security, legacy developments, scalability and integration are fundamental in this project.

Through various researches, prototype implementations, and benchmarking, a suitable solution has been found respecting all requirements. Hence, users of industrial control systems at CERN will soon be able to access their supervision applications through a web browser without using dedicated software. A proper security level will be obtained by reusing proven and standard solutions. The proposed architecture will allow us to scale with a growing amount of users while introducing more security layers. Finally, the reuse of web application servers provided as a central service by the organisation will help to keep the maintenance and administration tasks to a minimum.

This solution is the first step, in the long term, we will look for an implementation closer to traditional web developments that will allow us to use any major and mature web framework.

ACKNOWLEDGMENT

A particular thanks to M. Koller, C. Stoegerer, F. Glege, M. Janulis, S. Lueders, S. Lopienski, and S. Petrovski for their helpful contribution to the presented work.

REFERENCES

[1] P. Golonka et al, "FwWebViewPlus: integration of web technologies into WinCC-OA based Human-Machine interfaces at CERN", CHEP 2013, Amsterdam, Netherland.

[2] The LLVM Compiler Infrastructure website: http://llvm.org/

[3] Emscripten: An LLVM-to-Javascript Compiler project page: https://github.com/kripken/emscripten

[4] P. Golonka et al, "Integrated Access Control for PVSS-based SCADA Systems at CERN", ICALEPCS 2009, Kobe, Japan.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A NEW DATA ACQUIRING AND QUERY SYSTEM WITH ORACLE AND EPICS IN THE BEPCII*

Chunhong Wang, Luofeng Li#,
Institute of High Energy Physics, Chinese Academy of Science, Beijing 100190, China

Abstract

The old historical Oracle database in the BEPCII has been put into operation in 2006, there are some problems such as the program operation instability and EPICS PVs loss, a new data acquiring and query system with Oracle and EPICS has been developed with Eclipse and JCA. On one hand, the authors adopt the technology of the table-space and the table-partition to build a special database schema in Oracle. On another hand, based on RCP and Java, EPICS data acquiring system is developed successfully with a very friendly user interface. It's easy for users to check the status of each PV's connection, manage or maintain the system. Meanwhile, the authors also develop the system of data query, which provides many functions, including data query, data plotting, data exporting, data zooming, etc. This new system has been put into running for three years. It also can be applied to any EPICS control systems.

INTRODUCTION

BEPCII has successfully built its control system with EPICS[1]. Besides using Archiver, a historical database[2] with Oracle had developed for a long term storage. The data acquisition program developed by Python got the EPICS PVs from the EPICS IOCs. This program was scheduled by a back processing Cron. When it is scheduled timely, many CA were recreated and took too much times. As a consequence, the first schedule had been not finished, the next schedule had started. Many such data acquisition programs had been running in parallel so that the control network congestion caused EPICS PVs loss. Such kind of problem had happened many times since 2006. So, it is necessary to reconstruct a new historical database and data acquiring and query system with Oracle. Since EPICS has been widely applied to accelerator control systems, it's becoming a hot topic how to store EPICS PV data into a database for a long term storage. So, the purpose of this new system development is to be commonly used to EPICS control system.

THE OLD ORACLE SYSTEM

The old data acquiring and query system[2] with Oracle consisted of three parts: data acquisition and Oracle database and data query (see Fig. 1).

The Old Data Acquisition Program

The data acquisition software programmed by Python got the EPICS PVs from the EPICS IOCs.

It was scheduled by a Cron processing in Linux. It has the following disadvantages:

- Ineffective data acquisition sampling rate (once per minute). It took many times for the CA channels reconnection when it was executed.
- The program was repeatedly performed periodically. It means the next schedule has started when the first program had not been finished. Many same programs were running simultaneously so that the control network congestion caused PVs data loss.

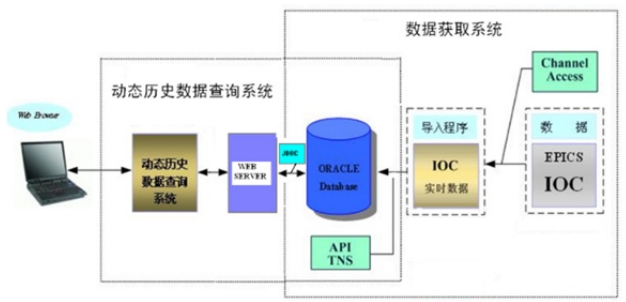


Figure 1: The old Oracle System[2].

Figure 2: the Interface of the old Query System[2].

The Old Query System

The old data query system was in B/S developed using Java. The query interface is shown in figure 2. It has the following disadvantages:

- It's impossible to implement the correlated subquery between the subsystems.
- The database tables are designed into one independent table for each subsystem (e.g. power supply control system table and vacuum control system table and so on).
- There is no any zoom in function and data without time stamp display in the interface.

*supported by NFSC(1137522)
#on leave from IHEP

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

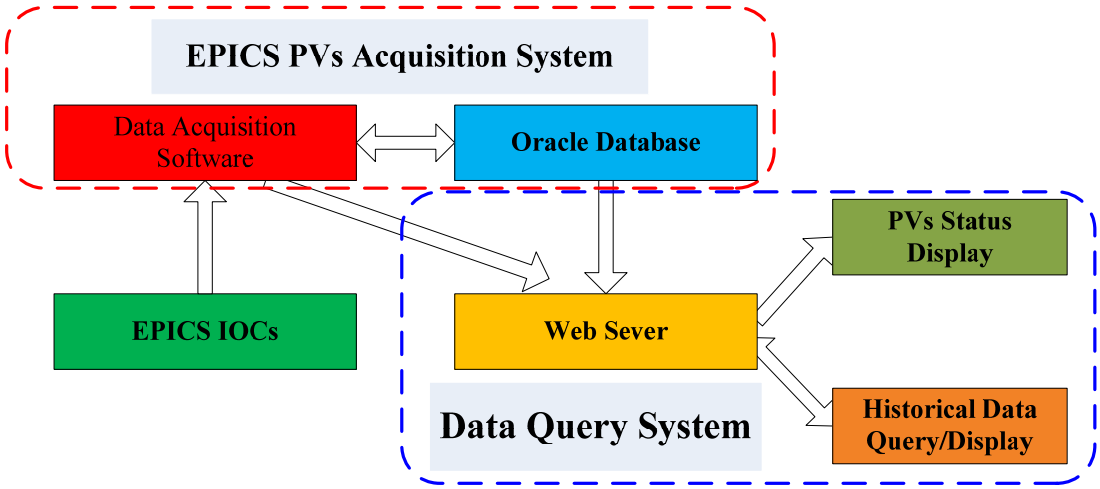


Figure 3: EPICS data access and query system architecture.

A NEW DATA ACQUIRING AND QUERY SYSTEM WITH ORACLE AND EPICS

The new system including data acquisition and query consists of three part: data acquisition and Oracle database and query (see Fig. 3).

The New Data Acquisition System

There are many interfaces to EPIC IOCs such as SCA、CA_Lab、MCA、CAP5、CaPython、JCA、CAJ and so on. JCA and CAJ are main interfaces to EPICS IOCs. After comparison on JCA and CAJ, we chose CAJ[3] as the interface to EPICS IOCs and used a monitor method to get EPICS PVs, then send them to the Oracle database. This system is developed using JAVA on RCP platform. The data acquisition engine is shown in figure 4.

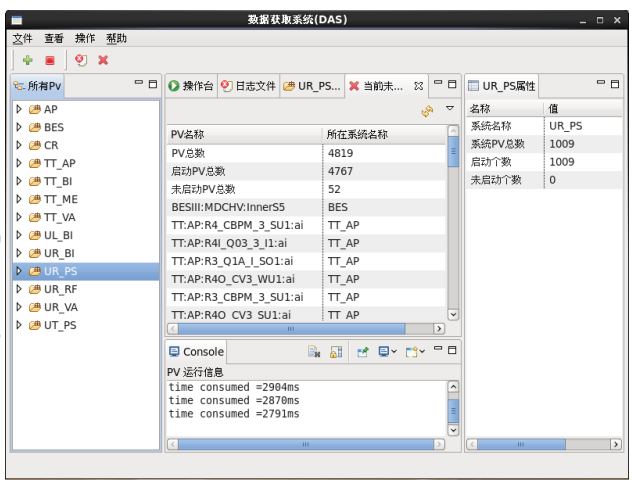


Figure 4: The data acquisition engine.

The data acquisition software using multithreading technology contains:

- Data acquisition thread.
- Data write thread.
- Data service thread.

This system has the following function:

- Data acquisition of start and stop.
- The recovery of PVs connection.
- Log management.
- PVs increases the wizard.
- PVs status view.
- Right control.

Oracle Database Model

There are about more than 5000 PVs to be stored in the Oracle database. These PVs are from EPICS IOCs of the different subsystems. There should be correlation between different data. So, we designed unique database tables: All_pv_list_tab and XX_tab. All_pv_list_tab mainly stored all PV static information, including name of PV, system name, alias, note, table, etc. XX_tab mainly stored PV concrete data, mainly including id, name of PV, PV date, PV value, including XX on behalf of the name of the system. All_pv_list_tab table structure as shown in table 1, XX_tab table structure as shown in table 2.

Table 1: All_pv_list_tab Table Structure

Field name	Type	Description
PVNAME	VARCHAR2(50)	PV 英文名称
SYSTEMNAME	VARCHAR2(100)	系统英文名称
LASTTIME	DATE	该字段保留
LASTVALUE	NUMBER (10,3)	该字段保留
DESCRIPTION	VARCHAR2(100)	系统中文名称
PVTABLENAME	VARCHAR2(20)	PV 所在表名
MACHINESTATUSNAME	VARCHAR2(100)	PV 中文名称

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Due to more than 5000 EPICS PVs to deposited in the database, in order to improve the query speed of data, optimize the database performance, the data table on the first of all in the name of the PV primary partitions, and then conducted on PV time partition. All PVs with a month of data is stored in a table space. It's convenient to remove historical data and the maintenance of the system.

Table 2: XX_tab Table Structure

Field Name	Type	Description
ID	NUMBER(20)	主键
PVDATE	DATE	PV 时间
PVVALUE	NUMBER(20,3)	PV 值
PVNAME	VARCHAR2(50)	PV 英文名称

The Historical Data Query System

The historical data query system takes EPICS PVs out of the database in the form of curve display on the web. This system mainly includes the web server program and web service, as shown in figure 5.

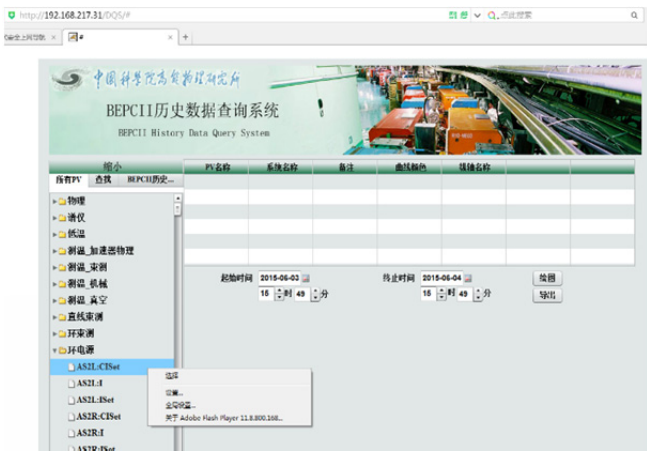


Figure 5: The interface of the historical data query system.

The web server programs are written in Java, database interaction using JDBC. The web page display service programs contain historical curve and the data acquisition engine status page. The web history curve display is developed using the Flex[4], the main function is as follows:

- EPICS PV search.
- Query the date selection.
- Curve color choices.
- Longitudinal axis curve range of choice.
- Common longitudinal axis selection.
- Curve magnification.
- Curve with a little time message.

The historical data query interface as shown in Fig. 6, shows from 2013/01/16 to 2013/01/18, in collision mode within two days of the shape of the beam intensity and collisions brightness curve.

The data acquisition engine status page is shown in Fig. 7. It's convenient for the general users to view data acquisition engine PV operation state and running state diagnosis in data acquisition system.

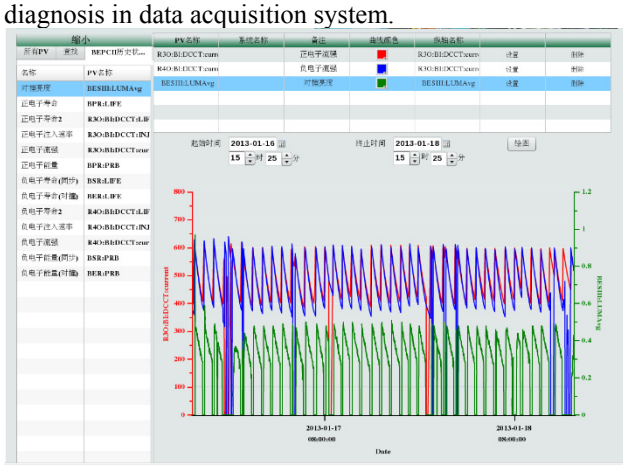


Figure 6: The historical data query interface and results.

数据获取软件PV状态

PV统计信息

1	PV总数	4819
2	已连接	4814
3	未连接	5

PV信息

	PV名称	状态
1	AS2L:CISet	连接
2	AS2L:I	连接
3	AS2L:ISet	连接
4	AS2R:CISet	连接
5	AS2R:I	连接
6	AS2R:ISet	连接
7	AS3L:CISet	连接
8	AS3L:I	连接
9	AS3L:ISet	连接
10	AS3R:CISet	连接
11	AS3R:I	连接
12	AS3R:ISet	连接
13	ASL:CISet	连接
14	ASL:I	连接
15	ASL:ISet	连接
16	ASR:CISet	连接
17	ASR:I	连接
18	ASR:ISet	连接
19	BER:LIFE	未连接
20	BER:PRB	未连接

1 共 241 页 20 1 - 20 共 4 819 条

Figure 7: PV data acquisition engine running state.

SYSTEM PERFORMANCE TESTING

We mainly use Jconsole for the system performance test. The Jconsole provides memory, threads, such as CPU performance monitoring. It can easily find memory leaks and thread deadlock. The Jconsole testing interface as shown in Fig. 8. The new EPICS data acquisition system and the data query system has been put into operation since July 2012.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The new data acquisition system test results are as follows:

- Data acquisition software runs stable.
- The minimum sampling time 2s.
- The biggest footprint of about 250 m.
- Data storage consumption time 1.5s (48 tables and 200 data into one table).
- No memory leaks.
- No thread deadlock.
- No thread deadlock.

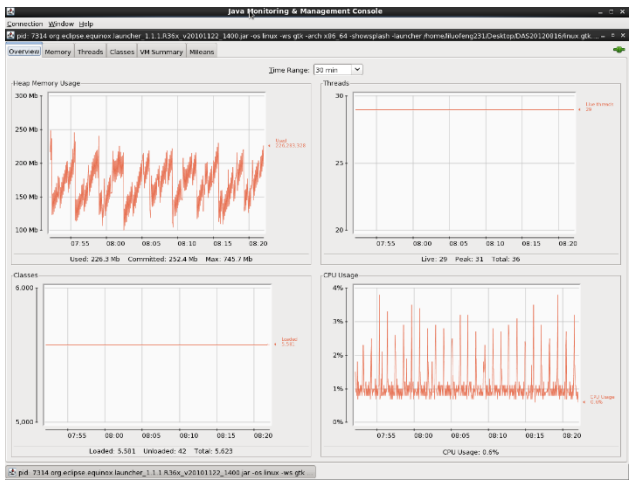


Figure 8: Jconsole test interface.

At the same time, data query system test results show that:

- Access concurrency > 50.
- Page 4s drawing time (1 day, 3 PVs), 15s (15 days, 3 PVs) .
- Web services work duration > 190 days.

According to the above test results, for the long time storage, we uses the minimum sampling interval time 1 minute feed into the database. So, 5000 PVs estimate 24 hours and need about 300 MB to 500 MB of storage space, including data storage and index files are stored.

From monitoring system, without any increase in the signals under the condition of invariable and access frequency, storage for one year is about 100 GB of data storage space. Compared with the old system 300 GB per year, it's significantly reduced and saved storage space.

CONCLUSION

Compared with the old dynamic historical data acquisition and the historical data query system, the new system mainly solves the following problems:

- Data acquisition program operation instability.
- EPICS PVs connection interface.
- EPICS PVs connection stability.
- EPICS PVs access loss.
- No friendly man-machine interface.
- Slow data query speed.
- Unrelated query data query.
- Data without time stamp prompt.
- Without magnification.

The new data acquisition and historical data query system with Oracle have been trouble-free operation more than three years since July 2012. The new system is stable and reliable, and can completely replace the old ones. It can not only meet the needs of physical beams personnel, but also greatly convenient equipment operation maintenance personnel equipment fault analysis and diagnosis.

REFERENCES

[1] EPICS website: http://www.aps.anl.gov/epics/
[2] 马梅，黄松等，“BEPCII 控制系统动态数据获取与历史数据查询系统的设计与实现”[J].核电子学与探测技术, 2009,4:764-768.
[3] M.Sekoranja, “Native Java Implement of channel access for Epics”[C]. Geneva: ICALEPCS, 2005.
[4] 王睿, “Flex 与 ActionScript 编程”[M].北京:机械工业出版社，2008.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PYTHON SCRIPTING FOR INSTRUMENT CONTROL AND ONLINE DATA TREATMENT

N. Xiong, N. Hauser, D. Mannicke, ANSTO, Sydney, Australia

Abstract

Scripting is an important feature of instrument control software. It allows scientists to execute a sequence of tasks to run complex experiments, and it makes a software developers' life easier when testing and deploying new features. Modern instrument control applications require easy to develop and reliable scripting support.

At ANSTO we provide a Python scripting interface for Gumtree [1]. Gumtree is an application that provides three features; instrument control, data treatment and visualisation for neutron scattering instruments. The scripting layer has been used to coordinate these three features. The language is simple and well documented, so scientists require minimal programming experience. The scripting engine has a web interface so that users can use a web browser to run scripts remotely.

The script interface has a *numpy*-like library that makes data treatment easier. It also has a GUI library that automatically generates control panels for scripts. The same script can be loaded in both the workbench (desktop) application and the web service application for online data treatment. In both cases a GUI will be generated with similar look and feel.

INTRODUCTION

The Gumtree scripting interface has a *numpy*-like Python library that makes data treatment easier. It also has a graphical user interface library that automatically generates control panels for scripts. The generic GUI rendering feature allows the same scripts to be loaded in both the desktop applications and the web service applications. The scripting interface benefits both the users and the developers. Users can easily make scripts to run experiments or treat the data, with a graphical interface automatically created for these scripts. Developers save a lot of time deploying new products when using this feature since they don't have to recompile. The total cost of operating this framework is therefore significantly reduced

SYSTEM ARCHITECTURE

It is always important to provide a decent scripting feature in scientific software. When users want to run an experiment, it is a more flexible and direct way to write a script to finish some complex experiment steps. A smart graphical user interface such as software wizard is very useful, but has limited scope. Once users become more familiar with the instrument and the software, they may want to write ad hoc scripts to run complex experiments. In another scenario, when users want to do data treatment on the experimental results, experienced users write their own scripts to do it. In this way they know exactly what will be done on their data and the scripts are always under their control. Many users love to write the scripts themselves. Other users who don't do programming still agree on the importance of scripting. And they will seek support from the instrument team to write scripts for them.

Gumtree is an application that provides general framework for building scientific software [2]. It is customised at ANSTO for neutron scattering experiments. It provides experiment control and data treatment services for 9 neutron scattering instruments in the Bragg Institute, ANSTO. It is designed to provide a simple and reliable scripting feature for instrument users.

Gumtree software is written in the Java programming language. But the scripting feature is provided in Python language. As shown in Figure 1, the Python engine is a core service of the application.

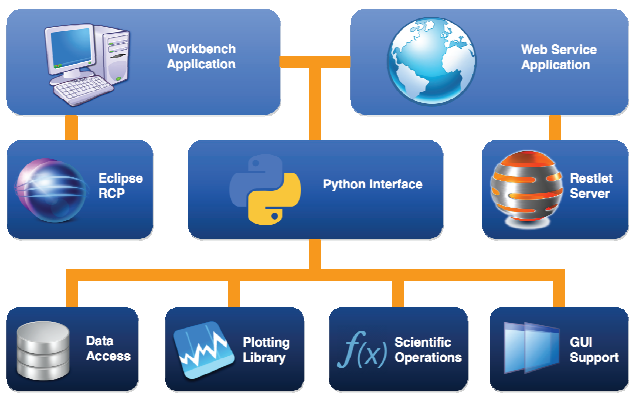


Figure 1: Gumtree system architecture.

MULTI-DIMENSIONAL ARRAY OBJECT

One of the most important scripting packages in Gumtree is the multi-dimensional array object package. It is a *numpy*-like library that provides convenient access to array operations, the Dataset interface. However it is more than just *numpy*. It provides functions for treating experiment data, such as error propagation, carrying axes information, carrying metadata, and so on. These functions make it easier for users to write scripts for data manipulation and visualisation.

Categories of the Scripting Interface

The Dataset interface borrows a number of *numpy* function names. Besides that, it also has exclusive function names to make Nexus data reduction easier.

In implementation, the python Dataset is a wrapper of a Java CDMA object [3]. The greater part of the logic

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

happens in Java. An object wrapped with Dataset can be referenced in Java.

The Dataset interface functions can be classified into the following categories:

- Constructing
- Indexing, Iterating, Slicing and Modifying
- Arithmetic operations
- Mathematical and statistical functions

The following *numpy* methods have been implemented in the Dataset package:

- Array attributes: shape, ndim, size, dtype, tolist
- Array methods: copy, fill, flatten, reduce, take, put, max, min, sum, prod
- Array creation: instance, zeros, ones, linspace, arange, zeros_like, ones_like, as array, rand
- Array manipulation: tile, concatenate, take, column_stack, vstack, hstack, dstack, array_split, split, vsplit, hsplit, dsplit
- Array modification: fill, append, delete, put
- Maths: add, subtract, multiply, divide, negative, power, exp, ln, log10, sqrt, divide, reminder, sum
- Trig: sin, cos, tan, arcsin, arccos, arctan
- Stats: Random: rand, engine, seed

The following methods are exclusive to the Gumtree Dataset package:

- Error propagation – quick access to var and err
- Normalisation – normalising against nexus metadata
- File access – Nexus HDF file access
- Nexus metadata – quick access to nexus metadata
- Rich axes information – carry Nexus axes

The Analysis Interface

The Dataset is mapped to a Nexus file either stored in the physical drive or in memory. One can efficiently load a Nexus file into a Dataset. The Dataset is the subject of data analysis scripts, so it carries interfaces for data reduction and analysis.

The following features are designed to help analyse neutron scattering experiment data.

Error Propagation: a Dataset normally carries the error information. By default this is setup for Poisson count statistics which are commonly used in neutron scattering. It gets stored as variance in the memory. To access the Dataset variance, use dataset.var. To access the error of the Dataset, use dataset.err or dataset.error. To get the error from the Dataset will call a square root function (Poisson statistics) and hence is costly in compute time. When initialising the Dataset, one can choose to set the variance or not. If no variance is set, by default it will use a copy of the data storage as the variance.

Nexus Axes: if the Dataset is loaded from a Nexus file, it carries axes information provided by the file. If a Dataset is created from a helper function, one can initialise the axes in the argument list. If no axes information is provided, by default it creates an integer index as axes for the Dataset.

Nexus Metadata: Dataset provides interfaces to access Nexus metadata. Nexus metadata are treated as public fields of the Dataset. For example, to get the wavelength value of a Dataset, simply call dataset.wavelength. To expose metadata in the Nexus file as an easy accessible property, a path table is provided. Before a Dataset is created, one can set a dictionary file that contains the path table information to the Dataset class.

Normalisation: the Dataset can be normalised against certain metadata. For example total counts or counting time. To enable the normalisation, set the normalising factor in the Dataset factory.

Nexus Import and Export: There is a helper function to help you load Nexus data. The requirements are to set the folder path and instrument prefix, and the use of the Nexus 'signal=1' attribute. Then use df[index] to access a file that follows the ANSTO naming convention of [instrument prefix][seven digit index number].nx.hdf. This shortcut can be customised to the naming convention of any facility. Dataset interface supports exporting to a Nexus HDF file.

PLOTTING LIBRARY

The Python interface for plotting in Gumtree provides scientific plot functionality for one-dimensional curve plot and two-dimensional image plot.

Curve Plot: The curve plot is also called plot 1D. The plot takes vector Datasets as input. If the Dataset has axis information, the axis will be used to scale the horizontal axis of the plot. It is allowed to plot multiple Datasets in the same plot. The interface provides functions for managing the Datasets and how they are rendered. Below is an example of the curve plot, as shown in Figure 2.

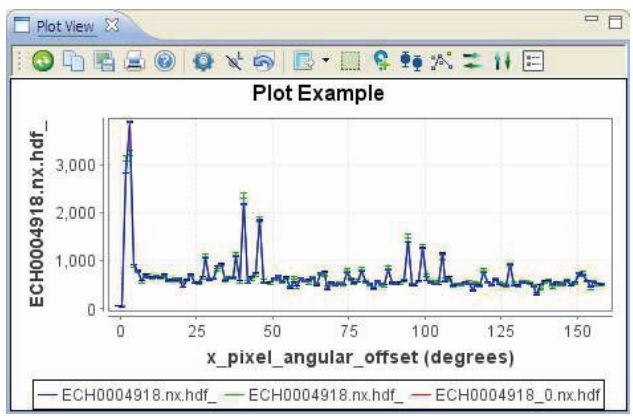


Figure 2: Example of curve plots in Gumtree.

Image Plot: the image plot is also called image 2D. The image plot takes a single two-dimensional Dataset as input. The Dataset may have up to two axes. The axes will be used to scale the vertical axis and horizontal axis of the plot. The plot renders the Dataset as a 2D histogram image.

Plot Control: The following scripting functions are provided for controlling the plots:

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- Axis Control – quick access to axes of the plot, such as zooming in and out, reversing axis, and switching to logarithm axis.
- Rendering Control – change rendering colour, shape, and weight, or add markers, legends and titles.
- Mask Control – add, remove, and change masks.
- I/O Control – import and export data to the plot.
- Event Control – add event handlers to the plot to handle mouse and keyboard event.

INSTRUMENT CONTROL INTERFACE

Gumtree Python scripting feature provides a library package for instrument control. This package helps users to write scripts that seamlessly combine experiment control and data treatment. For example, a calibration script for the 3-Axis Spectrometer uses the instrument control package to drive instrument devices and collect data. It uses the analysis package to calculate the offset, which will be used to calibrate energy and sample stage.

The instrument control package has two layers. The generic layer has functions for driving devices, setting parameters and collecting data. The instrument specific layer has helper functions for some unique procedures, such as driving the high voltage down on a detector before moving and ramping it up after the move.

AUTO GUI CREATOR

One of the most advanced features of the scripting application is the auto GUI creator. This feature creates a graphical user interface for scripts automatically. When users are writing scripts for their experiments, they don't need to worry about what the interface looks like in the application. They simply follow a template to create arguments and procedures. When scripts are loaded, Gumtree automatically generates GUI for the scripts. For example, Figure 3 shows the GUI automatically created by a data reduction script for the Strain Scanner instrument 'Kowari' at ANSTO.

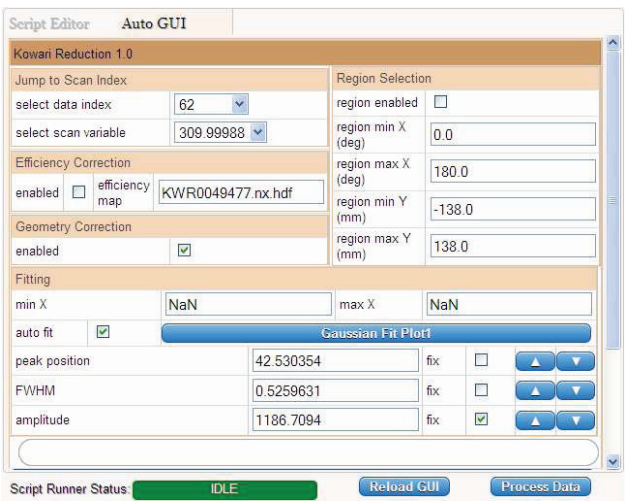


Figure 3: HTML auto-GUI created for a script.

When the user creates this script, all he needs to do is to define some arguments, and to use these arguments in procedures defined in the script. Such a facility minimises the learning requirements of programming with Python scripts. Even a novice user that doesn't have any programming experience with the Python language can make scripts that renders a rich, professional graphical interface.

The value of the feature is enhanced with the ability to execute the scripts on both the desktop application and online web application. Both applications provide similar graphical interfaces for the Python scripts. For example, in Figure 3, a data reduction script is rendered in the Gumtree web service application. The same script can also be used in the Gumtree desktop application for data treatment purpose.

When an argument is defined in the script, it can be defined in a list of types. The argument will be rendered differently according to its type. Table 1 shows a list of allowed types of argument for the Python scripts.

Table 1: Auto GUI Components

Python Type	SWT Widget	HTML Input Type
string, int, float	Text Widget	Text Input
bool	CheckButton	CheckBox
option	ComboViewer	Select
file	BrowseButton	Text Input
action	Button	Button
Group	MenuGroup	Table
progress	ProgressBar	jQuery widget
label, space	Label	Plain text

The Python scripting feature provides more GUI controls for advanced users. Users can define arbitrary table grid for the control panel to make the outlook more straightforward and more concise. Users can also have direct control of the widgets such as folding a group, enabling or disabling control widgets, and changing widget status.

WIDELY USED IN ANSTO

The Python scripting feature has been widely used for experiment control and data treatment in neutron scattering experiments at ANSTO. Software developers of Gumtree applications write python script modules for complex experiment control tasks. For example, the calibration task of the Thermal 3-Axis Spectrometer was written in Python scripts. The scripts combine instrument control and data analysis together, so that the calibration procedure is highly automated. Figure 4 shows an example of using the script to do Nickel calibration routine.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

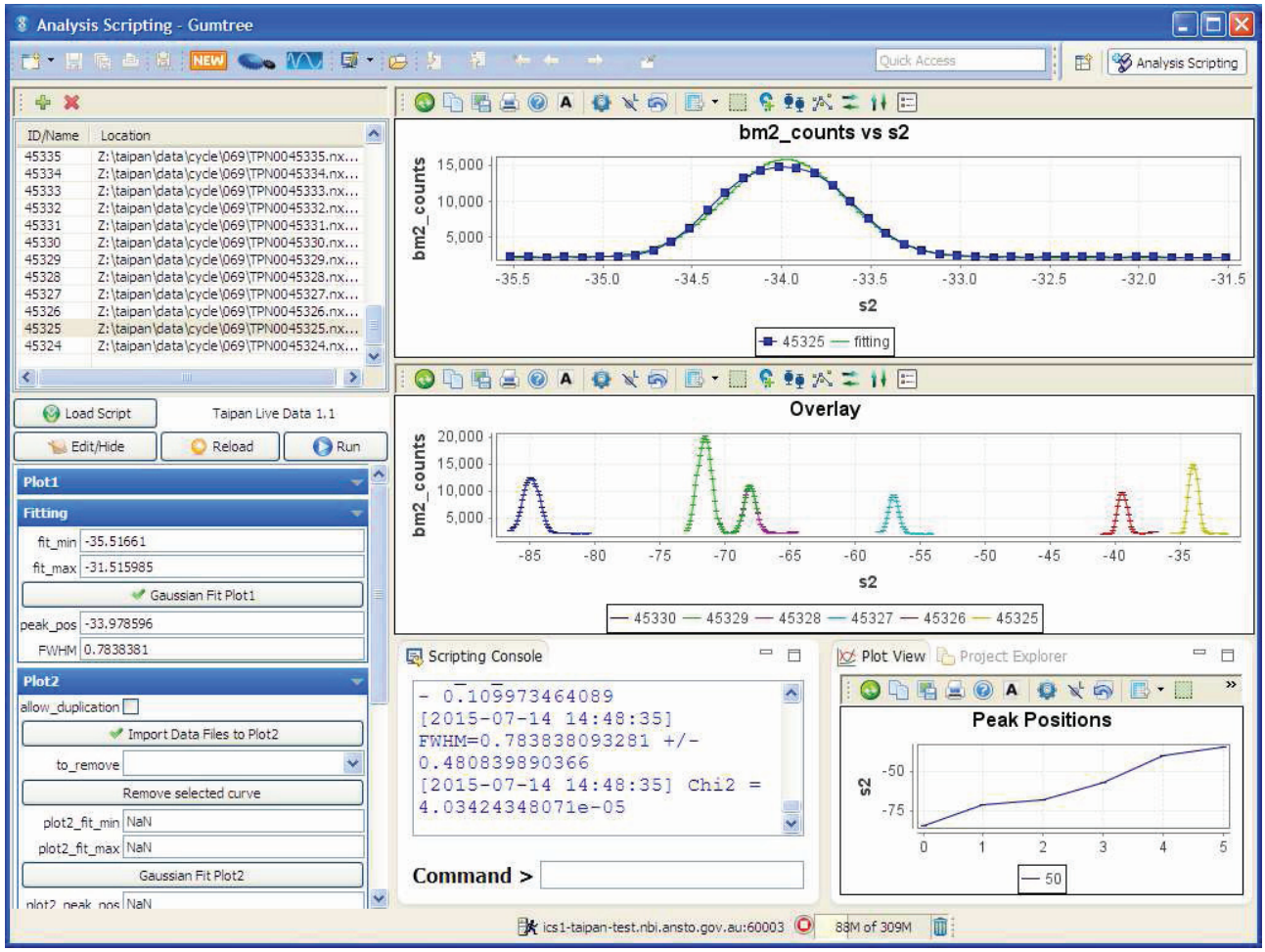


Figure 4: Calibration script for Thermal 3-Axis Spectrometer in Gumtree application.

Another example of using Python scripts in Gumtree is the implementation of the multi-sample workflow feature. This feature significantly improves the usability and efficiency of the Small Angle Scattering instruments in ANSTO. Python scripts play key technical roles in the software engineering of this feature.

Besides software developers, instrument scientists and users can quickly write Python scripts to do some ad hoc tasks on the instruments. For example, when new sample environment components are added to the instrument, it is more convenient to write a script on the fly to use them in experiments.

Python scripts are also used to provide data treatment features in both the Gumtree desktop applications and the Gumtree web applications. Data reduction applications of the Powder Diffractometer, the Strain Scanners, the Ultra-SANS, etc., are written in Python scripts.

In conclusion, Pythons scripts are used in the following actions:

- Preview data and send feedback to experiment
- Treat data on a local computer or on the server
- Align instrument
- Create quick experiment interfaces
- Make highly customisable features

CONCLUSION

The Python scripting feature in the Gumtree software plays an important roll in experiment control and data treatment for neutron scattering experiments at ANSTO. The *numpy*-like interface to Dataset, auto-GUI creator and the generic rendering feature are often used features that make it easier for users to write powerful scripts that can run in both desktop and web applications. The development and operations of Gumtree client and server for 9 instruments are supported with approximately 1 FTE with capacity to support more instruments in the future.

REFERENCES

[1] Gumtree website: https://github.com/Gumtree
[2] T. Lam, N. Hauser, A. Gotz, P. Hathaway, F. Franceschini, H. Rayner, “GumTree. An Integrated Scientific Experiment Environment”, Physica B 385-386, 1330-1332 (2006)
[3] S. Poirier, A. Buteau, M. Ounsy, C. Rodriguez, N. Hauser, T. Lam, N. Xiong, “Common Data Model Access; A Unified Layer To Access Data From Data Analysis Point Of View”, Proceedings of ICALEPCS 2013, Grenoble, France, pages 1020-1023 (2013).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PARAMETERS TRACKING AND FAULT DIAGNOSIS BASE ON NoSQL DATABASE AT SSRF*

Y.B. Yan, Y.B. Leng#, L.W. Lai, Z.C. Chen
Shanghai Synchrotron Radiation Facility (SSRF)
Shanghai Institute of Applied Physics, Chinese Academy of Sciences
Shanghai 201204, P.R. China

Abstract

As a user facility, the reliability and stability are very important. Besides using high-reliability hardware, the rapid fault diagnosis, data mining and predictive analytic s are also effective ways to improve the efficiency of the accelerator. A beam data logging system was built at SSRF, which was based on NoSQL database. The logging system stores beam parameters under some predefined conditions. The details of the system will be reported in this paper.

OVERVIEW

Shanghai Synchrotron Radiation Facility (SSRF) is one of the advanced third generation light sources in the world, and it supports and pushes the cutting-edge scientific research and the innovation in China. It is composed of a 150 MeV linear accelerator, a 3.5 GeV booster, a 3.5 GeV storage ring, more than ten beamlines and experimental stations.

Due to the large amount of devices in the accelerator system, the era of big data is in fact coming with the rapid development of the related technology. Owing to the significant relationship among the data acquisition, storage and analysis, scale effect of big data has brought great challenges to the current data storage and analysis. Although the distributed data acquisition and processing system has been widely used, the distributed data storage and analysis has not been achieved due to the constraint of storage, network, and others. Some preliminary attempts have been making at SSRF, mainly based on the beam instrumentation and control system. This work would unearth more value of runtime data, which would also improve the stability and reliability of existing accelerator system.

SSRF beam instrumentation system consists of more than 200 devices, which covered the beam position measurement, beam charge & current measurement, beam size & length measurement, fill pattern measurement and so on [1]. All these parameters are very important during the accelerator commissioning, operation and machine studies. More than 20k scalar process variables and hundreds of 2k-points waveform records are published online per second. With proper storage and analysis tool-kits, these data could be invaluable. Otherwise the potential of various advanced electronics will be wasted.

On the other hand, various hardware and software failures have been recorded in the past few years, such as global orbit disturbance, random glitch or offset jump of individual position readings [2]. All these failures affected the reliability and stability of the accelerator system. There are no effective tools to analyze the reason due to lack of adequate raw data. The regular sampling rate of achieved data is about one hertz. History of broadband data such as turn-by-turn (several hundreds kilohertz) orbit data or bunch-by-bunch data are required in this case. Due to the huge size, the data are not likely to be stored periodically. A logging system was designed and developed, which stores the broadband data under some predefined conditions.

SYSTEM ARCHITECTURE

The data logging system is based on the Couchbase [3], which is an open source, distributed NoSQL database. It provides key-value or document access with low latency and high sustained throughput. The system architecture is shown in Figure 1.

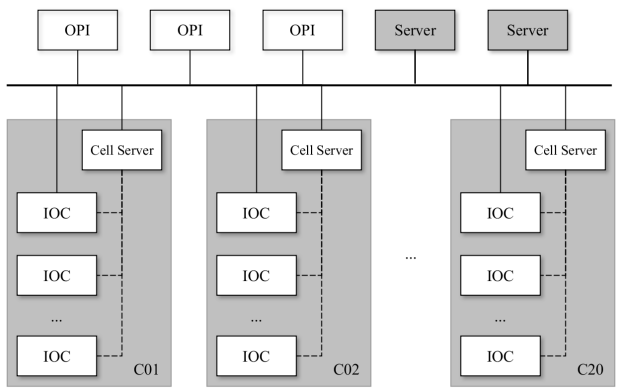


Figure 1: The architecture of data logging system.

Most part of the work is in process, especially the distributed data storage and analysis. The centralized part is mainly discussed below. In this system, IBM System x3550 M4 server and IBM Storwize V3700 storage system are adopted, which is a cost-effective option to achieve high performance. All the software run on the Linux operating system, and written using python, a widely used general-purpose, high-level programming language.

* Work supported by the Knowledge Innovation Program of Chinese Academy of Sciences
# lengyongbin@sinap.ac.cn

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Pre-processing

For the particle accelerator, the data can be divided into two categories, hardware device related and beam related parameters. The hardware device related (such as vacuity, magnet current, undulator gap or shift, etc) and scalar beam parameters (such as beam current, close orbit, etc) have been achieved in the channel archiver. The logging system mainly stores the waveform records, including the raw data (such as turn-by-turn orbit data, bunch-by-bunch charge data, synchrotron light image, etc) and processed data (such as beam spectrum data, tune or beta function measured, etc).

A lot of data pre-processing algorithms, such as the correlation analysis, the cluster analysis and the principal component analysis, can be used to extract the useful information from the raw data. Most of the scripts are implemented using python. Beside the standard library, python supports a large number of 3rd party libraries. It makes the algorithms can be implemented easily.

Data Storage

In the current system, data are obtained under some predefined conditions, such as the global orbit disturbance, beam injection, etc. The dedicated routines are used to decide whether the conditions have been satisfied. All data under the predefined conditions are packaged and stored into the database as an entry. In order to reduce the size, the bzip2 is adopted, which is a free and open-source file compression program and uses the Burrows-Wheeler algorithm.

The distributed data storage is important for broadband data. It is the basis of high-performance parallel and distributed computing (per cell). The related research work is being carried out, which mainly focus on the Ceph [4] object store and the Gluster [5] file system.

Data Mining

The signals from various probes are different aspects of a single measurement procedure. The correlation analysis of the overall signals fits the characteristic function such as the beta function and the dispersion function to the data, which would effectively increase the usage of the original information and promote the accuracy, reliability and feasibility of the results [6].

The principal component analysis finds a small number of uncorrelated principal components that can account for the maximum amount of observed variances and covariances in the data. Each principal component is a linear combination of the observed signals and retains the maximum variance along its direction. It can be achieved by a singular value decomposition of the data matrix, such as all the turn-by-turn orbit data of the storage ring. The spatial and temporal vectors can be used to identify the betatron motion, energy motion and others, such as electronics noise [7, 8].

Result Representation

It is worth paying attention to represent the result more effectively. According to the actual demand, the report files could be automatically generated daily, or at some specific time. The matplotlib and reportlab library are adopted in python. The former is a plotting library and can save plot to image file instead of displaying it. The latter library allows rapid creation of portable document format documents. The generated files can include text (such as title, comments, calculation results, etc) and the previous generated images.

As a user facility, the reliability and stability are very important. Before the failure occurs, if the qualitative or quantitative forecasting based on the achieved data can be made, it will effectively extend the mean time between failures of the accelerator. Especially for some slow drift, the early warning will be much helpful for the operators and physicists to optimize machine parameters.

PARAMETERS TRACKING

The beam parameter tracking is very useful for the accelerator operation or machine studies, such as the tune. Now the storage ring is operated in top-up mode, which improve the efficiency and quality of synchrotron light. The top-up injections are made continuously at the time interval of about ten minutes; each injection cycle takes about ten seconds. The tune can be archived during the injections.

The tune is extracted from the excitated turn-by-turn orbit data, which is the amplitude of resonance peak of betatron oscillation. The predefined condition is the gate signal of top-up injection and one entry is stored. Figure 2 shows tune drift over one hundred days. It shows that the tune is relatively concentrated in some continuous days, but scattered in different operating periods.

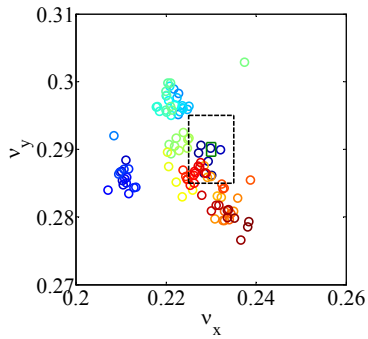


Figure 2: The tune drift during one hundred days.

The beam position monitor (BPM) system at SSRF was fully equipped with Libera Electrons. During the summer shutdown of the last few years, several units were upgraded to Libera Brilliance. Meanwhile, some other hardware and software optimization was carried out. After the upgrades, the stability and performance have been significantly improved. The track changes are shown in Figure 3.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

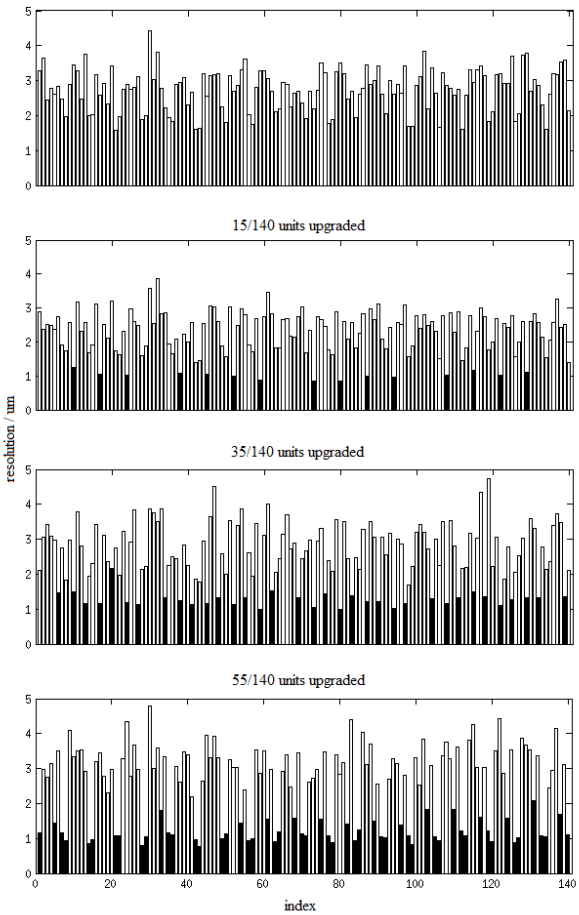


Figure 3: The track changes of electronics upgrade.

FAULT DIAGNOSIS

The particle accelerators are complicated system, with a large number of various components. The fault detection and diagnosis are a difficult task, and the data logging system will be helpful.

There is a typical example happened after a summer shutdown. The beam position monitor cables and part of the electronics were upgraded. As mentioned above, the beta function of the storage ring can be stored using some data pre-processing algorithms. But the data is abnormal at one position, shown in Figure 4. Finally the cause was found out, which is a cable connection error. The neighbouring cables (channel C and D) were cross-connected by mistake.

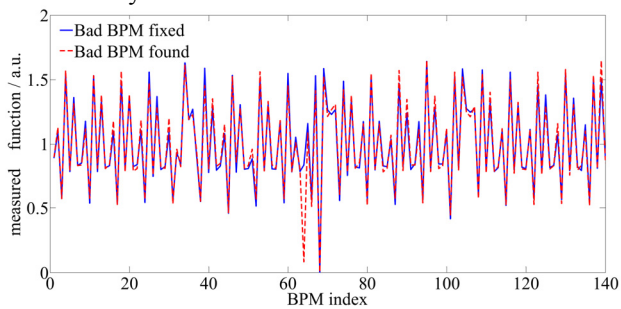


Figure 4: The beta function measured.

CONCLUSION

The beam data logging system has been designed and developed base on NoSQL database. More functionality will be added in the future. The preliminary applications show great potential for the parameters tracking and fault diagnosis. This will improve the efficiency of the operators and physicists.

With the increasing complexity of particle accelerator, the reliability and stability will become more and more crucial. Besides using high-reliability hardware, the rapid fault diagnosis, data mining and predictive analysis s are also effective ways to improve the efficiency of the accelerator.

ACKNOWLEDGMENT

We would like to thank everyone who contributed to this work through discussions and suggestions.

REFERENCES

[1] Y. B. Leng, K.R. Ye, et al.,"SSRF Beam Diagnostics System Commissioning", DIPAC'09, Basel, Switzerland (2009).
[2] Y. B. Leng, Y. B. Yan, et al.,"Beam Diagnostics Global Data Warehouse Implementation and Application at SSRF", IPAC'11, San Sebastian, Spain (2011).
[3] Couchbase website: http://www.couchbase.com
[4] Ceph website: http://www.ceph.com
[5] GlusterFS website: http://www.gluster.org
[6] Z.C. Chen, Y. B. Leng,"Study of Correlation Analysis of Global Data from a Storage Ring", Chinese Academy of Sciences, 2014.
[7] J.H. Chen, Z.T Zhao,"Preliminary Application of Turn-by-Turn Data Analysis to the SSRF Storage Ring", Chinese Physics C, Vol. 33, No. 7, 2009.
[8] N. Zhang, Y. Yang, et al.,"Application of Model Independent Analysis Based Method to Accelerator Bunch-by-Bunch Research", High Power Laser and Particle Beams, Vol. 26, No. 3, 2014.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

FPGA FIRMWARE FRAMEWORK FOR MTCA.4 AMC MODULES*

Lukasz Butkowski, Tomasz Kozak, Bin Yang, DESY, Hamburg, Germany
Paweł Prędki, DMCS, Lodz University of Technology, Lodz, Poland
Radoslaw Rybaniec, ISE, Warsaw University of Technology, Warsaw, Poland

Abstract

Many of the modules in specific hardware architectures use the same or similar communication interfaces and IO connectors. MicroTCA (MTCA.4) is one example of such a case. All boards: communicate with the central processing unit (CPU) over PCI Express (PCIe), send data to each other using Multi-Gigabit Transceivers (MGT), use the same backplane resources and have the same Zone3 IO or FPGA mezzanine card (FMC) connectors. All those interfaces are connected and implemented in Field Programmable Gate Array (FPGA) chips. It makes possible to separate the interface logic from the application logic. This structure allows to reuse already done firmware for one application and to create new application on the same module. Also, already developed code can be reused in new boards as a library. Proper structure allows the code to be reused and makes it easy to create new firmware.

This paper will present structures of firmware framework and scripting ideas to speed up firmware development for MTCA.4 architecture. European XFEL control systems firmware, which uses the described framework, will be presented as example.

INTRODUCTION

The MTCA.4 standard is derived from the Advanced Telecommunication Computing Architecture. It is enhanced for Rear I/O and Precision Timing and offers a compact environment for transmission and parallel processing of large amounts of data. The mechanics and connectivity is defined by the standard PICMG MTCA.4 specification [1,2]. Main modules are called Advanced Mezzanine Carriers (AMC). Each of them can be paired with the Rear Transition Modules (RTM); front-to-rear communication and signal transfer is handled over the so-called Zone 3 region. The basic architecture follows the idea of a centralized powerful processing unit that is connected to various AMC I/O boards over several PCIe lanes, dedicated trigger lines, clock lines and platform related management lines. The AMC backplane offers also 4 ports for low-latency-links connections (eight differential pairs, full-duplex) that can reach up to 10 Gbit/s allowing the boards to communicate using serial transmission (e.g. utilizing the MGTs).

The main function of the AMC modules is to provide communication interfaces and perform digital signal processing. All I/O lines are connected to the FPGA chip on AMC board, shown in Figure 1. Depending on the application and the used AMC, different RTM, FMC or front I/O modules can be connected. Thanks to flexibility of the FPGAs, many different applications can be run on one AMC, still using the same hardware resources If the support of hardware is already provided in the code, it can be reused in many applications. The MTCA.4 firmware framework introduces an additional abstraction layer that separates the hardware dependent logic from user application logic. The framework specifies universal interfaces on this layer. This allows the same firmware and software components to be reused, irrespective of the type of the used hardware. It means that one application logic can be run on different AMC modules.

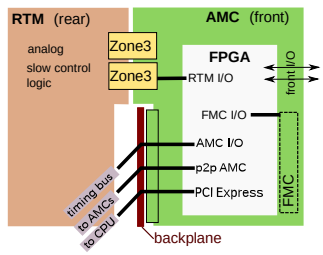


Figure 1: MTCA.4 front and rear modules

For the European XFEL all AMC boards will have one common feature: Xilinx FPGA chips, which perform, among other tasks, communication with the CPU as well as data acquisition and data processing required in control systems. All firmware algorithms are written using hardware description language (VHDL) code.

In the following sections of this paper the structure of a typical VHDL project used at European XFEL MTCA.4 [3] systems is presented.

GENERAL STRUCTURE

The code for the FPGA firmware was divided into tree main parts: board, board payload and user application.

The board section is hardware dependent and is specific for a given board. One AMC has one board support package (BSP). It has all components for configuration of peripherals on the board, such as clocks, I/O buffers, Analog to Digital Converters (ADCs), Digital to Analog Converters (DACs), etc. It is responsible for implementing communication interfaces with other boards and with the CPU. It also includes Direct Memory Access (DMA) and Data acquisition (DAQ) on DDR memory chips. The board part of the firmware provides interfaces for all resources on board.

Board payload is a middle layer section, connecting the board interfaces with the application. It contains all the

*Work supported by DESY MSK group.
#lukasz.butkowski@desy.de
ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

possible interfaces that the selected hardware can provide, allowing the user to choose which of them will be used in the application. However, some applications may require interfaces which are not provided by a specific board. In such cases, the application algorithm needs to be adapted to the hardware or interface adapters need to be implemented.

The Application section contains user specific algorithms. This part does not interact with hardware directly and is not strongly dependent on it, although it might use interfaces which are available in hardware (board part).

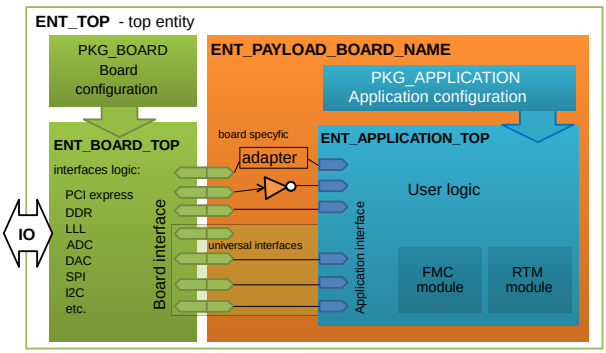


Figure 2: VHDL code block structure

Board and payload parts are connected in one top entity with all available interfaces that board can provide. One layer deeper, the application interfaces with the payload part, selecting the needed resources. Block diagram of top VHDL structure is presented in Figure 2.

INTERFACES

Between board and application we can distinguish two types of interfaces: universal and board specific. Definitions of universal interfaces are common for all boards. The include:

- **IBUS** – internal memory access bus, used to access registers and memory areas in FPGA from CPU, always connected to communication interface;
- **DAQ** – data acquisition bus, allows to write stream of data to external memory. This data is accessed using Direct Memory Access (DMA) over communication interface;
- **LLL** – low latency links bus, middle layer of point to point communication between boards using MGT;
- **FMC** – FPGA Mezzanine Card I/O;
- **RTM** – Rear Transition Modules I/O;
- **AMCIO** – backplane differential signals for clocks and triggers;
- **CLK** – clock resources.

All standard interface signals are grouped in records, which share the same name for all of the boards. This makes it easy to connect the same components in different applications. It is allowed to have more then one interface of the same type in one board. There can be a few independent DAQ channels or several point to point (LLL) connections available. In this case records are grouped in arrays.

Board specific interfaces include front I/O, ADC and DAC chips as well as any other board logic that should be controlled or accessed by application algorithms.

Communication with hardware and data acquisition is one of the most important and common task for all devices used for control systems. This is why the IBUS interface is mandatory on every application in MTCA.4 firmware framework.

Communication Interface

The data transfer between the AMCs and the CPU in an MTCA.4 crate is done using the PCI Express interface. Each board support package includes the same implementation of PCI Express transaction layer interface adjusted to the specific type of FPGA chip. It allows to access memory and memory-mapped register space on AMC board from CPU using the same abstraction layer. PCIe interconnect provides two independent IBUS interfaces. The first one connects PCIe Base Address 0 (BAR0) with board part registers address space while the second one connects PCIe BAR1 with application registers address space.

Address Space

Each VHDL module in firmware has an independent register address space definition: board logic registers, application registers and RTM, FMC, algorithms modules registers in applications. Access to those registers is done using the IBUS port on each module.

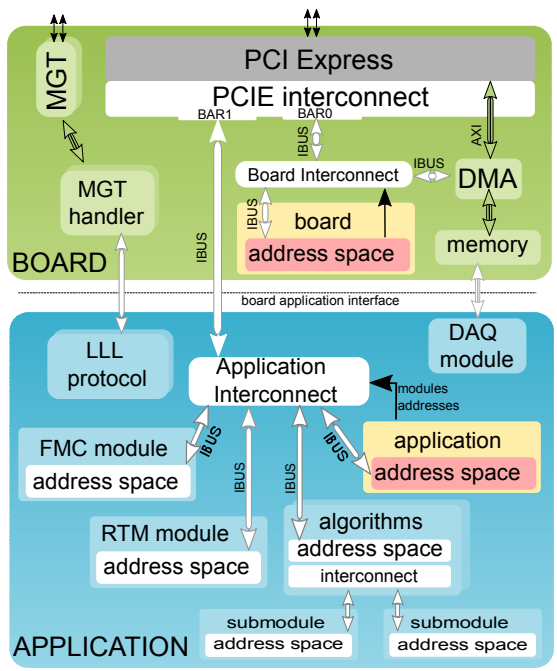


Figure 3: Block diagram of address space and interfaces connection of standard MTCA.4 firmware

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

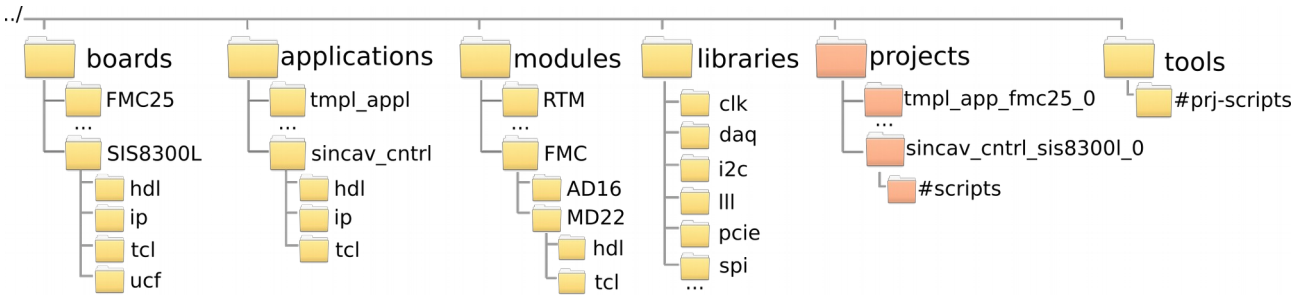


Figure 4: MTCA.4 firmware framework folder structure

This port can be directly connected to the IBUS interface from PCIe. Using interconnect many independent modules can be connected to the same bus. Information about offset of module address is defined in the board or application address space.

Main interfaces connection with modules and address space distribution is presented in Figure 3.

FOLDER STRUCTURE

Firmware files are grouped in folders based on code structure and functionality. The first level divides files into the following groups: boards, applications, modules, libraries, projects and tools. Each board support package has its own location inside the *boards* folder. The same rule applies to *application* and *modules*. Modules are additionally grouped by type of module. Structure of firmware framework is presented in Figure 4.

At the deepest level, the files are grouped by type into the following categories:

- **hdl**: all source code files written in VHDL or Verliog;
- **tcl**: scripts dedicated for source files under hdl;
- **ip**: Intellectual Property (IP) cores definition files;
- **ucf**: constraints files.

Projects folder contains the generated projects that connects all the other components together. *Libraries* as well as *modules* are shared over all projects. The tools folder contains all the general framework scripts.

AUTOMATION

The connection of board, application and modules with their configuration packages under one top entity is considered project. Project generation follows the idea that components are independent and have separate source code files list. The creation of projects is automated and done using scripts.

For firmware framework automation Tcl scripting language was chosen [4]. It is an easy, powerful and well-documented scripting language available on the majority of software platforms. It is also natively supported by Xilinx toolschain.

Tcl Scripting

Every module has its own Tcl script containing a list of source files comprising this module. This file is executed during project creation. Based on project and configuration packages variables different files can be added or additional steps can be executed, e.g. adding an automatically generated version file. Xilinx installation comes with a customized Tcl distribution called *xtclsh*. It has dedicated commands for Xilinx design flow [5].

Depending on their function, the main Tcl files included in the framework are:

- main.tcl – main Tcl script shared for all projects, main work of automation is done here, located in the tools folder;
- project.tcl – defines project constants, information about board application and its configuration packages, placed under #scripts folder of project location;
- board.tcl – defines hardware platform such as FPGA chip type and project properties, adds all board support package files to projects;
- application.tcl, module.tcl, lib.tcl – defines the user logic, modules and libraries source files to be added to the project.

Project Generation

Based on the project information and the source files lists, the project file is generated automatically. In case of the Xilinx ISE tools, a .xise file is generated. Only one command has to be executed from *#scripts* folder to perform the project generation process.

The diagram of the automatic project generation using Tcl scripts is shown in Figure 5.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

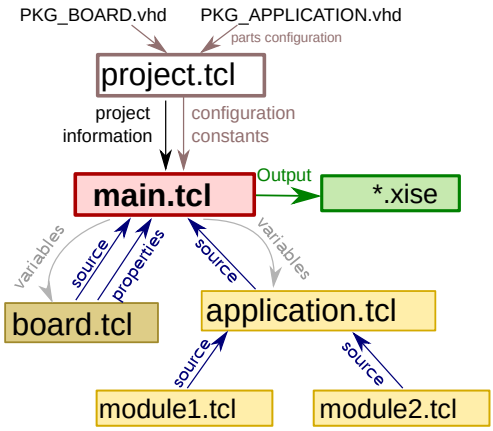


Figure 5: Project generation diagram

All the tasks are handled by the main.tcl script, which is run using Xilinx Tcl shell with the following parameters:

$xtclsh main.tcl create_project xise [ProjectName]

The script includes the projects.tcl file located in the folder given by the project name and loads the defined project variables. Those include the board and application name and their configuration packages. The packages are parsed and all the constants that are used in VHDL are accessible in the Tcl script. Base on this information, the board.tcl and application.tcl scripts are executed with access to all the variables values. Xilinx-specific Tcl commands are then used to generate the Xilinx ISE project file.

USE CASES

Presented MTCA.4 firmware framework is used for control systems at FLASH [6] and European XFEL accelerators at DESY in Hamburg. Mainly it is used at Low Level Radio Frequency (LLRF) control systems and for optical synchronization applications [7]. They are implemented using the MTCA.4 architecture. The framework was used throughout the entire process of development of the LLRF system. It helped and speed-up the hardware testing and implementation and evaluation of firmware.

One Board Many Applications

One of the most common cases is running a few different applications on the same board. A good example is the SIS8300L digitizer board from Struck [8]. Table 1 summarizes the main applications, in which this board is used at DESY.

The BSP code was successfully shared between the projects including different applications. Added features and bug fixed in the board part or in libraries of the VHDL code were immediately included in the all the projects, reducing the development time.

Table 1: Different Applications on the Same Board

Application	*RTM*	*AMC*
LLRF controller field detection part	DWC10	SIS8300L
Single cavity LLRF controller	DWC8VM1	
Toroid detection application	SIS8900	

One Application on Many Boards

Sometimes, an application needs to be run on different hardware than it was initially developed for. This is usually due to lack of available hardware or changes in the hardware version. This happens quite often in the hardware development process, where the prototype can differ significantly from the final version. This was the case for the main LLRF controller board. There are three versions of board with different FPGA chips and different I/O connections and all of them run the same application, as shown in table 2.

Table 2: One Application on Different Board Version

Application	*RTM*	*AMC*
Main controller application for LLRF distributed system.	DRTM-VM v1.2	uTC v.1.0
		uTC v.1.3
	DRTM-VM2	TCK7

CONCLUSION

The MTCA.4 firmware framework was developed and used at DESY. Proper structure and interfaces were defined as well as automation scripts were written. Firmware developed using this framework was successfully deployed for XFEL and FLASH accelerators. The framework by its modularity brings significant improvement in terms the development time.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REFERENCES

[1] http://www.picmg.org
[2] http://mtca.desy.de
[3] J. Branlard et al., "MTCA.4 LLRF system for the European XFEL" MIXDES, 2013 Proceedings of the 20th International Conference, pp. 109–112, June 2013.
[4] https://www.tcl.tk/ "Tcl documentation site [Online]"
[5] http://www.xilinx.com/products/design-tools/ise-design-suite.html ,"Xilinx ISE [Online]."
[6] J. Branlard et al., "Equipping FLASH with a MTCA.4-based LLRF system" SRF2013, Paris, France, pp. 1120-1122, (2013).
[7] U.Mavricˇ et al."Precision synchronization of optical lasers based on MTCA.4 electronics", IBIC2013, Oxford, UK, pp. 451-453, (2013).
[8] http://www.struck.de/ "Struck site [Online]."

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ENCODER INTERFACE FOR NSLS-II BEAM LINE MOTION SCANNING APPLICATIONS

Ruslan Kadyrov#, Kiman Ha, Eli Stavitski, Joseph De Long, Sung-Leung So,
BNL, Upton, New York, USA

Abstract

The NSLS-II synchrotron started operations at BNL, USA. Whereas first beamlines welcome first users, the other beamlines are under design or construction. The variety of motion control applications on existing and future NSLS-II beamlines demand custom control electronics developed to meet specific needs and ease integration into existing systems. Thus an encoder interface was designed for a number of detection techniques that require a live position capture. It implements four identical individually isolated channels with encoder input capturing and output quadrature encoder interface generated. Encoder input handles an incremental quadrature encoder with a digital RS-422A interface and output frequencies up to 10 MHz. The logic, based on Xilinx Virtex-6 FPGA, processes signals from an encoder and associates it with the machine timestamp. Several filtering and compression techniques are also applied. The device then retranslates the interface signals for the motion controller, allowing the device to be installed between encoder and motion controller with no interference to the system. Captured data is streaming to the server using TCP/IP stack, with the server side running an EPICS IOC. The device status and control registers values are also available through EPICS channel access. All operator screens are developed with Control System Studio toolset. The design harmoniously complements the family of the NSLS-II equipment sharing same mechanical and electrical platform. This paper describes the functional design, configuration, operation and current status of the encoder interface design.

INTRODUCTION

Motion scanning applications role in X-Ray experiments consists in cross correlation analysis with data obtained from various optical equipment. Position is usually detected by a quadrature incremental or absolute encoder. Delta Tau GeoBrick motion controller used in NSLS-II motor-based applications doesn't store or timestamp reported position. To keep the design simple, same encoder was decided to be used both for the axis motion control and position capturing. Thus the design should be able to retransmit the encoder signals for the motion controller without interference to the work of the latter.

The logic design concept was inspired by a similar development at Diamond Light Source Ltd [1]. The design instrumentation base shares existing hardware with several devices used for the NSLS-II accelerator FPGA-based applications: power supply controllers [2], beam position monitors [3], cell controller.

SYSTEM REQUIREMENTS

The design should provide applicability for various motion scanning applications, easy integration into existing machine control system, and compliance with the NSLS-II beamline instrumentation requirements [4]:

- The design should be able to handle a Renishaw Quadrature incremental encoder with 10 MHz operating frequency aiming to retransmit encoder interface signals for a motion controller allowing bypass connection with minimum or no interference to the encoder connection,
- Relative encoder position is to be timestamped and send to a server allowing to restore an encoder position with 10 kHz sampling rate,
- Standard EPICS driver should be used to minimize device support development needs,
- Ability to start capturing on external trigger or on EPICS PV write,
- Compatibility with existing timing system,
- Flexibility to custom experiment needs, e.g. generating triggers for external devices at certain axis positions or every N steps.

DESIGN COMPONENTS

Instrumentation Base

The design fits in a 2U 19" chassis with hardware based on standard NSLS-II platform. It is a metal enclosure with two power supplies, minimum internal wiring and place for two standard boards mounted on the bottom plate using stand-offs. Front-face mounted FPGA-based main board is in charge of purely digital part of the design, and secondary rear-face mounted interface board for device-specific circuitry. This modular design makes it possible to change functionality or build another device by putting a new interface board and developing a custom firmware for FPGA.

Hardware

The hardware leverages the NSLS-II BPM Digital Front End (DFE) board with Virtex-6 FPGA and periphery [3]. A custom interface board was developed to handle encoder interface specific circuitry. Four identical channels are implemented with each capable to receive and form the quadrature encoder interface signals.

Input and output interfaces are connected to double stacked D-sub 15-pin connectors along the rear panel of

rkadyrov@bnl.gov

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

the device. A standard Renishaw encoder pinout is used for input part and a standard Delta Tau motion controller encoder connector pinout is used for the output interface in compliance with NSLS-II beamline interface instrumentation standard [4].

Interface hardware leverage RS-422A standard with SN75ALS193D receiver IC handling input side and SN75ALS192D driver IC forming output interface. A 120Ohm termination is used to match the receiver to the transmission line impedance. Si8620BC digital isolators along with a 2 W 5 V DC to DC converter for the encoder supply and power domain isolation provide isolation for each channel individually. There is also an option to supply the encoder from a motion controller power directly if the bypass jumper is set.

Conditioned interface signals are passed to the main board to the FPGA buffered I/O through high speed backplane connectors.

Four general purpose TTL inputs and four outputs are also implemented on the board. They are connected to LEMO connectors on the rear panel of the device. The TTL GPIO is isolated and ESD protected with ADUM3402 digital isolator ICs.

Altium Designer software package is used to develop schematics for the board and PADS layout EDA used for PCB design.

Firmware Logic

The design FPGA-based logic for position capturing is implemented in Verilog hardware descriptive language with elements of VHDL using Xilinx ISE 14.7 IDE.

Being passed to the FPGA, the interface signals are processed in hardware modules. Standard modules are used for DDR3 DRAM interface, EMAC module is used for network connection.

The event receiver module (EVR) is used to pass the NSLS-II timing system interface to the logic. It provides a set of triggering events and the timestamp signal which make it possible to store position with 8 ns resolution. For test purposes, there is also an option to generate timestamp internally using on board oscillator.

The FPGA also executes a soft application running on Microblaze, a soft processor core by Xilinx. A set of control and status registers, FIFO, and memory is shared by the modules and the soft processor through the Xilinx PLB system bus and can be used for buffering. The soft processor application runs multiple threads dealing with captured data processing, status and control registers update as well as a serial console for debug and network configuration. Processed data and status registers are being encapsulated in TCP/IP packets using a Xilinx light-weight TCP/IP stack. Microblaze application boots automatically from the on-board flash on device power up. Application was implemented using Xilinx Software Development kit.

Total logic device resources utilization is 6% for registers and 13% for LUTs.

Software

Server side with EPICS soft IOC running communicates with the device through multiple socket connections. Captured position, GPIO lines state; device registers are available as PVs through EPICS channel access leveraging pscdrv, a portable streaming controller EPICS generic FPGA driver toolkit [5]. The driver also unpacks the streaming data from binary packets and put it into waveform PV records.

PV values than be written to the file using EPICS aSub routine for further plot and cross analysis together with data obtained from detectors and other optical equipment.

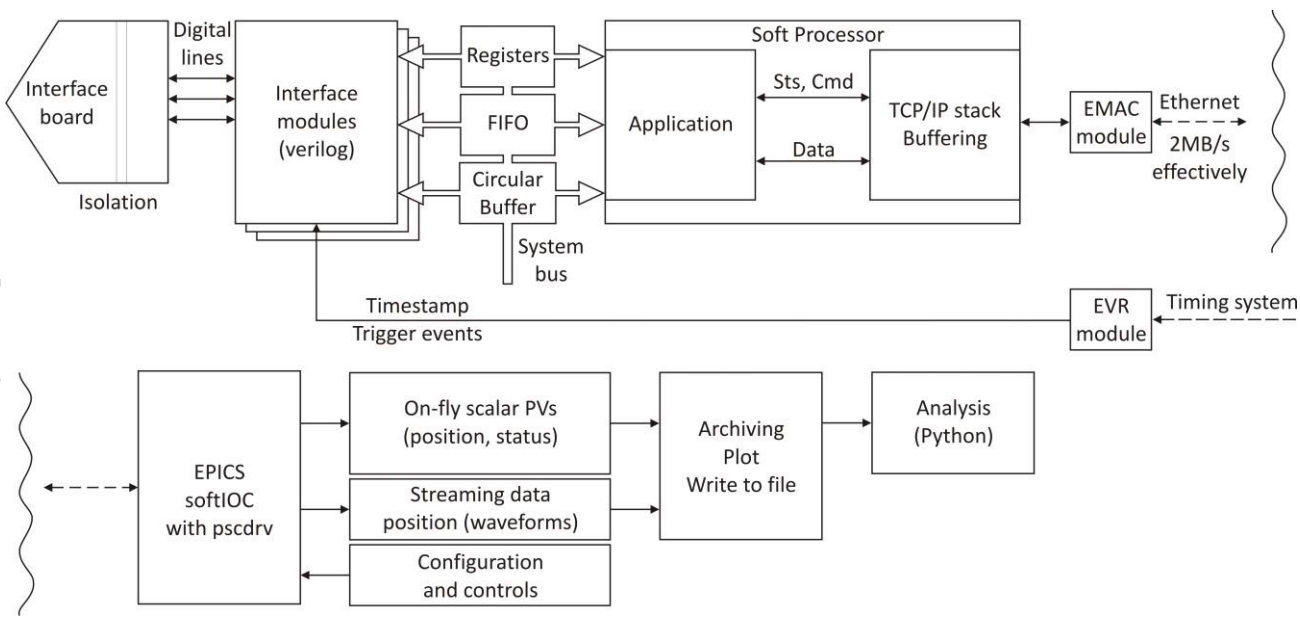


Figure 1: Simplified design dataflow

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Operator Panel

To ease work with the device registers and data representation, an operator screen was developed using Control system studio [6]. It allows user to choose operation mode as well as configure device parameters, check its status and operation statistics. Built-in Python scripts are resided for data plot and work directory management.

FUNCTIONAL DESIGN

Position Capturing

Encoder position change generates multiple pulses with 90 degrees phase offset on A and B digital lines of the interface. Quadrature decoder hardware module catches every single transition of quadrature signals using a simple hardware circuit shown in Fig. 2. Every movement detected is captured, and a relative position counter changes accordingly. A reference clock signal of 100 MHz is used to ensure oversampling for connected 10 MHz encoder. Extra D flip-flops are used to avoid metastability.

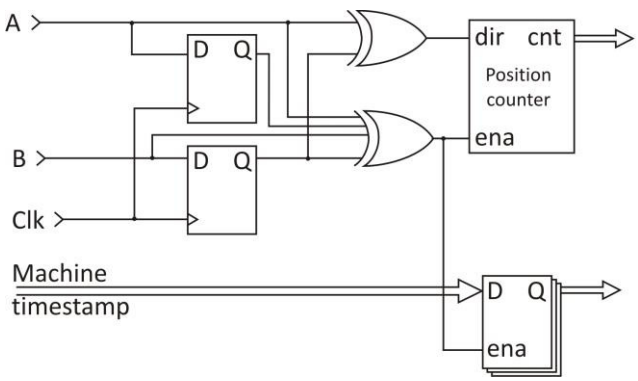


Figure 2: Position capturing simplified hardware circuit.

Encoder signals, relative position counter along with GPIO input states for the current transition are associated with the timestamp received from the timing system at the moment of movement detection. With every single transition detected a 16-byte portion of data (chunk) is captured (see Table 1). Built-in counters and status bits provide diagnostic means for missing position steps, missing pockets and encoder misconnection.

Table 1: Interface chunk data format

Bytes	Field name	Field description
15 to 12	Timestamp, sec	POSIX time of transition
11 to 8	Timestamp, ns	Nanosecond portion of timestamp, 8 ns resolution
7 to 4	Index counter	Chunk sequential number
3 to 1	Position counter	Relative position detected
0	Data byte	Interface signals state

Data byte is combined binary status of signals with A, B and reference mark Z encoder signals, encoder proper connection state followed by states of four GPIO inputs. Index counter with sequential chunk number is also added to ensure data proper processing and storing in the buffer as well as consistency in the data stream and while restoring it for the post analysis.

Circular Buffer

Data chunk with timestamp header is than stored in a circular buffer through an NPI interface. 1 GB of an on-board DDR3 DRAM is used to accommodate the buffer giving maximum read/write bandwidth 800 Mb/s. Thus 256 Mb of buffer space is available per channel. Auxiliary FIFO buffer is used between an encoder interface channel handling hardware module and the NPI for initial data accumulation. Once FIFO reports to be half-full, a relatively rapid NPI transaction reads it out.

Once a certain number of chunks are stored in the buffer, a TCP/IP packet is initiated. Chunk number threshold for a new packet is hardcoded to 1024. To ensure none of the saved chunks left in the buffer at the end of an experiment, circular buffer flush mechanism is implemented. In case of the buffer overflow, a skipping logic erases all or a part of data stored in the buffer. The buffer fill level is also available through the register value as a PV and operator panel widget.

Position Filtering

In an application with high speed movements with an encoder operating at frequencies close to the maximum required, position capturing generates enormous data volume that can overflow the circular buffer causing data skip. Also filtering should be used when the data is streaming from all four channels simultaneously due to TCP/IP stack bandwidth limitation of 2 MB/s of useful data. To use the buffer resources effectively and ensure TCP/IP stack proper operation, a number of filtering techniques are implemented. Among them, the position filtering feature leverages a window-based approach, when a new chunk is formed only if a relative position changes for a number of steps exceeding a predefined threshold level during given period of monitoring, thus the position doesn't fit in a $dY - dT$ window.

Several special use cases are possible: with null dY every single event is detected at the end of dT period. Filter with $dY = 1$ parameter catches the chunk if more than one event was detected during dT. Using dY values exceeding 3 is discouraged as it can filter out a steady drift movement. Setting a null size window disables the filter, and every single movement is registered at the moment of its detection. This can generate large volumes of useless data for fast encoders even in idle state in mechanically noisy environment. Default filter configuration is null dY with dT adjusted to the maximum encoder speed required assuming every step detected during 100 ns

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Generation: Bypass Mode

In bypass mode input encoder interface signals are mirrored inside FPGA providing a negligible delay for output interface. Bypass mode leaves reconstructed signals as they were received with no impact to the motion controller operation. There is also an option to disable generation at any mode using disable pin of an output driver IC. Bypass mode is set as default, with output drivers enabled.

Generation: Test Mode

Encoder signals pulse train generation is also implemented for the test purposes. A series of pulses of a configurable width, period and count simulating a real encoder signals transitions can be generated to provide a self-testing fixture if the output interface is looped back to the input simulating a real encoder channel.

Generation: Filter Mode

Another feature implemented implies input interface retransmission with a filtering before retransmission. A filtering window is applied to the raw position captured. If during the *dT* timing interval there was less than *dY* position offset detected, the interval is retransmitted without transitions on the lines as if an encoder didn't report any movement. It is up to the user to set an appropriate *dY* value in order not to filter out a useful position changes, e.g. a continuous drift. Setting *dY* to zero allows generation exact number of steps to catch up the real position at the end of previous *dT* timing interval.

Generation can be performed with a different rate then the real encoder is operated on, faster or slower. If regeneration rate exceeds an input rate, an overflow may occur when a filtered position cannot be reached on a lower rate.

The mode can be used to filter out unnecessary micro movements in noisy mechanical environment as well as roughing an encoder with an excessive resolution.

Control and Status Registers

Status update soft processor thread is continuously running transmitting status registers with 1 Hz update frequency. Circular buffer load and skip counter indicators, relative encoder position, GPIO inputs state, and device timestamp are examples of status fields.

Control packets are sent on demand. Trigger PV is toggled to initiate transaction if a writing request to one of the controls registers is received. Position and generation filters configuration, encoder direction reverse bit, timestamp source, generation enable bit are examples of device controls.

TTL Triggers and GPIO

Auxiliary digital TTL I/O is implemented for external triggering. Every transition of an input is timestamped and stored in the circular buffer as a part of an encoder chunk. Output lines state are mirrored from the register value allowing user to toggle an output state by writing to a PV or configuring soft processor application for user-specific needs such as triggering an external device at certain relative positions of an encoder.

SUMMARY

The product implements position capturing for motion applications with quadrature incremental encoders. Mechanical 2U 19" platform, internal wiring and Virtex-6 FPGA based main board are shared with other devices used on the facility. The encoder signals are passed to FPGA and then re-transmitted for the motion controller, allowing the device to be installed between encoder and motion controller with no interference to the system.

Ability to supply the encoder directly from the board, as well as generating internal timestamp and output encoder interface make it possible to use the design without direct connection to the machine infrastructure.

The design is in production state with first series of eight pieces built and tested waiting to be used in scanning applications at Inner Shell Spectroscopy (ISS) and Soft Matter Interfaces (SMI) beamlines.

Similar design concepts are going to be realized in a new design of general purpose data acquisition and control board. This and future design harmonically complements and extends application variety for the device family at the NSLS-II experimental floor.

REFERENCES

[1] T. Cobb et al., "Zebra: a Flexible Solution for Controlling Scanning Experiments", ICALEPCS'13, San Francisco, CA, USA, October 2013, TUPPC069, p. 736 (2014); http://www.JACoW.org.

[2] W. Louie et al., "NSLS-II Power Supply Controller", PAC'11, New York, NY, USA, March 2011, TUP193, p. 1187(2011); http://www.JACoW.org.

[3] Om Singh et al., "NSLS-II Beam Position Monitor Embedded Processor and Control System", ICALEPCS'11, Grenoble, France, October 2011, TUBL1, p. 316(2013); http://www.JACoW.org.

[4] A. Broadbent, W. Lewis, D. Chabot, "Beamline Systems Instrumentation Interfacing Standard" NSLS-II LT-C-XFD-SPC-CO-IIS-001 Version 5, April 2013.

[5] https://github.com/mdavidsaver/pscdrv.

[6] http://cs-studio.sourceforge.net/.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Em# PLATFORM: TOWARDS A HARDWARE INTERFACE STANDARDIZATION SCHEME

O. Matilla, J. Avila-Abellan, M. Broseta, G. Cuní, D. Fernández-Carreiras, A. Ruz, J. Salabert, X. Serra-Gallifa, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain

Abstract

Low current measurements developments have been pointed out as strategic for ALBA synchrotron. From the first day of operation of the seven Beamlines currently in operation ALBA Em electrometer has been successfully used. Today, the two new beamlines of Phase 2 that are being constructed and the new end stations have required specification changes in terms of increased accuracy, capability of synchronization, timestamping, management of large buffers and high performance closed-loop implementation. The scheme of full custom hardware design has been abandoned. ALBA Em# project approach has been based in the selection of industry standard interfaces: FMC boards design for custom needs, FMC carrier over PCIe using SPEC board from CERN under OHWR license, and Single Board Computer using PCIe to implement interfaces with the control system. This Paper describes the new design of the Electrometers at Alba, suited for the newer requirements, more flexible, performing and maintainable, which profits from the know-how acquired with previous versions and suits the new data acquisition paradigm emerged with the standardization of quick continuous scans and data acquisition.

INTRODUCTION

ALBA is a 3GeV third generation synchrotron light source located in Cerdanyola del Vallès (Barcelona). ALBA houses 7 commissioned beamlines and 2 more are being constructed in the facility construction Phase 2.

Low current measurements at ALBA for diagnostic or experiment applications have been performed thanks to the successful development of a 4-channel electrometer (ALBA Em) [1]. The project started in 2011 and more than 40 units are currently working at ALBA facility since then. Due to the increase of needs mainly in the experimental area an equipment redesign need raised in 2013 (Em#) [2]. Since then the project philosophy has evolved from the full custom hardware design to be based, as much as possible, in commercial components using expected long-life interfaces. The inclusion in the equipment of a Single Board Computer (SBC) integrating part of the control system increases the complexity of the project and positively forces the incorporation of software developers in the project development stage.

With the common objective of fostering the development of Em# project a collaboration agreement has been signed between ALBA and MAX IV. The agreement could be open to other institutes or facilities interested in the project.

HARDWARE APPROACH: DON'T SELECT THE COMPONENTS, SELECT THE INTERFACES

One of the lessons learned from first ALBA Em is that the obsolescence of commercial modules (like microprocessor cores) is a crucial factor in the equipment design. If an electronic module gets obsolete and its interface is not standard, the module replacement implies a dramatic effort of hardware redesign.

The project new approach is based first on selecting the industry standard interfaces intended to be used between the modules included in the equipment, and second selecting the modules that could fulfil these requirements. In this way, if a module gets obsolete, it can be replaced by another one using the same interfaces, and the redesign effort is lower.

In the Em# project three standard interfaces have been selected:

1. Ethernet: External communication interface from the embedded SBC to the Control System.
2. Peripheral Component Interconnect Express (PCIe): Internal interface between the SBC and the FPGA module.
3. FPGA Mezzanine Card (FMC): Interface between FPGA module and custom electronics based on VITA 57 standard [3].

The block diagram of the new hardware approach (see Fig. 1) is composed mainly by 3 well-bounded blocks, where the connections between them are the cited selected interfaces. The first block is a selected commercial fanless SBC including Ethernet and PCIe interfaces with a small 100mm x 100mm long life form factor: Intel NUC (DE3815TYBE). The second block is a FPGA module including PCIe and FMC (as carrier) interfaces: SPEC, a FPGA board developed by CERN and available under Open Hardware License (OHL) [4]. The third block contains custom electronics developed for the project. Custom electronics block includes an FMC board developed at ALBA under OHL license [5]. Its core is an isolated 4-channel 400kS/s 18 bits ADC capable of operating under ground voltage bias condition. This ADC FMC board has also a custom digital interface to control the rest of electronics; for the equipment I/O and the 4 ALBA Current Amplifiers. The 4 ALBA Current Amplifiers and the ADC of the FMC are designed to work with biased voltage ground, so the isolation is a key point with many implications in the hardware design.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

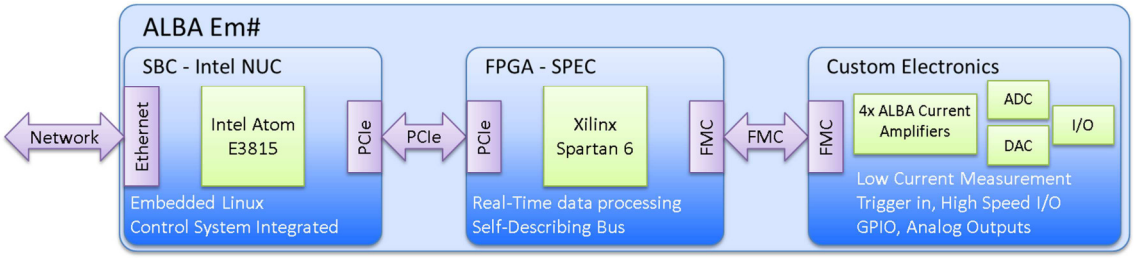


Figure 1: Em# block diagram, with the three main interfaces and elements.

MAIN FEATURES, IMPROVEMENTS AND INNOVATIONS

Current Amplifier

ALBA Current Amplifier, which is the sensing element and the equipment's core with 8 measurement ranges from 100pA to 1mA, is the only element used in this new design inherited from first ALBA Em. Its good performance, comparable with other well-known commercial current amplifiers (see Fig. 2), and its deep characterization knowledge, represents a big value in this new electrometer. An improvement to ALBA Current Amplifier is included from last version: a better thermal management in order to reduce heating and noise near the transimpedance amplifier by removing internal voltage regulator. A temperature sensor is also used for calibration and will be used for gain drifts due to heating during operation.

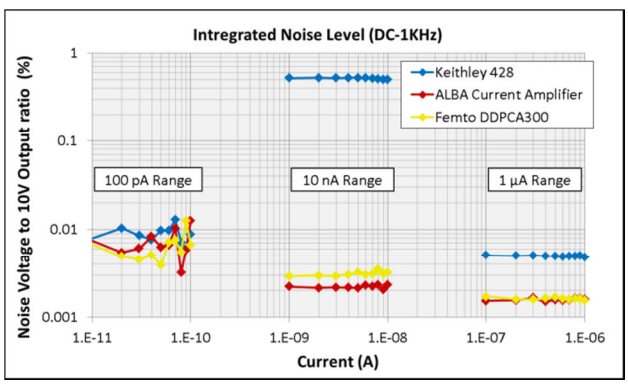


Figure 2: Integrated Noise Level comparison for 100pA, 10nA and 1µA ranges

Current Measurement with Voltage Bias

In some applications in accelerators the use of isolated ground is mandatory. Among them a few even needs a high voltage ground bias to be applied. This feature that is not widely considered in most of the electrometers has been considered in Em#. In it there is the possibility of biasing the whole detector ground with an external voltage source respect to installation ground. The part of the equipment sharing the detector ground is the part in charge of the low current measurement (ALBA Current Amplifiers and ADC), which is isolated in the custom electronics area (see Fig. 4). The input low current conversion from analog to digital is done by the ADC in the isolated area, so the digital data is sent to the FPGA through digital isolators capable of 1kV withstanding. The same strategy is followed in current amplifiers control signals. The bias between two grounds is measured in the FMC by an additional ADC, and this feedback is used by the FPGA to detect overvoltage conditions and disconnect voltage bias input, for safety reasons.

Integrated Control System

In first ALBA Em, the data acquisition control, data processing engine, high level functions and communication protocol with external control system via Ethernet were integrated in the same microcontroller. This strategy presented some difficulties and known functionality limitations very difficult to be overcome.

In new Em# the control strategy is clearly split: high-level functions and communication protocol integrated in SBC and low-level control integrated in the FPGA, both communicating via PCIe. The decision of centralizing high-level functionality in a SBC reduces the FPGA load with a relatively low cost.

The presence of SBC E3815 Intel Atom processor, which is much more powerful than last microcontroller, allows the control system to be designed based on layers. The design of control layers taking into account the whole hardware architecture from the beginning allows more flexibility when new requirements will rise in the future.

A custom Linux embedded distribution placed in the 4GB SBC Embedded Multimedia Card (eMMC) with a 5s boot time is nowadays used for the Em# working prototype (see Fig. 3). The Drivers low layer includes only needed drivers which give access to the hardware and its basic control. The Middleware layer contains functions, algorithms and the Em# control logic. In the Applications layer simple control applications for external user are included: display control and remote control through telnet or SCPI, which is the standard that specifies a common syntax, command structure, and data formats, to be used with measurement devices.

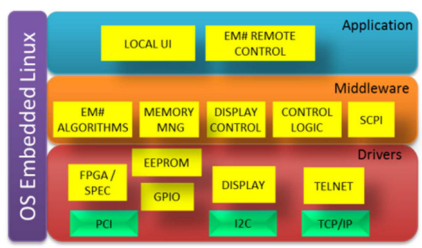


Figure 3: Control layers in SBC embedded OS Linux.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

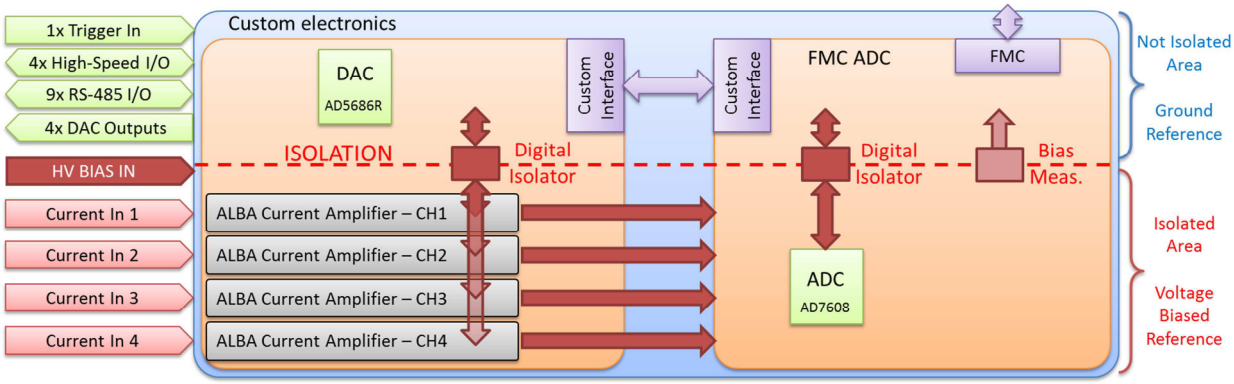


Figure 4: Custom electronics block diagram.

Large Acquisition Memory

A weak point in first ALBA Em was the limited data storage capacity. A maximum vector of 1kS per channel was not enough for experiments where continuous data acquisition was required, like quick continuous scans in spectroscopy beamlines.

This situation has been dramatically improved in Em#: the SPEC board includes a 2Gb DDR3 memory directly managed by the FPGA, allowing real-time data storage without latency problems. Data will be stored in DDR3 memory in 64 bits words format, including identifier or timestamp fields. So, up to 32MS could be stored in the memory. In an application where the 4 channels from the ADC would be stored simultaneously, up to 8MS could be stored per channel.

Intel NUC SBC offers also great storage capacity due to its 8GB DDR3 RAM memory, where the data could be transferred and stored after the end of the experiment. This data transfer from SPEC DDR3 RAM to Intel NUC DDR3 RAM is done via PCIe bus after the experiment acquisition in order to avoid latency problems, as PCIe is a shared bus for all the communications between the microprocessor and the FPGA.

Flexible Data Processing Via Dual Bus Implementation

The Spartan 6 FPGA contained in SPEC board is the core of the Em# low-level control, including key actions like data acquisition control, fast data storage control and data processing.

The FPGA firmware is based on the implementation of the Self-Describing Bus (SDB) specification, developed by CERN and GSI under General Public License (GPL) [6]. This specification enumerates and defines the Logic Cores synthetized in the FPGA (which are connected to a common 125MHz Wishbone bus), as well as defines their accessible registers from PCIe.

A dual bus strategy will be used in Em# (see Fig. 5). The SDB Wishbone bus, which is managed by the SBC and its latency is not predictable, will be used for the control and configuration of the Logic Cores synthetized in the FPGA. A secondary data bus internal to the FPGA will be used for the cores interconnection for high speed data transfer with minimum and reproducible latencies. This data bus is a 64 bits wide bus (data length), where the 32 data bits will be transmitted together with an 8 bits identifier and a 24 bits timestamp. Transmitter cores to the data bus will be configured via the Wishbone bus to have an identifier for all kind of data transmitted. Receiver cores will be configured to which identifiers should sniffer its data from the bus. Any processing core will be configured with the operation to perform. This general configuration scheme leads to a flexible processing strategy. For example: in the case of storing in the DDR3 a 64 averaged data from ADC channel 1, the ADC core would be configured to send channel 1 data with identifier 10. The math core would be configured to sniffer data with identifier 10, and average this data every 64 samples. The math core would be configured to send the average result with identifier 20. DDR3 managing core would be configured to sniffer data with identifier 20 to perform a write operation in the memory as a circular buffer.

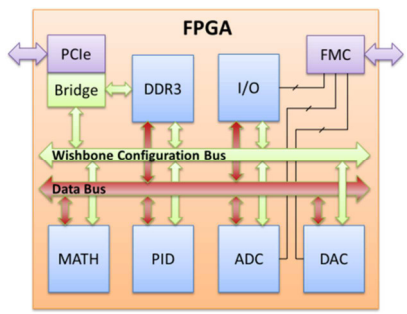


Figure 5: Block Diagram of FPGA firmware, with Wishbone Bus, Data Bus, and PCIe bridge.

Advanced Features

Based on the dual bus strategy, an important application that can be implemented is the feedback as a function of the current measured by the ALBA Current Amplifiers and the ADC. The presence of four high-speed digital outputs of up to 100MHz allows for example the implementation of Voltage-to-Frequency (V2F) application, where the signal used could be directly the ADC signal or processed by the Math Core.

The four high-speed digital signals can be configured also as inputs, for example to be used as delay-

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

configurable trigger inputs. An additional 200MHz bandwidth high-speed is always available as input trigger.

Nevertheless, the most expected application is the closed-loop implementation, via the usage of the four available analog outputs and a PID core. The PID input data could be directly the ADC signal or processed by the Math Core. The four analog outputs have a ±10V range generated with a low-glitch 16 bits with 100kHz bandwidth.

Em# also includes nine I/O configurable RS-485 signals that can work in unipolar or bipolar configuration, for connectivity with other equipment. The instrument is capable also to provide four +5V and 500mA outputs for external equipment supply.

STATUS OF THE PROJECT

During last two years the Em# project has evolved from a conceptual stage to the current situation with a new hardware approach as commented previously. Nowadays the project status is in its final stage of hardware design. Thanks to the successful implementation of low-level control Logic Cores (Current Amplifiers, ADC) with SDB and its control via PCIe a working prototype has been built (see Fig. 6).

New Em# is planned to be used in the new Phase 2 beamlines and Ids currently being installed at ALBA. Different experiments are pending of its final integration in the end station to enhance its experimental results (BL22-CLAESS, BL29-Boreas). On the other hand different beam stabilization via XBPMs and monochromator piezo actuators wants to be optimized with its use.

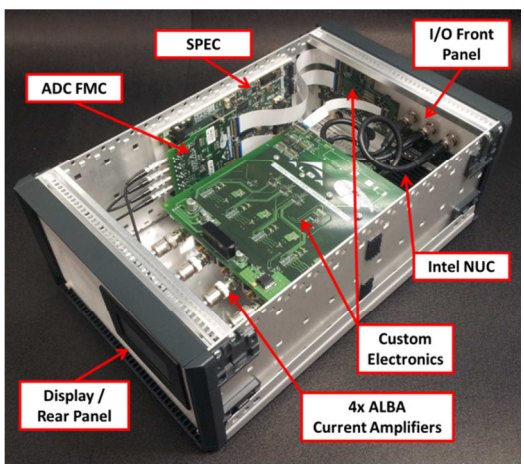


Figure 6: Em# working prototype.

Next Steps

The first target to achieve is to have the complete system hardware design closed. The first Em# working series are planned to be produced in January 2016, where the functionality of first ALBA Em will be achieved with a much higher resolution. After that the project will focus on the development of the FPGA firmware for advanced features and the control system embedded in the SBC.

CONCLUSION

After the successful conclusion of first ALBA Em project, the change from full custom hardware design to a scheme based in commercial modules using selected standard interfaces has become a successful strategy in Em#. Having the advantage of lower redesign impact due to commercial modules obsolescence, the fact of splitting the design in parts clearly separated by the selected interfaces allows a more efficient parallel tasking and an easier involvement of different development groups (hardware, FPGA firmware and software).

During last year the project has advanced with the selection of commercial modules and the design and production of custom electronics ones. The use of SDB in SPEC FPGA board and Intel NUC has started with successful results. First Logic Cores for the low-level control of the hardware have been included in the FPGA firmware as well as their control from the Intel NUC using high-level Python software. Nowadays there is a working prototype with the basic functionality of low current measurement with much improved accuracy than previous project ALBA Em. First equipment series will be produced in January 2016. After this target, where the hardware design will be closed, big effort on firmware and software development will be done to introduce important new features like big storage capability, closed-loop functionality or flexible data processing.

On the other side the collaboration agreement signed between ALBA and MAX IV for Em# project development will foster the project progress from now on due to the addition of other development groups. The inclusion of other institutes to the collaboration is open and welcome.

ACKNOWLEDGEMENT

Thanks to all the people who worked in first ALBA Em and the knowledge learned from them, the background for Em# development is well-established. Thanks to all of them and to all the people collaborating in the current project who are not authors of the present paper.

REFERENCES

[1] J. Lidón-Simon *et al.*, “Low Current Measurements at ALBA”, Proceedings of ICALEPCS’11. WEPMS025.

[2] X. Serra-Gallifa *et at.*, “Em# project. Improvement of low current measurements at ALBA”, Proceedings of ICALEPCS-13. TUPPC094.

[3] “FPGA Mezzanine Card (FMC) Standard”, ANSI/ VITA 57.1-2008 (R2010), ISBN 1-885731-49-3.

[4] http://www.ohwr.org/projects/spec

[5] http://www.ohwr.org/projects/fmc-adc-400k18b4cha-iso

[6] A. Rubini, W. Terpstra, M. Vanga, “Self-Describing Bus (SDB), Version 1.1”, 2013, www.ohwr.org/projects/fpga-config-space

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SINGLE NEUTRON COUNTING USING CCD AND CMOS CAMERAS

P. Mutti*, E. Ruiz-Martinez, M. Platz, P. Van Esch
Institut Laue-Langevin, Grenoble, France
M. Crisanti, Universitá degli studi di Perugia, Perugia, Italy

Abstract

The present paper explores the technical possibility to use classical imaging detectors such as Charge-Coupled Devices (CCD) and Complementary Metal-Oxide Semiconductor (CMOS) cameras as counting detectors at lower counting rates and transits smoothly to continuous imaging at higher rates. The gamma-rejection capabilities as well as the counting efficiency have been investigated with both an AmBe neutron source and in-beam experiments.

INTRODUCTION

Neutrons have mass but no electrical charge. Because of this they cannot directly produce ionisation in a detector, and therefore cannot be directly detected. This means that neutron detectors must rely upon a conversion process where an incident neutron interacts with a nucleus to produce a secondary charged particle. Classical neutron detectors are based upon particle detection technologies like gas filled proportional counters or scintillation detections. These detectors have a high dynamic range, and are very performing at low counting rates and fast timing (time-of-flight) applications. At high counting rates however, continuous imaging detectors such as CCD or CMOS camera's optically linked to scintillators, can have very good performances concerning linearity and spatial resolution but the dynamic range of these systems is limited by electronic noise and gamma background.

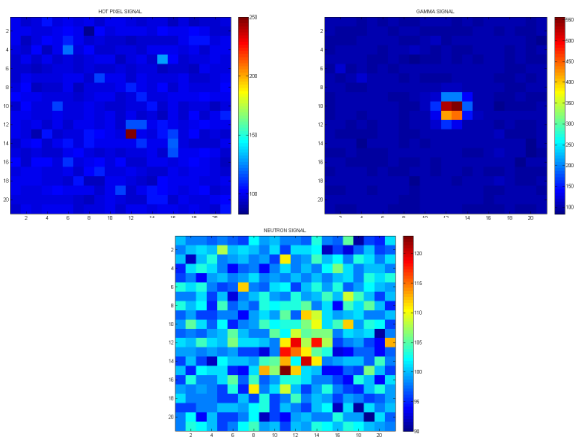


Figure 1: Different light patterns in a CMOS camera.

The capability of single neutron counting relies on the possibility to distinguish the neutron signature on the camera from those belonging to gammas or electronic noise (hot pixel). Neutron detection involves reactions releasing energies of the order of the MeV, while X-ray detection releases energies of the order of the photon energy ($\sim$10 keV). This 100-fold higher energy released in the interaction, transforms into a much different light yield allowing the individual neutron light signal to be significantly above the noise level and therefore opening the possibility for gamma-discrimination and individual counting. Figure 1 shows the different light patters obtained using our CMOS camera. The *hot pixel* is shown in the left-upper plot, the gamma detection in the right-upper and the neutron signature in the lower one.

NEUTRON RECOGNITION

^{3}He based proportional gas neutron detectors have an impressive gamma rejection ratio in the range from 10^{-6} to 10^{-8} [1]. This impressive noise rejection ratio comes from a non-linear operation, called discrimination. The noisy analogue signal is put to 0 in the end result, as long as it doesnt cross a threshold. In such a way, all the integrated noise (whether electronic noise, inherent detector noise, or gamma radiation) is eliminated entirely. If one assumes a Gaussian noise with a standard deviation σ and a threshold set at 6σ, the probability per independent sample (about every μs) to cross that positive threshold equals about 10^{-9} which gives a background rate of 10^{-3} counts/s. At a count rate of 10 neutrons per second, an integrating detector summing all the samples over a period of 1 second would have a total signal of about 60σ (assuming the individual signal level equal to 6σ) and a noise level equal to 1000σ: in other words, the signal would be entirely swamped by the noise. When used in classical integrating imaging mode, a CMOS like camera suffers a similar degradation of the final signal-to-noise ratio and, therefore, it can only really be used when the neutron fluxes are very intense, such as in neutrography, or in Laue diffraction. Normally, the camera pixel size is smaller than the size of the typical light spot from a neutron scintillator event on the camera and, therefore, one can assume that the scintillating light has a 2-dimensional Gaussian distribution as shown in Fig. 2.

We will consider a re-pixelisation where we combine the camera pixels into what we call *spot pixels*: pixels that are about the size of the FWHM of this Gaussian distribution. We take images with such short exposure times that the amount of noisy photo-electrons in a spot pixel is well below the expected number of photo- electrons corresponding to a neutron scintillation event, and we put a threshold that distinguishes them. The first problem related to this

* mutti@ill.eu

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

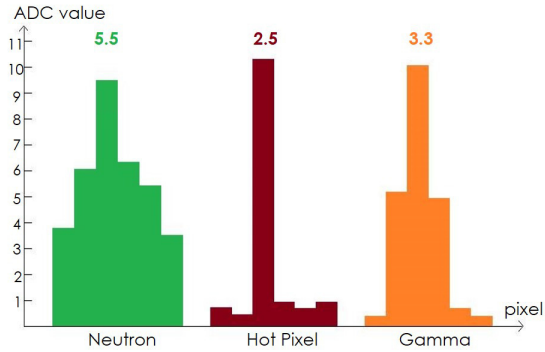


Figure 2: Pixel profiles for neutrons, hot pixels and gammas.

approach is that if more than one neutron impacts a certain spot pixel, we will nevertheless only count a single neutron. This means, essentially, that our system has a local non-paralysable dead time which is of the order of the exposure time. The second problem is that the neutron impacts will not coincide with the centres of the spot pixels. To solve the second problem, we assume that the probability of the location of impact of a single neutron on a spot pixel is uniformly distributed over the area of that pixel. We then pick a threshold so that statistically, we miss just as many neutron impacts as we double count others. This hypothesis is valid if the resolution given by the spot pixel is smaller than the features in the image. This is the case if the detector is not the resolution-limiting part of the instrument.

SIMULATION

A Monte Carlo simulation is performed in which a constant amount of light is distributed according to a Gaussian profile with a FWHM equal to the width of individual spot pixels, and with uniformly distributed centre position. Varying the threshold value as a percentage of the intensity of a neutron flash, the ratio of the number of neutrons counted over the number of neutron flashes simulated, is shown in Fig. 3.

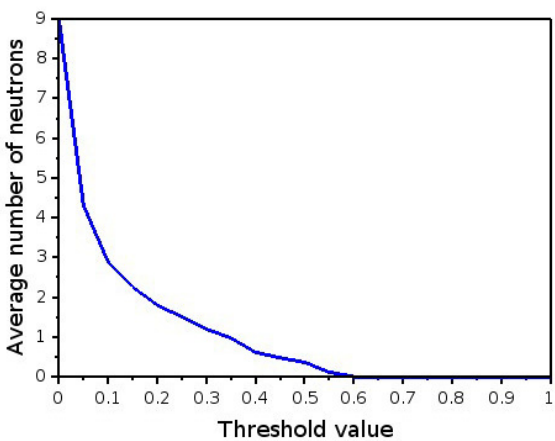


Figure 3: Average number of neutron counts for a single neutron impact of light intensity as a function of the threshold value.

As expected, for a low value of the threshold, neutrons are over-counted, namely a single neutron will trigger a count in several neighbouring pixels. For a high value of the threshold, not all neutrons are counted. But for a threshold equals to 34% of the full intensity, on average, one neutron is counted per neutron impact. More precisely, about 80% of the neutrons are correctly counted as a single neutron, while about 10% is not counted, and another 10% is counted twice. To tackle the first problem, we propose to use a function $f(e)$ that indicates, how many neutrons we should count as a function of the amount of photo-electrons in the spot pixel. Its asymptotic behaviour should be:

$$\lim_{e\to\infty} f(e) \to \frac{e}{e_n} \tag{1}$$

where e_n is the average total number of photo-electrons per single neutron impact. Near the threshold value te_n the function should be near 1. We have considered 2 different functions. The first equals the integrating value if the charge in a pixel is larger than a full neutron charge, otherwise, the threshold fraction t of e_n is applied to decide if one should count a neutron or not:

$$\begin{aligned} f(e) &= [\frac{e}{e_n}] \quad ; \quad e > e_n \\ f(e) &= 1 \quad ; \quad e_n > e > te_n \\ f(e) &= 0 \quad ; \quad te_n > e \end{aligned} \tag{2}$$

This function implements most closely the separation between single neutron counting and continuous charge integration. The second function is smoother, and applies the following prescription:

$$f(e) = \frac{e}{e_n} + (1 - t) \tag{3}$$

Applying both functions to our simulation resulted in a slight over-counting when the neutron density is between 0.1 and 10 neutrons per spot pixel. This error can be as big as 20% but can be corrected a posteriori once the average neutron density is known.

EXPERIMENTAL SETUP

Individual neutron counting with CCD-like cameras has already been reported [2], but image intensifiers were necessary to bring the light flash to a detectable level. What is relatively new, are fast CMOS cameras that allow up to 100 fps, of which the noise properties are sufficiently good, and the quantum efficiency sufficiently high to allow for the threshold application without prior light amplification. All the measurements presented in this paper were carried out using an ORCA-Flash4.0 V2 CMOS camera from Hamamatsu [3]. This camera is equipped with a 4 Mpixel CMOS image sensor adopting an on-chip column amplifier and ADC converter to achieve higher speed, performing simultaneous parallel signal readout. This allows low-noise and high-speed readout. The camera achieves

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

both the low noise of 1.0 electron (median) and 1.6 electrons (rms), and the high frame rate of 100 frames/s (at 2048×2048 pixels). The camera is coupled with a 50 mm f/0.95 MVL50HS objective by Thorlabs. For the neutron detection we used a ZnS(Ag) scintillator screen of a thickness of 200 μm on a 0.5 mm thick aluminium substrate from Applied Scintillation Technologies. In the scintillator, the ZnS powder is mixed with LiF in the proportion 2:1. The ^{6}Li capture a neutron with the reaction:

$$^6Li + n \rightarrow \alpha + ^3H + 4.8MeV \quad (4)$$

where the 4.8 MeV produced are shared between the α (2.05 MeV) and the tritium (2.74 MeV). The ZnS is activated by Ag that creates empty level nearby the conduction and the valence band of the ZnS. α particles or a tritium produce ionisation creating electron-hole pairs. When these couples make a transition from an excited state to the ground state, they emit photons in the visible range. The camera was first used with a 3.7 GBq Am-Be source which provides neutrons with a smooth energy distribution centred around 4 MeV. Those neutrons are subsequently moderated by a thick layer of polyethylene surrounding the source to move the neutron spectrum towards thermal energy. The source emission is nevertheless dominated by a strong gamma flux at 4.4 MeV resulting from the de-excitation of ^{12}C. Final tests have been performed on the T13C [4] beam line. A neutron beam of about 10^9 n·cm^2/s was diffracted by a Ge (111) perfect crystal monochromator. The intensity of the monochromatic beam at 3.26 Åhitting the scintillator was in the order of 10^5 n·cm^2/s.The total beam size was 50×30 mm^2. We have measured the neutron detection efficiency of the scintillator by mean of a reference 100% efficiency ^{3}He detector. The ratio of neutron counts with the aluminium support alone and Al plus ZnS resulted in a 12% efficiency at 3.26Å.

TEST WITH Am-Be SOURCE

For the first test we have exposed the ZnS(Ag) scintillator to an AmBe neutron source (see previous chapter for details). The CMOS camera was looking only partially at the scintillator in order to have clearly a region were no neutron signals should be present. The goal of those first measurements was to find out whether individual neutron detection is possible, how much photo-electrons are collected and whether a threshold-based algorithm can discriminate genuine neutron signals from other noise events. series of images were taken, varying the distance between camera and scintillator from 5 to 10 and 20 cm.

The first processing removes hot pixels which are due to the CMOS technology imperfections and also to the direct gamma impacts from the AmBe source on the camera sensor itself. The second processing recognises individual neutron flashes in order to study them. With the AmBe source, the neutron flux is very low and no measurements in integration mode were possible. Figure 4 report the obtained results. The left side represent the raw image before any

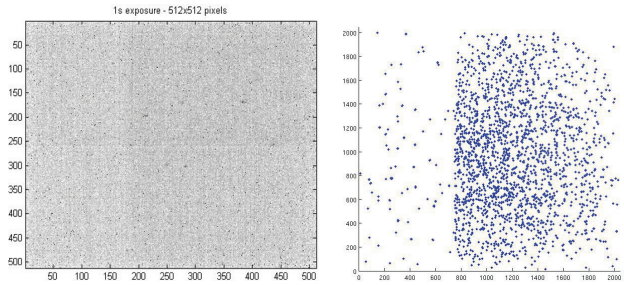


Figure 4: Neutron recognition after partial illumination of the scintillator with the AmBe source.

treatment, while the right side shows the recognised neutrons cumulated over 400 images. One can notice how the majority of neutrons sit on the scintillator side and only a few impacts are located on the side without scintillator. The border of the scintillator is, as well, clearly seen. A second validation comes from the fact that the number of neutrons detected increases proportionally to exposure time. Finally, removing the AmBe source lowers drastically the number of detected neutrons. Given the low intensity of the neutron source, we used exposure times which are longer than we would ideally prefer, in order to have sufficient neutron impacts in a reasonable amount of images.

IN-BEAM TEST

For this series of measurements the CMOS camera was placed inside a black box at 90° with respect to the incoming neutron beam to avoid a direct view of the beam. The 50 mm objective is focused on the scintillator via a 45° mirror obtained by coating a glass plate with titanium.

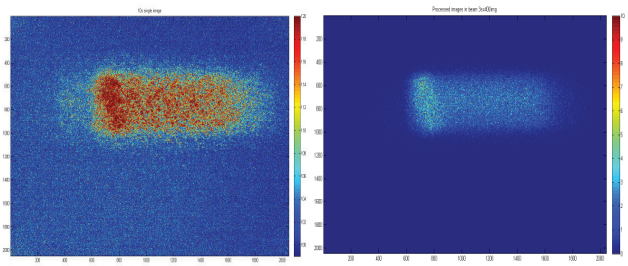


Figure 5: T13C beam profile. Left part is a single image of 10 s exposure, right part is the algorithm reconstruction.

Left part of Fig. 5 shows the beam profile obtained with the CMOS camera in normal integration mode and with an exposure of 10 s. The right part of Fig. 5 represent the result of our algorithm after the processing of 400 images with an exposure time of 50 ms each. A clear asymmetry in the neutron beam is present on both the direct and the reconstructed image. This effect was probably due to a non perfect alignment of the monochromator crystal with respect to the detector. One can notice as well how the reconstruction places all neutrons within the beam area. Also in this case we varied the exposure time to verify the

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

ity of the neutron identification and the results will be discussed in the next chapter.

RESULTS

it is reasonable to expect a linear increase of detected neutrons when increasing the exposure time per image. If this is clearly the case for the measurements performed with the AmBe source, this linearity is not maintained for the beam measurements as one can see in Fig. 6 (blue circles), where the counted neutrons are clearly overestimated.

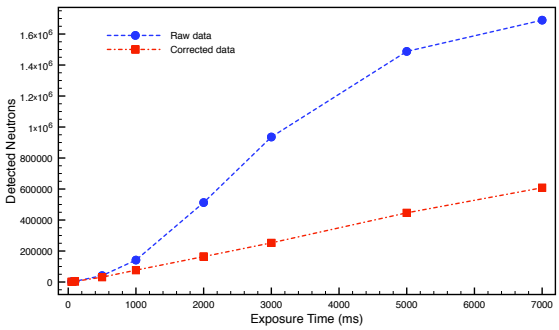


Figure 6: Counted neutrons as a function of the exposure time. Blue circles represent data from original images while red squares are obtained after removal of double counting.

From every single image we calculated the nearest neighbour distance in spot pixel between detected neutrons. This distribution is dominated by a value of 1 pixel which implies the fact that we are probably double counting neutrons. To prove this theory we have simulated a neutron distribution having the same density than the real image. Calculating again the nearest neighbour distance, the possibility of 1 pixel distance is not disappeared but now the peak value is at 4 pixels (see Fig. 7).

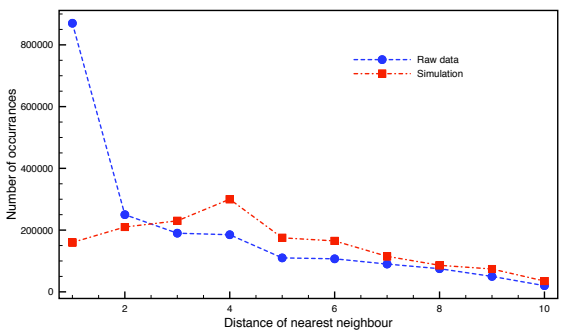


Figure 7: Shortest distance between 2 recognised neutrons. Blue circles represent data from original images while red squares are obtained from the simulated distribution.

After statistical removal of the artificial double counting we obtained the red squares curve reported in Fig. 6 where the linearity is reestablished. A possible explanation is related to the high sensitivity of the algorithm to the threshold parameter. At high neutron density the probability of multiple impacts on nearby pixels increases. The signals are than spread over a number of contiguous spot pixels which are interpreted as double counts. It becomes evident that to be able to exploit the single neutron counting algorithm one has either to consider only low neutron density or increase the camera frame rate to limit, in such a way, the total number of neutrons per frame.

CONCLUSION

Starting from the theoretical concept that a threshold-based neutron counting system can in principle be obtained using a CMOS camera looking at a scintillator, we verified experimentally that single neutron impacts could indeed be identified using an AmBe source as well as in a beam experiment. The recent advances in camera technology now permit us with a commercial camera at relatively low cost, and without the need of light amplification, to detect individual neutrons. This is sufficient justification to invest deeper into the question of a camera-based neutron counting detection system, as in principle, it can combine the good background rejection and the capability to sustain very low count rates, which is typical of neutron counting detectors, with the classical advantages of integrating detectors, such as absence of dead time and hence the capacity to sustain very high local count rates, and high spatial resolution. The beam measurements have shown the problems related to the double counting when the neutron flux increases pointing out the need for a high image rate to reduce the number of neutrons per single frame.

This work would have not been possible without the technical support of Neutron Optics department at the Institut Laue-Langevin. In particular the authors would like to thank Dr. Pierre Courtois for the help in setting up and running the measurements on the T13C beam line.

REFERENCES

[1] Institut Laue-Langevin (2012), France, "T13C high resolution diffractometer", http://www.ill.eu

[2] Hamamatzu (2015), Japan, "ORCA-Flash4.0 V2 Digital CMOS camera", http://www.hamamaztu.com

[3] M. Dietze et al., Nucl. Instr. Meth. A, 377 (1996) 320-324

[4] Kouzes et al., Nucl. Instr. Meth. A, 654 (2011) 412-416

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

NEW DIGITISERS FOR POSITION SENSITIVE ^{3}He PROPORTIONAL COUNTERS

P. Mutti*, E. Ruiz-Martinez, K. Ollivier, M. Platz, P. Van Esch
Institut Laue-Langevin, Grenoble, France

Abstract

The present paper reports on the development of a new digital front-end electronics for position sensitive ^{3}He neutron detectors. The classical analog approach for the charge readout, consisting of a shaping amplifier coupled with a peak sensing ADC, has been replaced by a 64 channels, 62.5 Msample/s and 12 bit digitiser. Excellent results have been obtained in terms of position resolution and signal to noise ratio when adopting a continuous digital filtering and gaussian shaping.

INTRODUCTION

^{3}He gas-filled detectors are a classical choice for the detection of thermal and cold neutrons. The incident neutrons are captured by the ^{3}He producing a tritium and an hydrogen which are sharing the 756 keV of energy generated in the reaction.

$$^3He + n \rightarrow {}^3H + {}^1H + 756keV \quad (1)$$

The electron avalanche initiated by the 2 ions generates the detector signal.

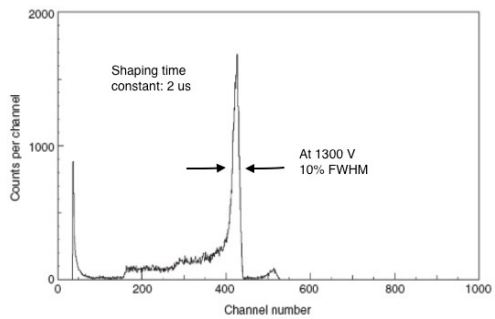


Figure 1: Pulse-height spectrum from ^{3}He tube.

Figure 1 shows a typical pulse-height spectrum from a ^{3}He tube at low gain. At higher gain, non-linear effects set in and the proportionality is lost. This limits the charge one can reasonably obtain from a neutron capture event to about 1 pC. The shape of the spectrum is due to the kinematic of the reaction but also to the choice of the amplifier time constant. The full peak is originated by the collection of the total energy of the two ions (765 keV). If one of the ions is absorbed by the tube walls then, the total collected energy is smaller. This results in the peak asymmetry visible at the left part of the peak in Fig. 1. A choice of a shaping time ranging from 0.5 to 2 μs is a good compromise between good resolution and high count-rate capability.

* mutti@ill.eu

The classical geometry of a charge-division neutron detector consists of a cylindrical volume housing a resistive anode (see Fig. 2).

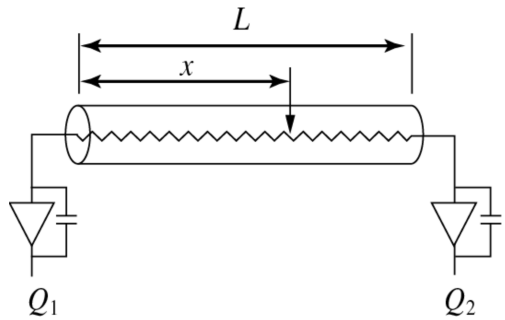


Figure 2: Principle of a charge-division neutron detector.

Electrical signals are extracted at both ends of the tube and the information about the interaction point along the tube can be derived by the ratio of the collected charged at both ends. The theoretical spacial resolution achievable for a detector of total length L is function of the temperature T of the resistive electrode, of the total detector capacity C and of the collected charge. This can be expressed by

$$\frac{FWHM}{L} = \frac{2.54 \cdot \sqrt{kTC}}{Q_1 + Q_2} \quad (2)$$

When using low-noise amplifiers, one can expect from the previous equation (2) a spacial resolution in the order of 0.1% of the tube length. However, Eq. (2) is not applicable in the case of ^{3}He neutron detectors because the charge collection time is much longer than the RC time constant of the detector, as shown in Fig. 3. This results in an actual spacial resolution of the order of 1% of the tube length. The limiting factor being dominated by the noise of the resistive wire and by the integration time.

FRONT-END ELECTRONICS

Our current front-end analog electronics for charge-division consists of a shaping amplifier containing a 4th order Gaussian filter, a baseline correction circuit and a single-shot 12 bit ADC [1], while the position is coded with a 8-bit word due to the limited resources available. The essential idea is to implement digitally an equivalent signal treatment using the CAEN type V1740 [2] digitiser board with an ad-hoc firmware. Since the pulse-shape of the signals depends on the interaction location and on the charges propagation direction (see Fig. 3), the conventional pulse-shape analysis is inappropriate.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

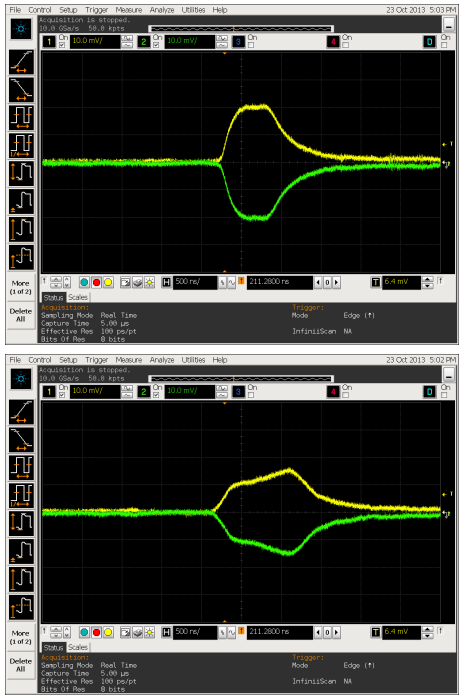


Figure 3: Examples of differential pulse-shapes from ^{3}He tube.

Moreover, in order to avoid errors in the position determination, each couple of digitiser's channels must have a common trigger based on the sum of the detected charges.

DIGITAL SIGNAL PROCESSING

Gaussian Filter

A 4th order approximation of a true Gaussian profile can be expressed by the following normalised transfer function, which has 4 poles in the complex plane:

$$H(s) = \frac{4.899}{4.899 + 11.42 \cdot s + 10.87 \cdot s^2 + 5.073 \cdot s^3 + s^4} \quad (3)$$

If we apply the input invariance method [3] that preserves the discrete samples of the impulse response of the continuous-time domain as the samples of the discrete-time impulse response, we can only transform exponential components of the form:

$$H(s) = \frac{1}{s - s_0} \quad (4)$$

which have continuous-time impulse response

$$h(t) = e^{s_0 t} \quad (5)$$

By sampling the response 5 at a period T_s we can represent the resulting geometric series in the Z-domain as:

$$H_z(z) = \frac{T_s}{1 - e^{s_0 T_s} z^{-1}} \quad (6)$$

If our continuous-time transfer function can be written as a linear combination of this kind of single-pole functions, the technique generalises and the corresponding Z-transform results:

$$H_z(z) = T_s \cdot \Sigma_i \frac{A_i}{1 - e^{s_i T_s} z^{-1}} \quad (7)$$

Therefore, if we can write the original continuous-time domain transfer function as a partial-fraction expansion with simple terms (which means that we don't have double poles), we can obtain the partial-fraction expansion of the corresponding Z-transform with the same coefficients (A_i in Eq. (8), and with the poles following a $s_i \rightarrow exp(s_i T_s)$ mapping. However, in principle the recombination of all these terms into a single rational function implies the possible appearance of finite zeros. We neglect these zeros assuming they remain at the origin to avoid the increase of the number of multiplications in the filter implementation since we are limited in computing resources in the FPGA. This approximation is reasonable since the zeros are less important the the poles. They apply a FIR filter to the input of a length equal to the order of the filter (4 in our case). This will only have an impact if all the coefficients in the numerator of Eq. (6) were 0. In reality those are all positive, in which case the FIR filter will only calculate a kind of weighted average over 5 successive samples. In addition, it turns out that only 3 coefficients have a significant value and therefore, this term can be neglected as far as the time constant of the filter is much more that 3 samples. The denominator of Eq. (8), corresponding to the IIR filter, has the dominant effect on the impulse response. With the previous considerations, we can reformulate Eq. (8) as:

$$H_z(z) = \Pi_i \frac{constant}{1 - e^{s_i T_s / 2\pi\tau} z^{-1}} \quad (8)$$

where *i* runs over the 4 s-plane poles mentioned earlier. This will result in 2 second-order factors for the denominator

$$(1 - a_1 z^{-1} + b_1 z^{-2})(1 - a_2 z^{-1} + b_2 z^{-2}) \quad (9)$$

which can easily be implemented as a succession of two second-order IIR filters. The fixed-point implementation will consist in multiplying the constants a1 and b1 of Eq. (9) with a power of 2, applying the integer calculation and dividing the result by the same power of 2. We opted for the 14th power of 2. It can also be interesting to multiply input and output with a small power of 2 to get some *digits after the comma*. These can be removed at the end of the calculation, to take care of part of the rounding errors.

Pole-zero Compensation

The continuous-time domain pole-zero compensation comes down in accepting a signal that went through a first order system with a long time constant, and modifying this such that it looks as if the signal went trough a first order system with a much shorter time constant. This can simply be implemented with the following transfer function:

$$H(s) = \frac{s + 1/\tau_l}{s + 1/\tau_s} \quad (10)$$

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

where τ_s is the short time constant and τ_l is the long one for which we want to compensate. In this case, τ_s is our own choice while τ_l must match precisely the time constant of the system. Transforming Eq. (10) in the discrete domain we can write

$$Z(z) = 1 + (T_s/\tau_l - T_s/\tau_s) \cdot \frac{1}{1 - e^{-T_s/\tau_s} z^{-1}} \quad (11)$$

If τ_s is much bigger that the sample period one can approximate $e^{-T_s/\tau_s} \approx 1 - T_s/\tau_s$. This can, then, be implemented as a linear combination of the direct input and a first order IIR filter:

$$w[n] = i[n] + (1 - \frac{T_s}{\tau_s}) \cdot w[n-1] \quad (12)$$

$$u[n] = i[n] + (T_s/\tau_l - T_s/\tau_s) \cdot w[n] \quad (13)$$

where we can calculate in fixed point integer and shift the result with a certain power of 2 to allow integer multiplication, followed by a division and truncation by the same power of 2.

Baseline Correction

This correction neutralises offset in the signal input and it adjusts the signal level in between pulses to zero. This means that the average of the signal is not zero (given the pulses are positive), and hence that the AC coupling is undone. The base line correction is a non-linear operation. Our implementation consists of chopping the samples up in chunks of length N. In every of these chunks we determine the minimal sample value and we fed this value to a first order low-pass digital filter:

$$w[n] = \frac{1}{k} \cdot i[n] + (1 - \frac{1}{k}) \cdot w[n-1] \quad (14)$$

If k is a power of 2 then, this can easily be implemented with just shifting binary words. In Eq. (14) $i[n]$ is the minimum of the last chunk and n counts the number of chunks, while $w[n]$ is the estimated baseline of the input signal. This method implies of course a small error since the minimal sample will be the minimum of the noise excursion and not the average baseline value. Therefore, on a pure noise signal, the average level after correction will not be 0, but it will be the negative of the average negative excursion (due to noise) on N samples. This amounts to a few sigma of the noise level if the noise has a Gaussian distribution.

Implementation

The signal treatment per channel consists of 5 blocs that can be independently activated or deactivated:

- first baseline correction applied to the incoming signal
- pole-zero compensation
- first second-order Gaussian filter
- second second-order Gaussian filter
- final baseline correction

The input is a 12-bit unsigned data and the output is a 16-bit unsigned data while between blocs the signals are of signed type to be able to treat positive as well as negative samples. All blocs use fixed-point arithmetic with a certain binary fraction to avoid too many significant rounding errors while saving resources.

DATA TREATMENT

Trigger

To avoid having a different trigger efficiency along the tube, the channels of the digitiser have been paired. Indeed, a neutron interacting close to one edge of the tube will generate a large signal on that end and almost no signal on the other end. To avoid missing triggers when the signal amplitude is very low, each channel independently makes the sum of his proper output data stream, and the data stream of its paired channel. A local trigger is fired if that sum crosses the set threshold for the channel. As such, if the paired channel (who is treating the same sum of course) has the same threshold setting, both independent triggers will fire at exactly the same clock pulse even though the trigger signals themselves are not coupled.

Maximum

When the trigger is fired a maximum finder on the sum signal will find that sample which is the largest since the trigger was fired. The corresponding local channel sample value is stored until a larger value on the sum is found or a new trigger is issued. As all this happens in real time, the value should be read out after the maximum has been reached on the output data stream, and before a next trigger is fired. The local output sample value corresponding to the time tick when the sum of the two channel outputs reached its global maximum since the threshold crossing, is what is sent to the output stream on an unsigned, 16-bit word.

RESULTS

To test the quality of the digital signal treatment and the achievable resolution we have compared the results obtained with our standard analog system versus those from the digitiser. To decouple possible problem in the firmware implementation from those of the algorithm we have first treated offline the samples from the digitiser with a c++ code implementing the digital filter. Signals at 1 kHz rate from an Agilent waveform generator have been injected into a PAD02 [1] shaping amplifier implementing in an analog way the Gaussian filter with a time constant of about 1.4 μs. A resistance of 3 kΩ was used to simulate the noise from the resistive wire of the ^{3}He gas detector. The same PAD02 amplifier but without the Gaussian shaping part has been used to amplify the signals before sending them to the digitiser. The shaping time of the digital filter has been set to 1.6 μs.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Table 1: Position Resolution Obtained With Analog and Digital Front-End Electronics

	Analog	Charge Integ.	C++ code	Firmware
Injected charge (pC)	0.2 0.6 1.0	0.2 0.6 1.0	0.2 0.6 1.0	0.2 0.6 1.0
Dynamic	256	4096	4096	65536
FWHM (measured)	4 2 1	65 20 15	124 43 25	553 187 113
FWHM (theory)	3.9 1.3 0.8	63.5 21.1 12.7	119.8 39.9 23.9	507.9 169.3 101.6
Resolution (%)	1.56 0.78 0.39	3.03 1.05 0.61	1.59 0.49 0.37	1.69 0.57 0.34

Table 1 summarises the obtained results. The measured FWHM of the position distribution has been compared with the theoretical one. This last include only the noise from the amplifier and the resistance. Therefore, a FWHM value close to the theory implies that the digitalisation and the Gaussian filtering do not contribute significantly to the noise balance. One can notice that, as expected, no substantial difference exists between the off-line (C++ code) and the FPGA implementation of the algorithm. The same measurements have been repeated using the standard firmware provided with the CAEN digitiser. In this case (column 3 in Table 1), a simple charge integration is performed. The obtained resolution is about a factor of 2 worst than for the Gaussian filter. A second set of measurements was performed using a 1 kΩ to simulate a different resistive wire and the obtained results are consistent with those reported in Table 1. For the final firmware validation we have acquired data using a large ^{3}He detector counting 128 tubes each of 1 m hight and 8 mm diameter and containing 12 bar of ^{3}He. This detector, installed at the D22 [4] beamline of Institut Laue-Langeving, provides a detection efficiency of about 70% at a wavelength of 6 A. Due to the lack of channels available in the prototype digitiser we could only use 8 out of the 128 tubes.

Figure 4 depicts the obtained results when exposing the detector to a AmBe neutron source. A Boral grid with horizontal linear gaps of 2 mm hight was placed right in front of the detector. Left image has been obtained with our standard

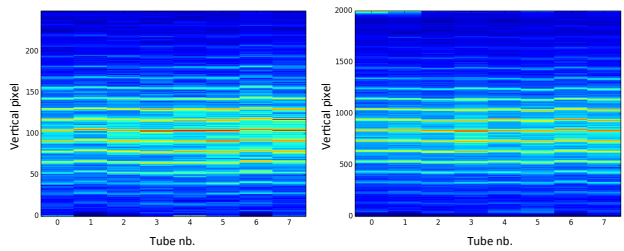


Figure 4: Comparison of D22 detector images acquired with our standard front-end electronics (left) and with the new digital system (right).

front-end electronics while the right side is the result from the digitiser running our new charge-division firmware. One can notice the excellent spacial resolution and, as expected from Table 1, no significant differences with respect to the analog approach.

CONCLUSIONS

Using as base hardware a commercial digitiser board from CAEN [2] we have implemented in the existing FPGA of the board a new charge-division firmware especially designed for position sensitive ^{3}He proportional counters. A 4th order digital Gaussian filter for the pulse shaping, as well as pole-zero and baseline correction have been included in the firmware. Obtained results both on laboratory test and on real detector show excellent performances of the digital electronics that will be soon ready for commissioning on various instruments at the Insitut Laue-Langevin.

This work would have not been possible without the help and the technical support of CAEN. In particular the authors would like to thank Luca Colombini, Carlo Tintori and Gianni Di Maio for the engagement in the project and the fruitful discussions.

REFERENCES

[1] P. Van Esch *et al.*, "Design criteria for electronics for resistive charge division in thermal neutron detection", Nucl. Instr. Meth. A, Vol. 526, pp. 493-500, 2004

[2] CAEN (2013), Italy, Technical information manual mod. V1740 64/32 channel 12-bit 62.5 MS/s digitiser, `http://www.caen.it`

[3] L. Jackson, "A correction to impulse invariance", IEEE Signal Processing Letters, Vol. 7, Oct. 2000

[4] Institut Laue-Langevin (2012), France, "D22, a small-angle neutron scattering diffractometer", `http://www.ill.eu`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE CONSTRUCTION OF THE SUPERKEKB MAGNET CONTROL SYSTEM

T. T. Nakamura#, A. Akiyama, M. Iwasaki, H. Kaji, J.-I. Odagiri, S. Sasaki, KEK, Ibaraki, Japan
N. Yoshifuji, EJIT, Hitachi, Ibaraki, Japan
T. Aoyama, T. Nakamura, K. Yoshii, Mitsubishi Electric System & Service Co., Ltd, Tsukuba, Japan

Abstract

There were more than 2500 magnet power supplies for KEKB storage rings and injection beam transport lines. For the remote control of such a large number of power supplies, we have developed the Power Supply Interface Controller Module (PSICM), which is plugged into each power supply. It has a microprocessor, ARCNET interface, trigger signal input interface, and parallel interface to the power supply. The PSICM is not only an interface card but also controls synchronous operation of the multiple power supplies with an arbitrary tracking curve. For SuperKEKB we have developed the upgraded version of the PSICM. It has the fully backward compatible interface to the power supply. The enhanced features includes high speed ARCNET communication and redundant trigger signals. Towards the Phase 1 commissioning of SuperKEKB, the construction of the magnet control system is ongoing. First mass production of 1000 PSICMs has been completed and their installation for Phase 1 configuration has been done. The construction status of the magnet control system is presented in this report.

INTRODUCTION

KEKB is the asymmetric energy electron-positron collider, which is dedicated to the B-meson physics. Its operation started in December 1998 and finished in June 2010. The KEKB accelerator control system has been constructed based on EPICS (Experimental Physics and Industrial Control System) tool kit. EPICS provides core mechanism for the distributed control system. EPICS runtime database is running on a local control computer called IOC (Input/Output Controller). More than 100 VME/VxWorks computers have been installed as IOC in the KEKB accelerator control system.

SuperKEKB is the upgrade of KEKB. It aims at 40-times higher luminosity than the world record by KEKB. Its operation is going to be performed in three phases. The Phase 1 is the operation without QCS (final focusing superconducting magnets) nor BelleII detector. Start of the Phase 1 is scheduled in Feb. 2016. Phase 2 follows after installation of QCS and BelleII,.

In the KEKB storage rings and the injection beam transport lines, about 2500 magnet power supplies were installed [1] and controlled by 11 IOCs. To connect such a large number of power supplies to the IOCs, we adopted ARCNET as the field bus and developed the PSICM (Power Supply Interface Controller Module) [2]. The PSICM is the ARCNET interface board for the power supply. The PSICM has the shape of 3U Euro-card format (100mm × 160mm) with a DIN 64-pin connector and can be plugged into the power supply. Figure 1 shows the photo picture of the PSICM and the magnet power supply with PSICM. Figure 2 shows the details of the interface between the PSICM and the power supply.

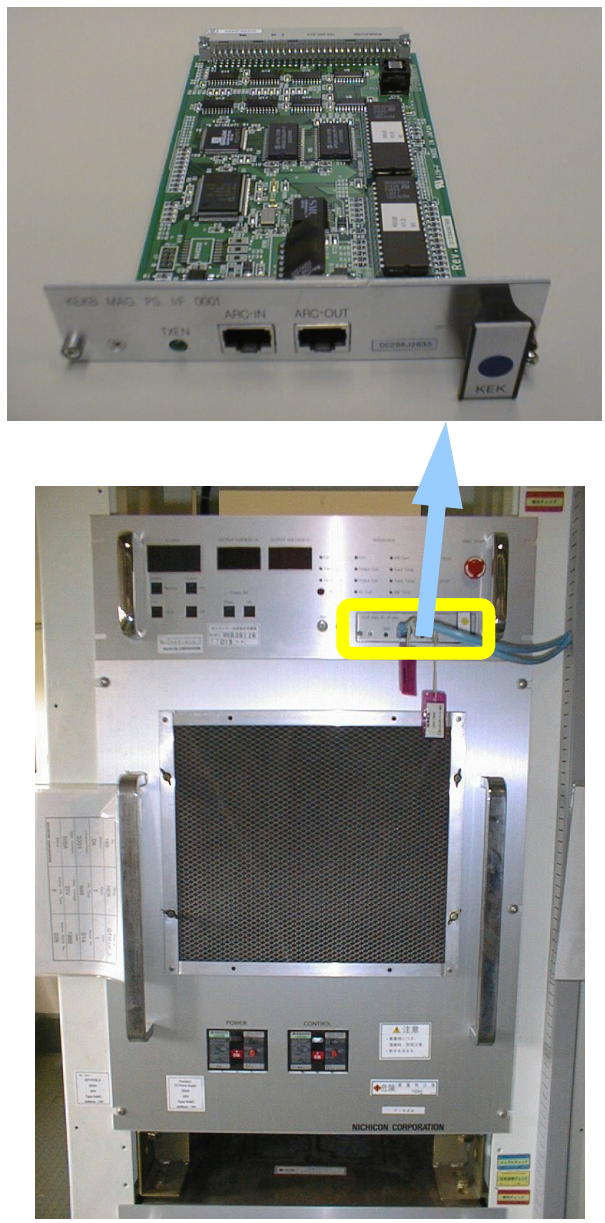


Figure 1: Whole shape of the PSICM (above) and the PSICM plugged in a magnet power supply (below).

tatsuro.nakamura@kek.jp

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The ARCNET allows using several kinds of media. We adopted shielded twisted-pair (STP) cable as the media and HYC2485 as the media driver. This configuration allows up to 20 ARCNET nodes to be connected on single segment in the daisy chain manner. The STP cable includes an auxiliary twisted-pair for the external trigger signal together with the ARCNET. Figure 3 shows typical configuration of ARCNET used in the KEKB magnet power supply control system.

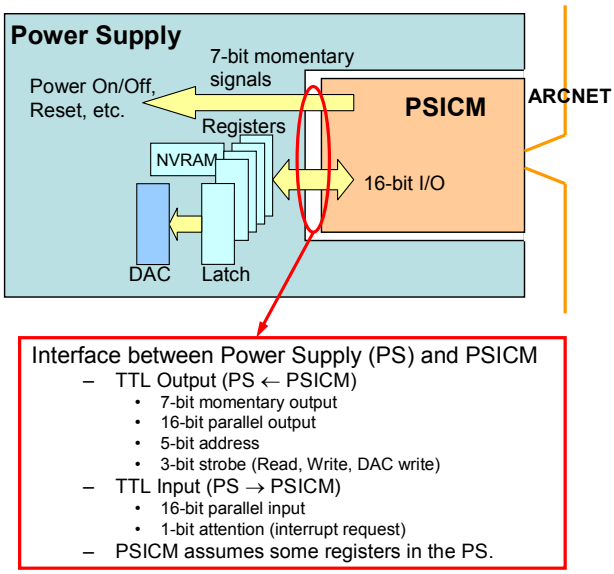


Figure 2: The interface to the magnet power supply.

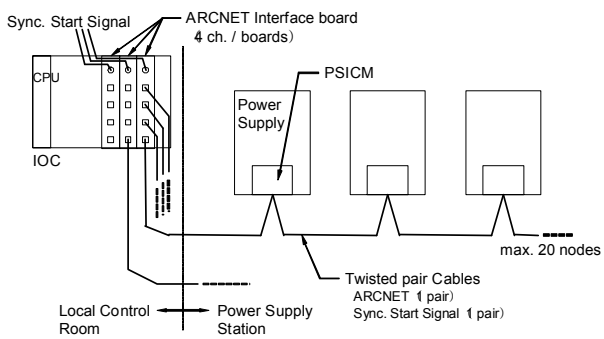


Figure 3: Typical configuration of ARCNET in KEKB.

The PSICM is designed to control the output current of the power supply according to the arbitrary tracking curve. The tracking data are sent from the IOC to the PSICM as an array of the output current values. After receiving the data the PSICM start tracking by a start command or an external trigger signal. Figure 4 shows the schematic diagram of this sequence. Using the external trigger signal all magnets in the storage ring can be synchronously operated. The sequence control of the synchronous operation is performed by the IOCs with the arbitration by the server process on the central workstation [3].

The magnet power supplies of the Photon Factory Advanced Ring (PF-AR) in KEK are also controlled in the similar manner using the PSICM.

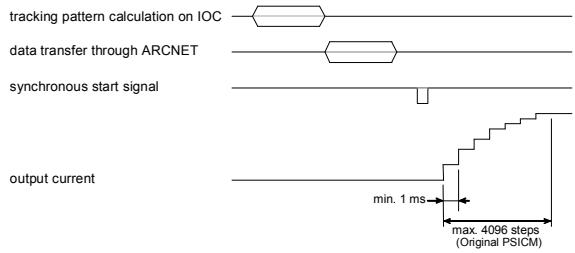


Figure 4: Schematic diagram of synchronous operation.

NEW VERSION OF THE PSICM

For the SuperKEKB, significant numbers of power supplies are newly manufactured. The PSICMs have worked in KEKB for 12 years. During these years some of the parts have been discontinued and it becomes much harder to reproduce the PSICM. Thus we have developed the new version of the PSICM [4]. Table 1 shows the hardware specifications of the new PSICM together with the original one. Figure 5 shows the photo picture of the new PSICM.

Table 1: Hardware Specifications

	Original PSICM	New PSICM
Microprocessor	AM186	MPC8306
Clock frequency	20MHz	133MHz
Data memory	256kB SRAM	128MB DDR2 SDRAM
Program memory	256kB EPROM	64MBit NOR FLASH
ARCNET interface	2.5Mbps Backplane mode	2.5/5/10Mbps Backplane mode
Controller	COM2002	COM20022
Media driver	HYC2485	HYC5000
Power required	5V 0.4A	5V 1A

Figure 5: The upgraded PSICM.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Compatibility is the most important issue for the design of the new PSICM. The specifications of the interface to the power supply are fully backward compatible to the original PSICM. The new PSICM can be plugged into any existing power supplies.

The application-level communication protocol between the IOC and the new PSICM is also backward compatible to the original PSICM. All of the command messages defined for the original PSICM can be accepted also by the new PSICM.

In addition following extensions are introduced.

(1) The high speed ARCNET communication is supported. 10Mbps, 5Mbps and 2.5Mbps are supported for the new PSICM while the original PSICM supports 2.5Mbps only.

(2) 32-bit data handling of the tracking data array is available in order to support high resolution DAC (24-bit, 20-bit and 18-bit). The original PSICM supports 16-bit DAC only. (Later special version for 18-bit DAC was developed based on the original PSICM. But this version was ad hoc and limited.)

(3) Dual trigger inputs for synchronous start signals are available. This feature enables trigger signals redundant. More reliable synchronous operation is expected.

(4) More reliable RJ-45 connectors which have optional protectors against dust are adopted.

INSTALLATION OF THE NEW PSICM AND THE CONSTRUCTION STATUS

First mass production of 1000 modules have been completed in March 2014. We finally need 3000 modules for all magnet power supplies of the main storage rings, the damping ring and the beam transport lines. The schedule of the second mass production of 2000 modules has not yet fixed due to the budget limitation. Thus we have decided to use both old and new PSICM at least at the beginning of the commissioning of SuperKEKB. For the Phase 1 operation of the main storage rings, 426 New PSICM (out of 2162 magnet power supplies) have been installed. The remote operation test of the power supplies with PSICM and IOC is now in progress. Figure 6 shows the example of the combination of the old and new PSICM.

We also upgrade the VME-based IOC for the magnet power supply control. The CPU board has been upgraded from PowerCore6750 to MVME5500. VxWorks has been updated from Ver. 5.3.1 to Ver. 6.8.3. EPICS on the IOC has been updated from R3.13.1 to R3.14.12.3. VME board of the 4-channel ARCNET interface have been also upgraded. Figure 7 shows the renewal of the IOC.

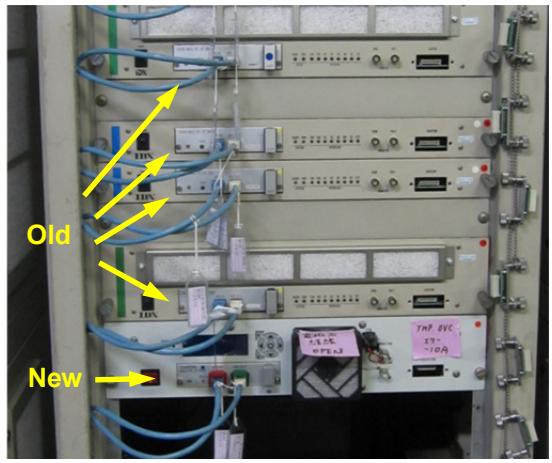


Figure 6: The combination of the old and new PSICM.

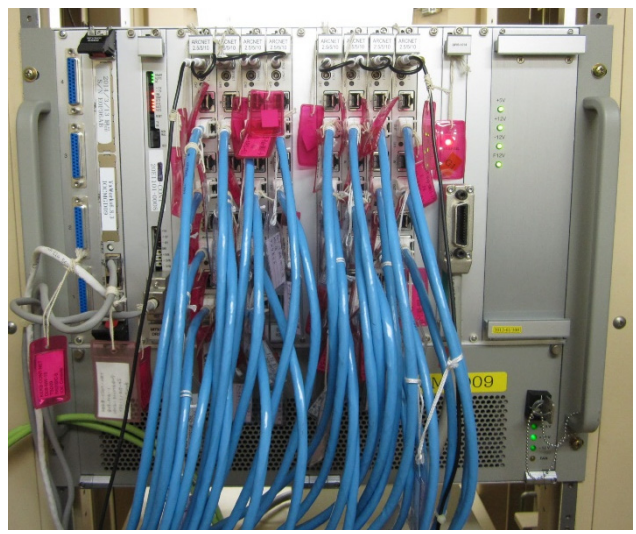


Figure 7: The setup of the IOC for SuperKEKB.

INTERFACE TO THE MAGNET INTERLOCK SYSTEM

We have completely changed the interface to the magnet interlock system. In KEKB, Modbus-plus was used as the interface between the interlock system and the IOC located in the D8 local control room. In SuperKEKB, the interlock system has been completely replaced with new one, which use Yokogawa FA-M3 PLC. In the PLC, We have installed F3RP61 CPU module, on which EPICS running. Thus the EPICS IOC is embedded in the interlock system. Figure 8 shows the PLC in the D8 power station.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

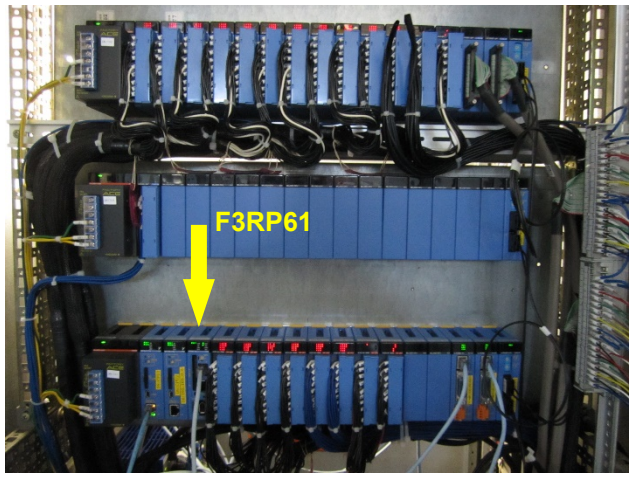


Figure 8: The PLC for the magnet interlock system. The photograph is taken at the D8 power station. F3RP61 CPU module is mounted (yellow arrow).

CONCLUSION

We have developed the new PSICM for SuperKEKB. It is backward compatible to the original PSICM. Several enhancements are introduced. The high speed ARCNET communication is expected to enable more efficient operation of the magnet power supplies. Wide variety of the high resolution DAC can be supported by 32-bit data handling. Dual trigger input for the synchronous start signals, together with the dust protectors of the RJ-45 connectors, is expected to enable more reliable and stable operation of the magnet power supplies. We have completed the first mass production of 1000 modules of the new PSICM and installed for the Phase 1 operation. We are going to start with the combination of the old and new PSICM because of the budget limitation. The compatible design of the new PSICM enables such a flexible configuration.

REFERENCES

[1] M. Yoshida et al., "Magnet Power Supply System for KEKB Accelerator", EPAC-98, Stockholm, 22-26 June 1998.

[2] A. Akiyama et al., "KEKB Power Supply Interface Controller Module," ICALEPCS'97, Beijing, 3-7 Nov. 1997.

[3] T.T. Nakamura et al., "Magnet Power Supply Control System in KEKB Accelerators," TC1P40, ICALEPCS'99, Trieste, 4-8 Oct. 1999; www.JACoW.org

[4] T.T. Nakamura et al., "Upgrade of the Power Supply Interface Controller Module for SuperKEKB," TUPPC089, ICALEPCS2013, San Francisco, 6-11 Oct. 2013; www.JACoW.org

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CERN OPEN HARDWARE EXPERIENCE: UPGRADING THE DIAMOND FAST ARCHIVER

I.S. Uzun and M. Abbott, Diamond Light Source, Oxfordshire, UK

Abstract

Diamond Light Source developed and integrated the Fast Archiver [1] into its Fast Orbit Feedback (FOFB) communication network in 2009. It enabled synchronous capture and archive of the entire position data in real-time from all Electron Beam Position Monitors (BPMs) and X-Ray BPMs. The FA Archiver solution has also been adopted by SOLEIL and ESRF. However, the obsolescence of the existing PCI Express (PCIe) based FPGA board from Xilinx and continuing interest from community forced us to look for a new hardware platform while keeping the back compatibility with the existing Linux kernel driver and application software. This paper reports our experience with using the PCIe SPEC card from CERN Open Hardware initiative as the new FA Archiver platform. Implementation of the SPEC-based FA Archiver has been successfully completed and recently deployed at ALBA in Spain.

INTRODUCTION

The Diamond FOFB system is operating successfully and achieving the required level of suppression of beam disturbances since 2007 [2]. The system architecture for the FOFB for a single Storage Ring (SR) cell is shown in Fig. 1. Diamond SR consists of 24 cells, and overall FOFB architecture consists of:

- 172 Electronic Beam Position Monitors (BPMs) located around the storage ring.
- 24 VME-based processors, each populated with a PMC receiver, running the feedback algorithm.
- Power supply controller units for 168 slow corrector magnets.
- Communication network to distribute the beam position values.

BPMs are used for electron beam diagnostics. In the BPM FPGA the sampled button intensities are converted to X,Y positions and reduced by decimation and filtering to machine revolution frequency (534 kHz) and then to the "Fast Acquisition" (FA) data rate of 10 kHz and the "Slow Acquisition" rate of 10 Hz for EPICS clients. The FA data is broadcast by Diamond Communication Controller IP (DCC) [3] on the FPGA through a dedicated communication network, which makes this data stream simultaneously available to all PMC receiver nodes on VME processors for fast feedback processing.

Following the successful operation of FOFB system, a need for recording longer periods of synchronous position data from all nodes on the communication network (the whole orbit) at FA data rate was identified. The DCC IP and dedicated communication network enabled us to additional nodes with little or no impact on the operation of the FOFB; and the FA Archiver is one such node implemented on a PCIe hardware [1].

Two FA Archiver nodes have been running at Diamond, one for Booster and one for Storage Ring since 2009, giving us access to records of up to two weeks of position data across the both rings. We have been using this data for:

- Identification and detailed investigation of any orbit events that happened in the recent past.
- Correlation of orbit events with other measurements by adding X-ray BPMs and RF cavity voltage readings to the FA network.
- Model analysis of orbit motion over longer periods for the optimisation of the fast orbit feedback.

FA Archiver interface was implemented on a commercial Xilinx ML555 PCIe FPGA [4] board as a cost-effective solution so that we could focus our efforts on firmware development and software integration. Following its successful commissioning at Diamond, SOLEIL and ESRF integrated FA Archiver as part of their fast feedback system. This was possible since both facilities built their FOFB data distribution network using DCC giving them the protocol compatibility required.

Recent interest from ALBA as the latest user of DCC, and Brazilian Light Source (LNLS) for evaluation purposes brought the obsolescence of the ML555 platform to our attention. In order to support the community, we took on an upgrade project targeting implementation of FA Archiver on a modern platform with longer term support and availability. Following evaluation of commercially available products and in-house built possibilities, we have decided that SPEC card from CERN Open Hardware initiative would be the best

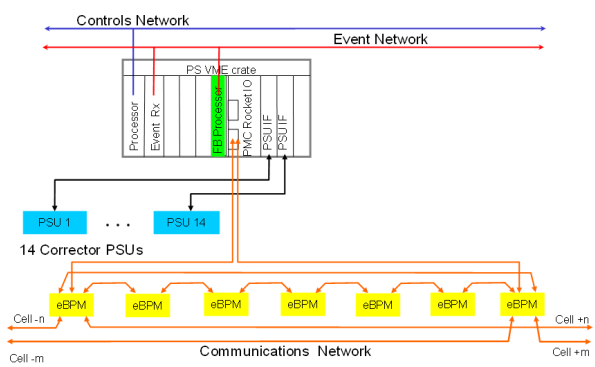


Figure 1: FOFB architecture for single SR Cell. It is repeated for each 24 cells in Diamond SR.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

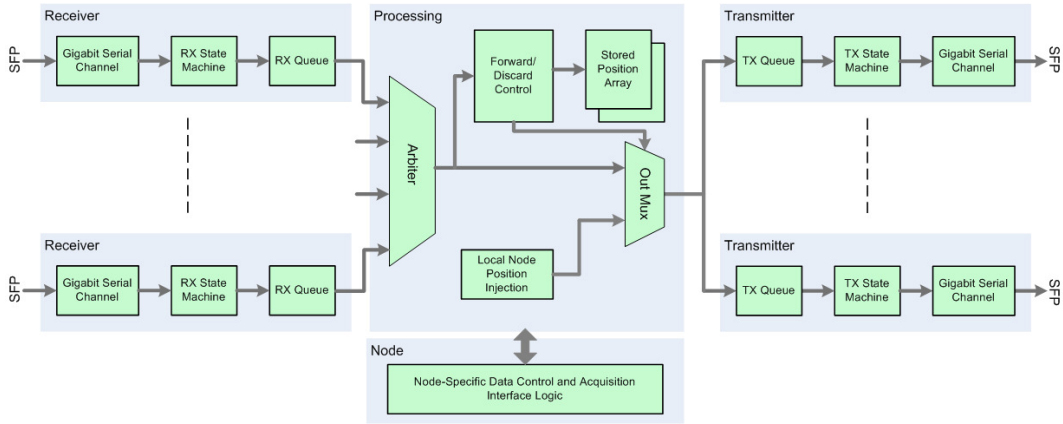


Figure 2: Major building blocks of DCC design.

candidate due to its wide use in the accelerator community for various projects and openness of the design sources.

DIAMOND COMMUNICATION CONTROLLER

To realise the required data transfer (BPMs to computation nodes) at a 10 kHz update rate, DCC IP was designed and implemented in VHDL. The IP uses Multi-gigabit Serial Transceivers (MGTs) on the FPGA achieving communication rate of 2.12 Gbps for Diamond.

DCC utilises a data-forwarding protocol whereby each node forwards its own data, plus each piece of incoming data once. The data is encapsulated in a low-overhead packet including CRC. The architectural block diagram of the DCC is shown in Fig. 2. The IP design consists of three main blocks:

- Receiver block implements up to 4 input channels, and is responsible for de-serialising, decoding, CRC checking and queuing the incoming packets.
- Data Processing (Store, Analyse and Forward) keeps tracks of incoming packets from each node on the communication network, and transmits a specific node's packet only once in a given timeframe. This approach prevents flooding the communication network. It also stores received values for feedback or archiving purposes.
- Transmitter block implements output queues for each channel. It is responsible for adding CRC, encapsulating and encoding position data for transmission through the serial links.

A dedicated fibre network between BPMs was chosen as a cost effective method of achieving low-latency communication. The BPMs have four SFP multi-gigabit serial transceivers, the 24 storage ring cells are wired in a torus network with single-mode fibre while copper patch cables are used intra-cell. The DCC network topology is structured as a 2D torus, with one computation node per SR cell, and gives a degree of resilience to failure of single or combinations of BPMs or links. The communication network structure is shown in Fig. 3.

DIAMOND FA ARCHIVER

The design of the DCC and communication network makes it easy to add new nodes to the network, both as contributors and as passive listeners. The FA archiver acts as a passive listener storing the data stream in a 30TB ring buffer. The stored data is then available to clients via a set of software tools.

Implementation of FA archiver interface requires a PCIe form factor hardware platform with following on-board resources:

- Minimum 1-lane of PCIe bus interface. This could be either through dedicated IPs blocks on an FPGA device or a bridge chip external to the FPGA.
- At least 1 MGT on the FPGA for DCC implementation.
- On-board SFP connector(s) for physical connection to the communication network.
- Clocking resources required for PCIe, DMA data transfer and DCC FPGA logic.

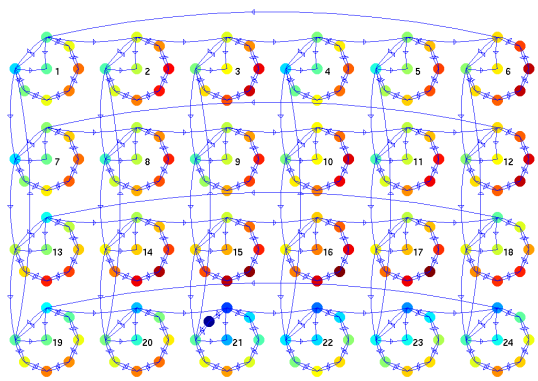


Figure 3: Communication network topology, with each circle being one cell consisting of seven BPMs, and a PMC receiver in the centre. The coloring represents total distribution time for each node.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

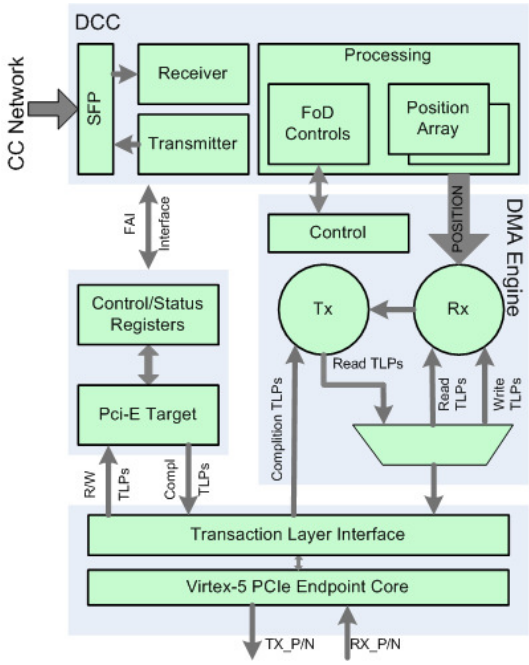


Figure 4: Architecture of the FA Archiver FPGA design on ML555 board. The design directly interfaces to PCIe physical connector.

The FPGA design for the FA Archiver is shown in Fig. 4. It integrates the DCC into a bus master PCIe architecture on the ML555 FPGA board. The design consists of target logic, DMA initiator logic, status/control registers and the Virtex-5 endpoint core for PCIe. Target logic is responsible for capturing single doubleword memory write and memory read PCIe Transaction Layer Packets (TLPs) for control and status register access. The DMA initiator logic generates memory write TLPs to transfer 4K byte frame data from the DCC core to the host's system memory through PCIe endpoint.

CERN OPEN SOURCE HARDWARE

Open source hardware (OHW) is hardware whose design is made publicly available so that anyone can study, modify, distribute, make, and sell the design or hardware based on that design [5]. Open hardware has in recent years established itself as a very effective way for CERN to make electronics designs and in particular printed circuit board layouts, accessible to anyone, while also facilitating collaboration and design re-use. It is creating an impact on many levels, from companies producing and selling products based on hardware designed at CERN, to new projects being released under the CERN Open Hardware Licence (OHL) [6]. Today the open hardware community includes large research institutes, universities, individual enthusiasts and companies. Many of the companies are actively involved in the entire process from design to production, delivering services and consultancy and even making their own products available under open licences.

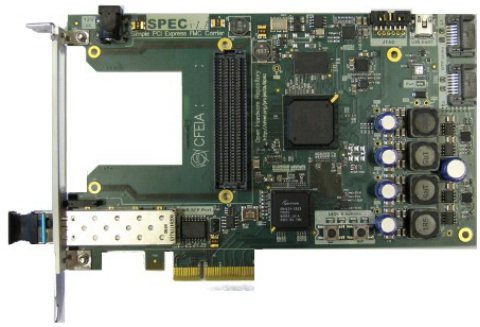


Figure 5: Commercially available SPEC card.

Open Hardware Repository

Recognizing the benefits that an open source philosophy could provide to hardware development and dissemination, CERN created the Open Hardware Repository(OHR) in 2009. It is a place on the web for electronics designers at experimental physics facilities to collaborate on open hardware designs. Technical specifications, research documents, CAD drawings and HDL code relating to projects are available for download from the OHR version control system. The Open Hardware Repository currently hosts more than 100 projects, ranging from small projects with a few collaborators to bigger projects with multiple contributors from both industry and academia. A dozen companies are actively involved in projects in the OHR and some of them are producing the physical hardware for CERN and other customers.

One of the prominent projects within OHW framework is the Simple PCIe FMC carrier (SPEC) card developed by CERN, as shown in Fig. 5. SPEC is a data-acquisition card that can be plugged in to a PC, allowing the use of different FMCs, such as a digital input/output module, a time-to-digital converter and a fine delay module. The SPEC card was designed with the White Rabbit (WR) timing project in-mind and is the most widely-sold card offering WR support for synchronization. Several companies sell the card and are involved in its development. SPEC cards have found their way to many scientific applications, such as fusion energy, tracking of space debris and control systems for accelerator complexes.

The SPEC card hardware specification, availability of design sources on OHR along with its low-cost commercial availability made it an attractive solution as the new hardware platform for upgrading Diamond FA Archiver. It was also our aim to interact and build experience with OHW initiative as a user.

FA ARCHIVER ON SPEC CARD

The initial challenges with the OHW repository at the start of our project were:

- Understanding our way around the OHR since hardware, firmware and software projects for the SPEC card are maintained in separate repositories.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- Finding details of the components and features of the SPEC card due to lack of a Hardware User Guide.

By studying the open-source board schematics and reference firmware design available on the repository, we were able to identify the modifications to be taken towards implementing the FA Archiver on the SPEC card.

PCI Express Interface

The Spartan-6 FPGA on the SPEC card doesn't have an embedded PCIe IP block unlike the Virtex-5 FPGA on the ML555 board. As a result, an external PCIe bridge, Gennum GN4124, is used to handle the host communication on the SPEC card. This hardware difference required a completed re-implementation of FA Archiver's existing PCIe and interrupt interface logic so that the communication is achieved by using the external bridge's local interface rather than the embedded Xilinx IP block.

Clocking Resources

The SPEC card was design specifically targeting the White Rabbit project, therefore the clocking resources around the FPGA was built on WR requirements which wouldn't suit the DCC. To achieve the required clocking frequency of 106.25MHz for DCC, an external on-board programmable oscillator is used. This was achieved by modifying the DCC firmware to route the clock using on-chip global buffers rather than dedicated MGT clocking pins. This scheme introduces unwanted jitter on the clock, but since MGTs on the FPGA are qualified up to 6.8 Gbps, its effect at the DDC line rate of 2.12 Gbps does not create any unwanted bit errors rates. This modification on the clock routing created additional conflicts for the main FPGA system clock input. This was solved by eliminating the use of the dedicated system clock inputs, and using the PCIe bridge's local clock as the FPGA system clock.

On-Chip Communication

The reference firmware design for the SPEC card uses the Wishbone System-on-Chip (SoC) bus for interconnecting IP cores. The FA Archiver design uses ARM's Advanced Microcontroller Bus Architecture (AMBA) which is the adopted solution for interconnecting Xilinx IPs. Towards achieving the maximum re-use of reference design IPs, FA Archiver's Target and DMA Control Logic design blocks handling the status and control registers for operating the board were also modified to adopt the Wishbone bus.

The resulting FPGA design architecture for the FA Archiver on the SPEC card is shown in Fig. 6.

Software

The only software modification was in the device driver to detect and identify the card installed (ML555 or SPEC), since the SPEC card's PCIe chip has to be initialised before bringing the interface up. The archiver processing software and client tools as explained in [1] didn't require any modifications keeping the back-compatibility.

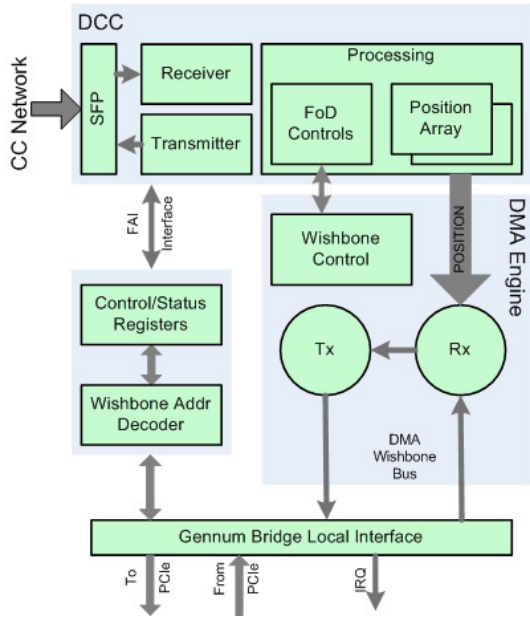


Figure 6: Architecture of the FA sniffer FPGA on SPEC card. The design interfaces to an external PCIe bridge using the parallel local interface.

SUMMARY

The motivation for the new FA Archiver interface has been to address obsolescence of the existing PCIe platform, and move to an open platform for continuing support in the community. We took this opportunity to get familiar with Open Hardware initiative, and build experience with the inner workings of the repository. The upgrade project has been successfully completed with initial challenges due to learning stages involved. The new FA Archiver has been running at ALBA for the last year, and will be evaluated by LNLS with DCC running at an increased line rate of 5 Gbps.

REFERENCES

[1] M.G. Abbott, G. Rehm, I. Uzun, *A New Fast Data Logger and Viewer at Diamond: the FA Archiver* , Proc. of ICALEPCS'2009, Grenoble (France).

[2] M.T.Heron, et al., *Diamond Light Source Electron Beam Position Feedback: Design, Realization and Performance*, Proc. of ICALEPCS'2009, Kobe (Japan).

[3] I. Uzun, et al., *Initial Design of the Fast Orbit Feedback System for Diamond Light Source*, Proc. of ICALEPCS'2005, Geneva (Switzerland).

[4] Xilinx, *Virtex-5 LXT ML555 FPGA Development Kit for PCI Express*, `http://www.xilinx.com/products/boards-and-kits/HW-V5-ML555-G.htm`

[5] CERN, *Open Hardware At CERN-Knowledge Transfer*, `http://knowledgetransfer.web.cern.ch/files/OH-web.pdf`

[6] CERN, *Open Hardware License*, `www.ohwr.org/cernohl`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DESIGN OF EPICS IOC BASED ON RAIN1000Z1 ZYNQ MODULE*

Tao Xue, Hongming Li, Guanghua Gong, Jianmin Li, Tsinghua University, Beijing, China

Abstract

ZYNQ is the new architecture of FPGA with dual high performance ARM Cortex-A9 processors from Xilinx. A new module with Giga Bit Ethernet interface based on the ZYNQ XC7Z010 is development for the High Purity Germanium Detectors' data acquisition in the CJPL (China JinPing under-ground Lab) experiment, which is named as RAIN1000Z1. Base on the nice RAIN1000Z1 hardware platform, EPICS is porting on the ARM Cortex-A9 processor with embedded Linux and an Input Output Controller is implemented on the RAIN1000Z1 module. Due to the combine of processor and logic and new silicon technology of ZYNQ, embedded Linux with TCP/IP sockets and real time high throughput logic based on VHDL are running in a single chip with small module hardware size, lower power and higher performance. We will introduce how to porting the EPICS IOC application on the ZYNQ based on embedded Linux and give a demo of IO control and RS232 communication.

INTRODUCTION OF RAIN1000Z1

ZYNQ [1] is developed by Xilinx for high performance real time processing and embedded application. In the chip of ZYNQ, dual core of ARM Cortex-A9 processor with rich peripherals is the half of the chip, the other half part is the FPGA. People can develop the embedded application in the processor with C programming, such as embedded Linux and uC/OS or VxWorks real time OS can be running on the Cortxe-A9 dual core processors. In the other part of FPGA, people can develop the real time processing logic or DSP function with VHDL/Verilog programming. It is amazing that combine the two parts in one chip, the total footprint is more compact, board space is saved and the power is saved. At the same time, the assistive circuit, such as the on board power circuit, supervisor circuit and debug circuit is simpler.

With this feature of ZYNQ, we develop a readout module for the High Purity Germanium Detectors' data acquisition in the CJPL (China JinPing under-ground Lab) experiment, which is named as RAIN1000Z1 [2]. Figure 1 shows the system architecture of the RAIN1000Z1 module.

The RAIN1000Z1 module is based on the XC7Z010 chip of ZYNQ series, in the CLG225 footprint, which is the smallest package of ZYNQ series. A 256Mbytes DDR3 SDRAM running at 1066MHz and 1000Mbps Ethernet PHY chip 88E1512 from Marvell Inc. are embedded in the RAIN1000Z1 module. Also the power circuit and Flash for firmware saving in QSPI interface are all integrated in the module. Figure 2 is the top view of the RAIN1000Z1 module.

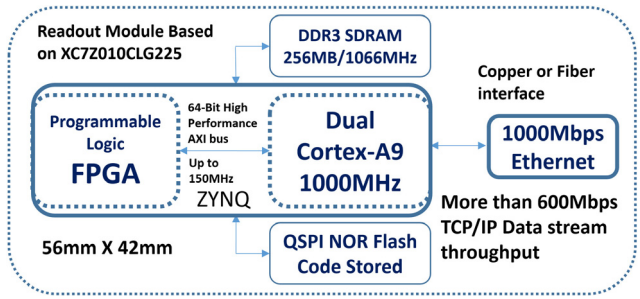


Figure 1: Architecture of the RAIN1000Z1 module.

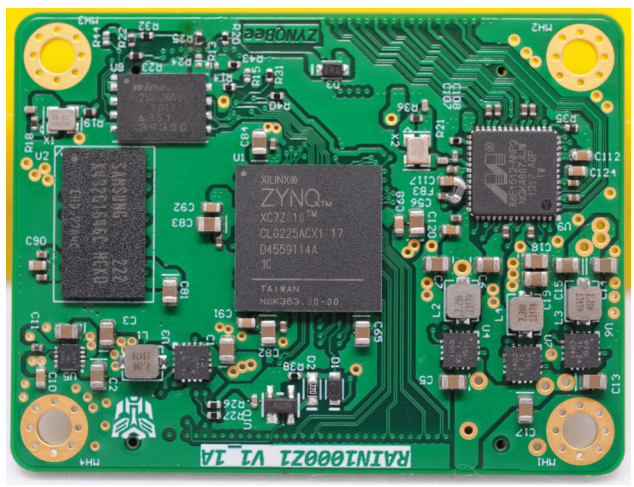


Figure 2: Top view of the RAIN1000Z1 module.

China JinPing underground Laboratory (CJPL) is a deepest underground in the world and provides a very good environment for direct observation of dark matter.

The CDEX experiment will going to direct detect the interaction of WIMP with nucleon in CJPL with high sensitivity in low mass region. Both CJPL and CDEX have got much more progress in recent two years. The CDEX use point contact germanium semi-conduct detector PCGe which has less than 300eV threshold.

The RAIN1000Z1 is used in the readout for HPGe detectors for the CDEX (China Dark Matter Experiment) in CJPL, we can easily get more than 600Mbps TCP/IP data throughput [2].

In the RAIN1000Z1 module, the embedded Linux is running. Currently, we use the version of 3.19 from Xilinx support git hub. Figure 3 shows the booting log of embedded Linux.

Some GPIO and one UARTlite (use the IP core from Xilinx in the Vivado development environment) are im-

*The authors are with Key Laboratory of Particle & Radiation Imaging, Department of Engineering Physics, Tsinghua University, Beijing, China. 100084 (email: gbe.tao.xue@gmail.com).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

plemented in the FPGA part of ZYNQ, and the associated drivers are developed for the embedded Linux.

EPICS PORTING

The EPICS (Experiment Physics and Industrial Control System) is a software environment used to develop and implement distributed control systems to operate devices such as particle accelerators, telescopes and other large experiments [3]. In the next generation of CJPL, which is named CJPL-2, there are more complex and space to develop many physics experiments, we want to use the EPICS as the distributed control system. Then we porting the EPICS on the RAIN1000Z1 and implement an IOC based on the RAIN100Z1 module.

The EPICS software system can running on the Linux OS, and there are many engineers, researchers install the EPICS on Linux [4]. These give a good reference for our application.

```
Booting Linux on physical CPU 0x0
Linux version 3.19.0-xilinx (root@ubuntu) (gcc version 4.9.1 (Sourcery CodeBench
Lite 2014.11-30) ) #11 SMP PREEMPT Mon Oct 12 09:01:35 PDT 2015
CPU: ARMv7 Processor [413fc090] revision 0 (ARMv7), cr=18c5387d
CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine model: xlnx,zynq-7000
cma: Reserved 16 MiB at 0x0f000000
Memory policy: Data cache writealloc
PERCPU: Embedded 9 pages/cpu @4edd3000 s8128 r8192 d20544 u36864
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 65024
Kernel command line: console=ttyPS0,115200 root=/dev/ram rw earlyprintk
PID hash table entries: 1024 (order: 0, 4096 bytes)
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
Memory: 231164K/262144K available (4557K kernel code, 268K rwdata, 1608K rodata,
216K init, 210K bss, 14596K reserved, 16384K cma-reserved, 0K highmem)
```

Figure 3: RAIN1000Z1 Linux booting.

With the embedded Linux, it is not difficult to install the EPICS on RAIN1000Z1 module. The EPICS is a set of software, there are two way to install it on the embedded Linux of RAIN1000Z1. The directly way is compile the EPICS with the gcc compile within the RAIN1000Z1 file system and the other way is cross compile with the arm cross gcc on the Linux (such as Ubuntu) running on a PC [5].

For directly compile in RAIN1000Z1, we need a rootfs with gcc for ZYNQ, and more time is needed, for example, maybe 50 minutes is needed, because the ARM Cortex-A9 is not fast than a CPU in PC. But this way is very simple, we do not need to change the setting of cross compile environment to match the EPICS's compile, although the cross compile is so fast in the PC. Then we use the directly compile in the RAIN1000Z1 module. And a more complete rootfs is built for use with Yocto, the gcc compiler and python support are included.

For the version of EPICS, we use a stable version of EPICS Base 3.14.12.5 (more other version is used, but not described in this paper).

Step 1, Download the EPICS Base (it can be download in PC or directly by RAIN1000Z1).

```
wget http://www.aps.anl.gov/epics/download/base/baseR3.14.12.5.tar.gz
tar xzf baseR3.14.12.5.tar.gz
```

Step 2, Building the EPICS Base. EPICS's build is directed by the EPICS_HOST_ARCH environment variable, the following command is used to set this variable in embedded Linux running on RAIN1000Z1.

```
export EPICS_HOST_ARCH=$(/home/epics/base base-3.14.12.5/startup/EpicsHost
Arch)
```

After setting, this command can check the value:

```
echo $EPICS_HOST_ARCH
```

The "linux-arm" echoed show that EPICS base will be built for Linux OS on ARM architecture which is the Cortex-A9 core in ZYNQ. Then we can build the EPICS Base with the following command:

```
cd ./base-3.14.12.5
make
```

In the platform of RAIN1000Z1, about 40 minutes is needed for building.

Step 3, Start the EPICS Base

After the building, it's possible to start an EPICS IOC, we also implement some GPIO function for IO control and a RS232 port for communication demo. Start EPICS as the following:

```
./bin/linux-arm/softIoc
```

And EPICS will start with a basic command line prompt:

```
epics>
```

type iocInit and we will get the following:

```
Starting iocInit
#####################################
## EPICS R3.14.12.5 $Date: Tue 2015-03-24 09:57:35 -0500$
## EPICS Base built Oct 12 2015
#####################################
iocRun: All initialization complete
epics>
```

And the EPICS Base has been built on embedded Linux of the RAIN1000Z1.

More test is done with the synApps, PyEpics and a GUI window is developed in Ubuntu for IO and RS232 test. The graphical user interface was developed using the EPICS CA client CSS (Control System Studio). And the user interface is execute on a remote PC. The function is running well and more peripherals can be added in the RAIN1000Z1 platform.

CONCLUSION

EPICS is porting on the embedded Linux within the ARM Cortex-A9 processor which is included in the RAIN1000Z1 module. It's suitable to develop more flexible interface and function with this platform. More software in the EPICS software set are under test on the RAIN1000Z1 module and the new result will be show.

REFERENCES

[1] www.xilinx.com/zynq
[2] Tao Xue et al., "Design of Giga Bit Ethernet Readout Module Based on ZYNQ for HPGe", Real Time Conference (RT), 2014 19th IEEE-NPSS Nara, Japan (2014).
[3] www.aps.anl.gov/epics/
[4] prjemian.github.io/epicspi/
[5] wiki-ext.aps.anl.gov/epics/index.php

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A FORMAL SPECIFICATION METHOD FOR PLC-BASED APPLICATIONS

D. Darvas*, E. Blanco Viñuela, CERN, Geneva, Switzerland
I. Majzik, Budapest University of Technology and Economics, Budapest, Hungary

Abstract

The correctness of the software used in control systems has been always a high priority, as a failure can cause serious expenses, injuries or loss of reputation. To improve the quality of these applications, various development and verification methods exist. All of them necessitate a deep understanding of the requirements which can be achieved by a well-adapted formal specification method. In this paper we introduce a state machine and data-flow-based formal specification method tailored to PLC modules. This paper presents the practical benefits and new possibilities of this method, comprising consistency checking, PLC code generation, and checking equivalence between the specification and its previous versions or legacy code. The usage of these techniques can improve the level of understanding of the requirements and increase the confidence in the correctness of the implementation. Furthermore, they can help to apply formal verification techniques by providing formalised requirements.

INTRODUCTION AND MOTIVATION

The complexity of process control systems is steadily increasing, resulting in more and more complex control software. Without an appropriate specification it is increasingly difficult to understand the requirements, hence to develop and maintain the programs.

The motivation of this work originates from CERN (the European Organization for Nuclear Research), where numerous control systems are in use to operate the research facilities. Many of these control systems rely on Programmable Logic Controllers (PLCs). The increasing complexity of the control systems results implies an increasing complexity issue. The non-formal, ad-hoc specification methods in use are less and less able to cope with the complexity of the systems. Therefore we propose a new specification method that aims to handle the increasing complexity of PLC program units.

Obviously, this is not the first work on specification methods for PLCs. *Grafcet* is a standardised specification method [1] based on safe Petri nets. It is convenient to describe finite state machines, but they are not universal enough to be generally used in the development of CERN's control software as the finite state machines are just one part of the final code deployed. *ProcGraph* [2] is one of the recent attempts to improve specifications of PLC programs, but it is on a low abstraction level, too close to the real program code. Other PLC-related specification methods also exist, but none of them seemed to be able to cope with the size and complexity of the PLC programs used at CERN.

Also, there are numerous general-purpose modelling methods that can be used for specifying PLC programs. One of them is the widely-known Simulink Stateflow, that has a complex semantics not adjusted to the needs of the PLC domain, making its usage difficult and potentially error-prone.

The structure of the paper is the following. The next section introduces the main concepts and the structure of the proposed formal specification method. After, the potential benefits of the new method are discussed. Finally, the last section concludes the paper and draws up the future work.

Due to space limitations, the discussion of the syntax and semantics of the specification method is introductory. The reader can find more details in our technical report [3].

MAIN SPECIFICATION CONCEPTS

The goal of our work is to provide a specification method that is a *complete*, *formal* behaviour description of specific functionality which corresponds to one or some PLC components. Therefore it can be used to describe individual PLC components with high precision. Then, these precise descriptions can be naturally composed into complete control systems in the future.

Previous work [4] discussed the requirements towards such a formal PLC specification method. The main requirements can be summarised as follows: (1) providing multiple formalisms adapted to the needs and knowledge of the PLC community, (2) keeping the "core logic" clean by decoupling the input/output-handling, (3) supporting events with proper semantics, and (4) limiting the set of possible errors by limiting the expressivity. The domain-specific requirements are complemented by general requirements, most notably the need for a formal, precise specification formalism, that is also lightweight (can be introduced with small effort) and complete (it describes all required and allowed behaviours). These requirements resulted in PLCspecif, a complete formal specification method for PLC programs.

Structure of the Specification

The main building block of the specification is the *module* that is either a composite module (its behaviour is described by some submodules) or a leaf module (its behaviour is directly described). To provide a specification method that is familiar for the PLC developers, the leaf module descriptions are based on three widely-known formalisms: state machines, data-flow diagrams, and standard PLC timers.

Each module (both composite and leaf modules) is further decomposed into three main parts: (1) input definitions, (2)

* Corresponding author. E-mail: ddarvas@cern.ch, darvas@mit.bme.hu

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

core logic, and (3) output definitions. According to the semantics of PLCspecif, these parts are executed sequentially in a loop, following a structure similar to the cyclic execution scheme of the PLCs.

In the *input definitions* part, named expressions can be defined to simplify the specification. For example, if there are three digital inputs from three buttons (`Button1`, `Button2`, `Button3`), but the program provides the same response for pressing any button, writing "`Button1 OR Button2 OR Button3`" in the core logic makes the understanding more difficult. Instead, the user can specify a `ButtonPressed` expression that is defined once and used later in the core logic. This helps to decouple the physical structure (i.e. three digital inputs) from the concepts to be used in the core logic (i.e. a button was pressed).

Some special inputs called *event inputs* can also be defined. An event input is an expression with Boolean type that has a priority assigned. We call an event input *enabled* if its expression is evaluated to true. In each module only the enabled event input with the highest priority can *trigger*.

The input definitions are followed by the *core logic description* part. It can be described using various formalisms, that are introduced later in this paper.

The *output definitions* part is responsible to assign values to the output variables, based on the input values and on the core logic (e.g. the current state of a state machine). This helps to keep the core logic clean. Including the output variable assignments in a state machine (as entry/exit actions or as transition actions) might make the state machine difficult to overview, therefore it is error-prone.

Optionally, the output definitions part can be followed by *invariant properties*. These are additional requirements and assumptions identified during the specification phase that are not obviously described by the core logic, but have to be satisfied by the module.

Expression Descriptions

The input or output definitions may contain complex expressions. While the arithmetic form is suitable to describe simple expressions (e.g. "`a OR b`"), it does not scale up well. PLCspecif supports the usage of other expression description methods: AND/OR tables and switch-case tables.

AND/OR tables were introduced in the RSML formalism [5], but not widely used since. In an AND/OR table each column represents a case, that is true if in all the rows the value of the expression in the row header equals to the value in the corresponding cell of the case. The whole expression is the OR-connection of the defined cases. The symbol "·" marks that the value of the variable is not taken into account ("don't care"). Figure 1(a) shows an example, representing the `a AND NOT b AND (c OR NOT d)` expression.

Switch-case tables —as their name suggests— are based on similar principles as the case constructs of various programming languages. Figure 1(b) shows an example, representing `_Value`, limited by lower limit `PMin` and upper limit `PMax`.

	Case 1	Case 2
a	true	true
b	false	false
c	true	·
d	·	false

(a) AND/OR table

	_Value < PMin	_Value > PMax	result
Case 1	true	·	PMin
Case 2	false	true	PMax
Case 3	false	false	_Value

(b) Switch-case table

Figure 1: Tabular expression description examples.

Core Logic Descriptions

One single formalism cannot conveniently fit the different types of modules (with state-based, data-flow-oriented and time-dependent behaviour). Therefore we introduced three different types of core logic descriptions.

State Machine. A *state machine module* is composed of hierarchical *states* and *transitions*. A state can be *basic* or *composite* (grouping several basic states together). A transition can go from any state to a basic state[1]. It can have a Boolean expression as condition: the transition can fire only if the expression is evaluated to true. There are two kinds of transitions: *event-triggered* and *non-event-triggered*. A transition is enabled, if its source state is currently active, the condition of the transition is evaluated to true, and if it is event-triggered, the connected event triggers. When an enabled transition *fires*, the current active state of the state machine is changed to the target state of the transition. A transition firing cannot cause any other effects (e.g. it cannot provoke actions or do variable assignments).

The brief semantics of the state machine is the following. First, *all* enabled non-event-triggered transitions are exhaustively fired. Then the triggering event input (i.e. enabled event input with the highest priority) is selected and if there is any enabled transition triggered to this event input, this transition will fire. Finally, all enabled non-event-triggered transitions are exhaustively fired again. Note that in each execution cycle at most one event-triggered transition can fire per state machine module.

In addition, state machines can be extended by deep history states, similarly to UML State Machines.

Input-output Connection Module. State machines are suitable for modules that are stateful and the state to be stored can conveniently represented by a handful of states in the specification. However, if the state can only be described by some integers or real numbers (e.g. storing previous measurements or value requests), state machines are inappropriate and we suggest the usage of *input-output connection modules*. The idea of the input-output connection module is inspired by Function Block Diagrams (FBDs) [6] and similar data-flow-like formalisms. It graphically defines how the outputs of the module should be assigned based on the current inputs and outputs from the previous cycles.

This module description consists of *pins* representing input and output values, and *edges* representing data con-

[1] Transitions going to composite states are not allowed as they could make the semantics of a state machine more difficult to understand.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

nections between pins. Furthermore, it can contain blocks, representing common functions (e.g. logic operations, arithmetic operations, selection), user-defined functions, and platform functions.

Figure 2 shows a simple example. Here, `ValueOutput` keeps *its previous value* if the Boolean input `Sample` is false. If `Sample`=true, the new value of `ValueOutput` will be $-1 \times$ `ValueRequest`.

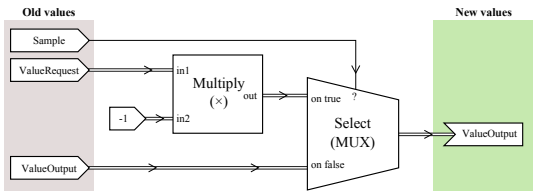


Figure 2: Input-output connection core logic example.

PLC Timers. The state machines and input-output connection modules of PLCspecif do not contain timed behaviours to keep them simple. However, it is crucial to be able to define time-related operations. State machines are often extended by clock variables to describe time, but this method is error-prone, also it does not fit to the existing knowledge of the target group. Instead, we propose to use PLC timers defined in IEC 61131-3 [6] (TP, TON, TOFF). Their semantics is well-known by the developers and they can use these timers confidently.

Example. Figure 3 shows the specification of a simple state machine module. The described component is a combination of a flip-flop and a multiplexer. If the module is enabled, its `Value` output is the `ValueReq` input, limited by `PMin` and `PMax`. The module can be made enabled by having a true signal in one of the `EnableReq` inputs. If there is a rising edge on the `DisableReq` input, the module will be disabled, and in this state the `Value` output will be 0. Disabling the module has priority over enabling it. The module keeps its state if no enable or disable request is received.

In the example one can observe the structure and general elements of the specification, the decoupled input/output handling, and the different ways of specifying expressions. To help the understanding, each part of the specification can be annotated by textual descriptions, however we omitted them in this example to reduce its size.

BENEFITS

This section summarises the different benefits of using a formal specification method tailored to the PLC domain, such as PLCspecif.

Improved Understanding

Software of control systems, especially the ones evolving for long time can become complex. Proper documentation and specification is primordial to be able to fully understand such PLC program. This understanding is necessary both for reusing and maintaining the components.

ExampleModule

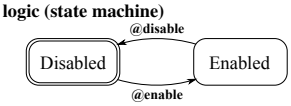
Assigned inputs:
- ValueReq : INT16
- EnableReq_fromLogic : BOOL
- EnableReq_fromScada : BOOL
- EnableReq_fromField : BOOL
- DisableReq : BOOL
- PMin : INT16 param
- PMax : INT16 param

Assigned outputs:
- Value : INT16
- Status : BOOL

Input definitions: — (none)

Event definitions:
- @disable ⇐ `rising_edge`(DisableReq) (`pri`=1)
- @enable ⇐ EnableReq_fromLogic OR EnableReq_fromScada OR EnableReq_fromField (`pri`=2)

Core logic (state machine)



Output definitions:
- _Value =

ValueReq < PMin	ValueReq > PMax	result
T	·	PMin
F	T	PMax
F	F	ValueReq

- Value =

in_state(Enabled)	result
T	_Value
F	0

- Status = `in_state`(Enabled)

Invariant properties:
- ALWAYS *PMin* ≤ *Value* ≤ *PMax* ASSUMING *PMin* ≤ *PMax*

Figure 3: Example module specification.

A simple, restricted, domain-specific formal specification method that matches target platform (PLC) as PLCspecif can help the users to express their requirements and intents confidently. Decoupling the input/output handling also helps to have a clean, easy-to-understand specification, where the core logic can be simple. PLCspecif provides a simple specification method with simple semantics, especially suitable for the needs of the PLC-based process control domain. For example, handling rising and falling edges is a base feature of PLCspecif, as it is an often used feature in PLC programs. Its manual implementation in similar formalisms usually involves concurrency, that can make the behaviour of the module less intuitive and less understandable.

Consistency Checking and Formal Verification

A specification with formal semantics may not only improve the understanding of the specifiers and developers, but it opens the door for formal verification as well. Different requirements can be checked on the specification directly using formal techniques, such as model checking: *general properties* (e.g. the model does not contain deadlock, livelock or nondeterminism) and *application-specific properties* (expressing additional requirements and assumptions). These properties can be checked in the specification phase on the formal specification, as well as later on the implementation. For example, in Figure 3, a safety property expresses the assumption that the *Value* output will be between the two given limits: "ALWAYS `PMin` ≤ `Value` ≤ `PMax`" assuming that the `PMin` value is not above `PMax`. However, using formal verification it can be shown in the specification phase that

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

the property is violated if `PMin` > 0. The given counterexample demonstrates that if the module is in disabled state, the `Value` is set to zero, even if this is out of the `PMin`..`PMax` range.

The verification is made possible by the definition of formal semantics for PLCspecif, defined as a transformation to a lower-level formalism: timed automaton. Timed automaton have a well-defined semantics, also this representation can be directly the input of a suitable model checker tool.

A formal specification can also be the source of automated test generation, thus to use a variety of verification methods.

Code Generation

A complete, formal specification contains the behaviour description that is required to implement the PLC module. Based on this knowledge, a developer can manually create the code, or this process can be automated. The basis of the code generation is the formal semantics of PLCspecif. The code generation method is constructed as a systematic representation of the structures of the timed automata describing the formal semantics in a PLC programming language. This way the generated code can be correct by construction.

While the specification contains the behaviour of the module, the code generator shall be configurable to select implementation alternatives in order to provide a code expected by the user that is easily understandable. In case of PLCspecif, the user has influence on the structure of the code (which submodules should be extracted into functions or function blocks, how to represent state machines, etc.). Based on the specification and the configuration of the way of implementation, it is possible to generate code for PLCs. Currently, our proof-of-concept code generator supports the Siemens implementation of the IEC 61131-3 standard ST (Structured Text) language [6], the Structured Control Language (SCL).

A common counterargument against PLC code generation is the fact that most code generators provide complex, unmaintainable code, while it is required to have full understanding of the generated PLC code. By careful design and configurable structure PLCspecif allows to generate readable code, following the structure of the specification.

Equivalence and Conformance Checking

Equivalence and conformance checking is the process of comparing the behaviour of models with formal semantics. It allows to compare two models against each other by checking for example if they provide the same outputs for the same input sequences. This can show the equivalence of two modules with different internal structure. Besides this use case, conformance checking can be applied to check implementation against specification (e.g. if a manual implementation is conformant to a specification, or if a specification properly captures the behaviour of a legacy implementation).

Different equivalence and conformance relations have been defined with different sensitivity. This way the user can focus on the differences that are relevant from the point of view of the correctness of the PLC program.

These equivalence/conformance relations can be checked on the timed automata constructed for the formal semantics description, or on execution traces. Comparison between specification and implementation is allowed by previously designed methods [7] for building models of PLC code.

Examples and Experiences

As an initial validation of the specification method, we mention our experiment that targeted one of the base objects used in CERN's UNICOS framework [8]. The PLCspecif specification was made on the basis of understanding the behaviour (by having discussions with the developers) of the OnOff object, that is the UNICOS representation of a binary state equipment (e.g. valve, heater, pump). By being able to capture and clarify the intended behaviour of a PLC code of 600 lines, we demonstrated that PLCspecif can scale up to the size of real PLC program components. Code generation was also performed and the produced code was tested in the test bench used for UNICOS modules.

CONCLUSIONS AND FUTURE WORK

In this paper we have introduced PLCspecif, a formal behaviour specification method for PLC programs. Using such a method can provide benefits in many ways: it improves understanding, it can be a basis of verification and automated implementation. Our plans for future work includes the definition of extensions that address the integration of modules, to support the specification of the software of complex control systems.

REFERENCES

[1] *IEC 60848:2013 GRAFCET specification language for sequential function charts*, IEC Std., 2013.

[2] T. Lukman, G. Godena, J. Gray, M. Heričko, and S. Strmčnik, "Model-driven engineering of process control software – beyond device-centric abstractions," *Control Engineering Practice*, vol. 21, no. 8, pp. 1078–1096, 2013.

[3] D. Darvas, E. Blanco, and I. Majzik, "Syntax and semantics of PLCspecif," CERN, Report EDMS 1523877, 2015, `https://edms.cern.ch/file/1523877/`.

[4] D. Darvas, I. Majzik, and E. Blanco, "Requirements towards a formal specification language for PLCs," in *Proc. of the 22th PhD Mini-Symposium*. BUTE DMIS, 2015, pp. 18–21.

[5] N. Leveson, M. Heimdahl, H. Hildreth, J. Reese, and R. Ortega, "Experiences using statecharts for a system requirements specification," in *Proc. of the Sixth Int. Workshop on Software Specification and Design*. IEEE, 1991, pp. 31–41.

[6] *IEC 61131-3:2013 Programmable controllers—Part 3: Programming languages*, IEC Std., 2013.

[7] B. Fernández, D. Darvas, J.-C. Tournier, E. Blanco, and V. M. González, "Bringing automated model checking to PLC program development – A CERN case study," in *12th Int. Workshop on Discrete Event Systems*. IFAC, 2014, pp. 394–399.

[8] E. Blanco *et al.*, "UNICOS evolution: CPC version 6," in *Proc. of the 13th Int. Conf. on Accelerator & Large Experimental Physics Control Systems*, 2011, pp. 786–789.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PLCverif: A TOOL TO VERIFY PLC PROGRAMS BASED ON MODEL CHECKING TECHNIQUES

D. Darvas*, B. Fernández Adiego, E. Blanco Viñuela, CERN, Geneva, Switzerland

Abstract

Model checking is a promising formal verification method to complement testing in order to improve the quality of PLC programs. However, its application typically needs deep expertise in formal methods. To overcome this problem, we introduce PLCverif, a tool that builds on our verification methodology and hides all the formal verification-related difficulties from the user, including model construction, model reduction and requirement formalisation. The goal of this tool is to make model checking accessible to the developers of the PLC programs. Currently, PLCverif supports the verification of PLC code written in ST (Structured Text), but it is open to other languages defined in IEC 61131-3. The tool can be easily extended by adding new model checkers.

INTRODUCTION AND MOTIVATION

Operating an accelerator complex to provide facilities for particle physics research involves numerous process control tasks. Many of them (such as cooling and ventilation, cryogenics, gas systems) are controlled by Programmable Logic Controllers (PLCs) at CERN, the European Nuclear Research Organization. As these systems are critical for the operation of CERN, the correctness of the executed PLC applications is a high priority.

Testing is a widely used solution to find potential failures in software. However, testing is not an universal solution for the verification of programs, for the following main reasons:

- Testing cannot show the absence of bugs, it can only show their presence.
- Testing can only check the outputs given by the software under test for some *selected input sequences* (test inputs). It cannot check efficiently general statements (e.g. "If output `FireAlarm` is true, then the output `NoAlarmPresent` should always be false.") or liveness properties (e.g. "If a request is received, a response will be sent *eventually*.").

Model checking is a good candidate to *complement* testing in order to reduce these weaknesses [1]. This paper introduces the high-level concepts of model checking and our proposed solution to incorporate model checking in the PLC software development process.

CHALLENGES OF MODEL CHECKING

Model checking is a formal verification technique that takes (1) a mathematical *model* of the system to be checked and (2) a formalized *requirement*. The model checker algorithms can decide if the given requirement is satisfied for the given model or not. Contrarily to testing, model checking checks *all possible* executions of the program and reports if any of them violates the requirement. Model checking is also able to generate *counterexamples*, i.e. input sequences demonstrating the violation of the given requirements.

However, model checking is not a silver bullet. There are three main obstacles of using this technique:

1. The model checker tools need a mathematical representation of the program. Constructing them needs lots of effort and experience in the formal methods domain.
2. The requirements should also be formalized for model checking. This is a similarly challenging task.
3. Model checking is computation- and memory-intensive. The generated models of the programs are often too large or too complex to be analysed with the available computation capacity.

These obstacles are difficult to overcome. The available tools require deep expertise in the formal verification domain. This could be the main reason why model checking is not widely used in industry yet, apart from some highly safety-critical applications in avionics, railway industry, etc.

Our goal is to provide a model checking solution for the PLC domain by overcoming the mentioned obstacles. All of them contain both theoretical and technical challenges. In earlier work [2, 3] we have provided solutions for the theoretical obstacles. All these solutions have been incorporated in a tool called PLCverif[1] that makes model checking accessible for the developers in the PLC domain. This paper focuses on this tool and on the bridge between the formal methods and PLC domains.

PLCverif: A BRIDGE BETWEEN FORMAL VERIFICATION AND PLC DOMAINS

In this section we overview the main features of the PLCverif tool. We focus on the user's point of view, therefore the structure of this section follows the normal workflow of a user.

Typical Workflow

The typical user workflow of the PLCverif tool consists of four steps:

1. Defining (importing or writing) the PLC code to be checked,
2. Defining the requirement to be verified,
3. Executing the verification,
4. Evaluating the results of the verification.

In the following sections, each step is described in detail.

* E-mail of the corresponding author: daniel.darvas@cern.ch

[1] `http://cern.ch/plcverif/`

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Defining the PLC Code. The PLCverif tool provides an editor for PLC programs. The aim is to support the languages defined in the standard IEC 61131-3 [4]. Currently the tool supports SCL (Structured Control Language), which is the Siemens' implementation of the standard, high-level language ST (Structured Text). Additionally, SFC (Sequential Function Chart) and STL (Statement List) are partially supported. The code editor of the PLCverif (Figure 1) provides the main features required nowadays in modern development tools, e.g. syntax highlighting, content assist, support for refactoring. The PLC code to be verified can either be written in this PLC code editor, or imported if the program is already existing.

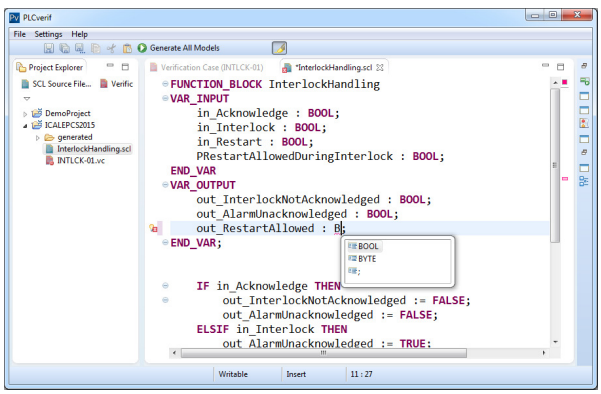


Figure 1: PLC program editor.

Defining the Requirement. Similarly to the test cases in testing, a *verification case* should be defined by the user. A verification case contains all necessary information for the verification. The user has to define:

- The general information of the verification case (ID, name, description),
- The source code to be checked,
- The requirement to be checked,
- The model checker tool to be used.

In addition, the user has the possibility to fine-tune the verification method by setting some parameters (e.g. the way the model is simplified by reductions). These parameters are set automatically using various heuristics. For example, the user can provide assumptions about the input variables (e.g. the value of an input variable is always false) to facilitate the task of the model checker, however this is optional.

The verification case can be edited on a form that can be seen in Figure 2.

Requirement patterns As previously mentioned, the requirement has to be formalised for the model checker. Typically, requirements are formalised using CTL (Computational Tree Logic) or LTL (Linear Temporal Logic). Both formalisms are difficult to be used for non-experts. Furthermore, even with experience it is easy to make semantic mistakes in the formalisation of requirements.

To avoid these issues, we have introduced a predefined set of *requirement patterns*, similarly to [5, 6], focused on the PLC domain. In our method, a requirement pattern is:

- A precise English sentence with gaps for logic expressions (e.g. "*If* $\ldots^1$ *(at the end of the PLC cycle), then* $\ldots^2$ *is always true (at the end of the same cycle).*"); and
- A formalization of the textual representation in LTL or CTL using the same gaps. (e.g. "AG $\left((EoC \wedge \ldots^1) \rightarrow \ldots^2\right)$", where *EoC* stands for end of cycle).

The task of the user is to choose a requirement pattern that corresponds to the requirement to be checked, and to fill the gaps of the pattern. Each gap has to be filled by an expression containing constants, variables, and logic operators (e.g. AND, OR).

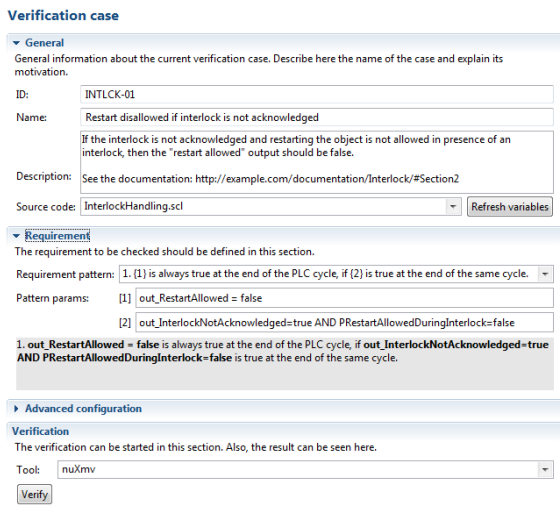


Figure 2: Verification case definition form.

Verification Process. After loading or writing the PLC code and providing the verification case, the verification procedure is fully automated.

In the background PLCverif performs the following steps, completely hidden from the user:

1. The formal, mathematical requirement (in LTL or CTL) is produced based on the given information.
2. The PLC code is parsed and translated into an intermediate model.
3. The intermediate model is reduced using various techniques [3]. The goal of them is to simplify the model without modifying its meaning, and to remove any part of the model that is not necessary for the evaluation of the current requirement. Consequently, for each requirement a unique verification model is produced.
4. The reduced intermediate model and the given requirement is then converted to the input syntax of the selected model checker tool.
5. When the model and the requirement is produced, the model checker tool is invoked. All its outputs, including

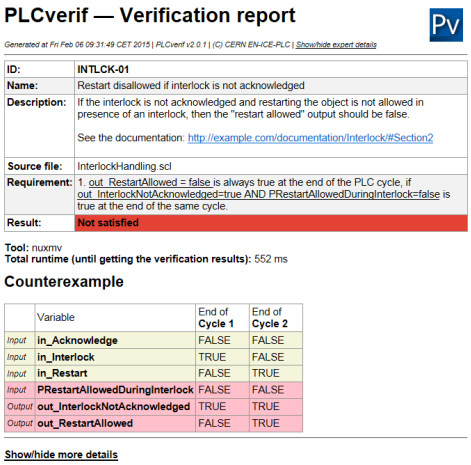
Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

the error messages are stored in the PLCverif tool to provide feedback to the user.

Evaluating the Results. The output of the model checker tools (the counterexamples) are typically difficult to understand. For example, the counterexamples can be huge, they have to be reduced before manual analysis. Also, the referred variable names can be different from the ones defined in the PLC code due to the automated generation process and the various restrictions of the formalisms. These have to be replaced by the names meaningful for the user.

The result of this phase is the verification report (see Figure 3 for example) — a self-contained document produced automatically that includes the details of the verification case, the result of the verification and the reduced counterexample, if applicable.



PLCverif — Verification report

Generated at Fri Feb 06 09:31:49 CET 2015 | PLCverif v2.0.1 | (C) CERN EN-ICE-PLC | Show/hide expert details

ID:	**INTLCK-01**
Name:	Restart disallowed if interlock is not acknowledged
Description:	If the interlock is not acknowledged and restarting the object is not allowed in presence of an interlock, then the "restart allowed" output should be false. See the documentation: http://example.com/documentation/Interlock/#Section2
Source file:	InterlockHandling.scl
Requirement:	1. out_RestartAllowed = false is always true at the end of the PLC cycle, if out_InterlockNotAcknowledged=true AND PRestartAllowedDuringInterlock=false is true at the end of the same cycle.
Result:	**Not satisfied**

Tool: nuxmv
Total runtime (until getting the verification results): 552 ms

Counterexample

	Variable	End of Cycle 1	End of Cycle 2
Input	**in_Acknowledge**	FALSE	FALSE
Input	**in_Interlock**	TRUE	FALSE
Input	**in_Restart**	FALSE	TRUE
Input	**PRestartAllowedDuringInterlock**	FALSE	FALSE
Output	**out_InterlockNotAcknowledged**	TRUE	TRUE
Output	**out_RestartAllowed**	FALSE	TRUE

Show/hide more details

Figure 3: Verification report.

Applied Technologies

The PLCverif tool is implemented as an *Eclipse RCP* application. This provides a standalone, multi-platform tool with an environment (look and feel) familiar to the users with moderate development effort. The PLC codes are parsed using *Xtext*. This framework provides editor, parser and an object model based on the defined grammar of the input language, that is SCL in our case.

Parsed programs are then translated to an intermediate model representation. This is a formalism conceptually close to the input of many model checking tools. This model is implemented using *Eclipse Modeling Framework*. The intermediate models are then translated into the concrete input syntax the model checker tools. The translation is developed using *Xtend*.

Currently the format of the NuSMV/nuXmv, UPPAAL and BIP tools are supported, but other, similar model checker tools can be easily integrated into PLCverif.

PLCverif has a command line interface too. It allowed us to set up a "continuous verification" workflow, where a *Jenkins* job automatically (re-)checks all verification cases on each SVN commit, then the results are sent in e-mail.

CASE STUDY

To illustrate our method, a simple example for the previously presented workflow is shown. Listing 1 shows a code excerpt, extracted from a base object of the UNICOS framework [7]. The original code is much larger, containing around 200 variables and 600 lines of code. The following steps correspond to the steps of the workflow presented before.

1. As a first step, the user writes or imports the example PLC code in Listing 1 to the tool.
2. The next step is the requirement specification. The example informal requirement is the following: *if the interlock is not acknowledged and restarting the object is not allowed in presence of an interlock, then the "restart allowed" output should not be true.*
   For this informal requirement a pattern should be chosen. The requirement pattern presented before is convenient to describe this requirement.
   Filling that pattern results in the following:
   > "If `out_InterlockNotAcknowledged`=true and `PRestartAllowedDuringInterlock`=false (at the end of the PLC cycle), then `out_RestartAllowed`=false is always true (at the end of the same cycle)."
3. Once the PLC code and the requirement are defined, the user should press the "Verify" button, every step is then automated. The model generation, the model reductions, invocation of the model checker and generation of the verification report takes less than a second in this case[2].
4. The verification report (Figure 3) shows that the requirement is not satisfied. A counterexample is also provided for the user. The counterexample can be seen in Table 1. Executing the code with the inputs shown in the table results that at the end of the second PLC cycle `out_InterlockNotAcknowledged`=true and `out_RestartAllowed`=true is possible, violating the requirement defined above.

After knowing that the requirement is not satisfied, using the counterexample, the violation can be reproduced by the developer. Investigation of the code can show the source of this problem: a a pair of parentheses is missing from the condition in lines 19–20 of Listing 1 (see the yellow marks). After fixing this issue the requirement will be satisfied.

This example highlights the main differences between testing and model checking:

- It is enough to define the requirement. Using testing, careful test planning and numerous test cases would have been necessary to check this requirement.
- After fixing the problem, model checking can even prove that the given requirement is satisfied. Contrarily, testing can only show the presence of bugs.

[2] The measurement was executed on a PC with Intel Core i7-3770 CPU, 8 GB RAM on Windows 7 x64. The selected model checker tool was nuXmv 1.0.1.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

```
FUNCTION_BLOCK InterlockHandling
VAR_INPUT
       in_Acknowledge : BOOL;
       in_Interlock : BOOL;
       in_Restart : BOOL;
       PRestartAllowedDuringInterlock : BOOL;
END_VAR
VAR_OUTPUT
       out_InterlockNotAcknowledged : BOOL;
       out_AlarmUnacknowledged : BOOL;
       out_RestartAllowed : BOOL;
END_VAR;
    IF in_Acknowledge THEN
        out_InterlockNotAcknowledged := FALSE;
        out_AlarmUnacknowledged := FALSE;
    ELSIF in_Interlock THEN
        out_AlarmUnacknowledged := TRUE;
    END_IF;
    IF (in_Restart AND NOT in_Interlock) OR
       (PRestartAllowedDuringInterlock AND in_Restart AND
            in_Interlock)
      AND NOT out_InterlockNotAcknowledged THEN
        out_RestartAllowed := TRUE;
    END_IF;
    IF in_Interlock THEN
        out_InterlockNotAcknowledged := TRUE;
        out_RestartAllowed := FALSE;
    END_IF;
END_FUNCTION_BLOCK
```

Listing 1: Example PLC code.

Table 1: Counterexample for the Example Requirement

Variable	Cycle 1	Cycle 2
in_Acknowledge	FALSE	FALSE
in_Interlock	TRUE	FALSE
in_Restart	FALSE	TRUE
PRestartAllowedDuringInterlock	FALSE	FALSE
out_InterlockNotAcknowledged	TRUE	*TRUE*
out_RestartAllowed	FALSE	*TRUE*

Even checking this small PLC code can show some of the advantages of model checking. The code above is an excerpt from a real PLC program. We were able to find the same fault using the same requirement on the whole PLC program, without knowing its presence a priori. As the number of inputs in the original program is much higher, it would be extremely difficult to be checked using testing.

CONCLUSION AND FUTURE WORK

We have presented an automated method for model checking PLC programs. This method is incorporated by the tool PLCverif that allows users not expert in formal methods to use model checking without a need for long training. We have applied this method for many PLC programs at CERN and we have found several problems in supposedly well-tested, mature PLC programs, including the example presented above.

Again, model checking is not a solution for all verification-related challenges. The models generated from PLC programs can be huge, even after using reduction techniques. If a requirement can be checked using a couple of test cases, then testing can have an advantage. Nevertheless, model checking can complement testing by providing solutions for different types of requirements, where testing can be inefficient or practically impossible.

Related Work. There are only a few of available tools providing formal verification of PLC code: the most known is Arcade.PLC [8] that focuses on model checking of PLC code. While they provide solutions for hiding the difficulties of building the formal models, the requirements should be provided directly in CTL or LTL. Many other authors propose other solutions, but their tools are not available (not downloadable, nor described in detail in any paper). Another constraint is the support of the ST language which is not common in those tools.

Future Work. The future work is twofold. First, the reductions and model checking algorithms could be improved to increase the set of verifiable problems. The second is coming from the fact that model checking needs unambiguous requirements to be verified. If there is no formal specification for the program, it is difficult to extract these requirements. Therefore we are working on the design of formal specification methods for the PLC domain [9].

REFERENCES

[1] B. Fernández, E. Blanco, and A. Merezhin, "Testing & verification of PLC code for process control," in *Proc. of the 14th Int. Conf. on Accelerator & Large Experimental Physics Control Systems*, 2013, pp. 1258–1261.

[2] B. Fernández, D. Darvas, J.-C. Tournier, E. Blanco, and V. M. G. Suárez, "Bringing automated model checking to PLC program development – A CERN case study," in *Proc. of the 12th Int. Workshop on Discrete Event Systems*. IFAC, 2014, pp. 394–399.

[3] D. Darvas, B. Fernández, A. Vörös, T. Bartha, E. Blanco, and V. M. G. Suárez, "Formal verification of complex properties on PLC programs," in *Formal Techniques for Distributed Objects, Components, and Systems*, ser. LNCS. Springer, 2014, vol. 8461, pp. 284–299.

[4] *IEC 61131-3:2013 Programmable controllers—Part 3: Programming languages*, IEC Std., 2013.

[5] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *Proc. of the 21st Int. Conf. on Software Engineering*. ACM, 1999, pp. 411–420.

[6] J. C. Campos, J. Machado, and E. Seabra, "Property patterns for the formal verification of automated production systems," in *Proc. of the 17th IFAC World Congress*. IFAC, 2008, pp. 5107–5112.

[7] E. Blanco *et al.*, "UNICOS evolution: CPC version 6," in *Proc. of the 13th Int. Conf. on Accelerator & Large Experimental Physics Control Systems*, 2011, pp. 786–789.

[8] S. Biallas, J. Brauer, and S. Kowalewski, "Arcade.PLC: A verification platform for programmable logic controllers," in *Proc. of the 27th IEEE/ACM Int. Conf. on Automated Software Engineering*. ACM, 2012, pp. 338–341.

[9] D. Darvas, E. Blanco, and I. Majzik, "A formal specification method for PLC-based applications," in *Proc. of the 15th Int. Conf. on Accelerator & Large Experimental Physics Control Systems*, 2015, in press.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CXv4, A MODULAR CONTROL SYSTEM

Dmitry Bolkhovityanov, Pavel Cheblakov, Fedor Emanov,
Budker Institute of Nuclear Physics, SB RAS, Novosibirsk, Russia

Abstract

CX control system is used at VEPP-5 and several other BINP facilities. CX version 4 is designed to provide more flexibility and enable interoperability with other control systems. In addition to device drivers, most of its components are implemented in a modular fashion, including data access at both client and server sides. The server itself is a library. This approach allows clients to access several different control systems simultaneously and natively (without any gateways). CXv4 servers are able to provide data access to clients from diverse CS architectures/protocols, subject to appropriate network module being loaded. The server library, coupled with "null link" client-server access module, allows to create standalone monolythic programs for specific small applications (such as test benches and device test screens/utilities) using the same ready code from large-scale control system but without its complexity.

CXv4 design principles and solutions are discussed and first deployment results are presented.

CX

CX was designed at BINP in late 1990s as a general control system framework which runs on Linux and several flavors of *NIX. Initially made for CAMAC, CX now supports a wide range of hardware attached via different fieldbuses, including PCI/CompactPCI, VME, CANbus, RS232/485 and Ethernet.

CX is used at VEPP-5 Injection Complex [1] and at several other BINP facilities and small-scale experiments [2–4].

VEPP-5 will supply electrons and positrons to two BINP machines — VEPP-4 [5] and VEPP-2000 [6] in the near future. VEPP-4 uses a mix of an inhouse-designed software (with roots dating back to 1970s) [7] and EPICS; VEPP-2000 employs a custom software named VCAS [8]. For VEPP-5 to carry out its mission, its control system must be able to communicate with partners' control systems.

THE PROBLEM OF INTEROPERATION

CX, like most control system frameworks used in high energy physics experiments, is distributed and is based on a 3-layer model (Fig. 1a). A typical framework is closely tied to some network protocol and is often designed jointly with a dedicated protocol. I.e., the client library implements the client side of a protocol, and server implements its side (Fig. 1b); CX belongs to this class.

However, a need to interact with a different control system framework arises sooner or later in real world. This is often solved via "gateways" — dedicated software, translating one protocol to another (Fig. 2). This approach has its disadvantages besides a necessity for an additional software component. Not only does such gateway have to convert between protocols, but sometimes between different data paradigms, it convering in a very inconvenient place.

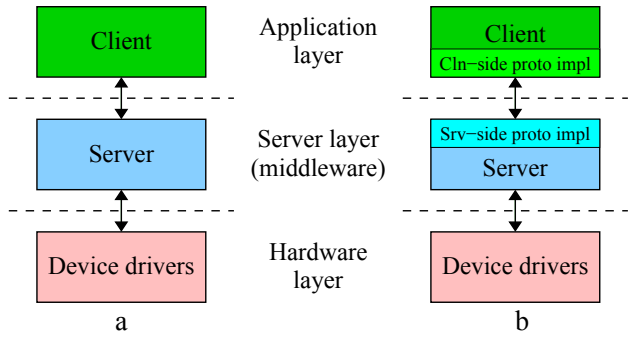


Figure 1: 3-layer architecture. (a) General layout; (b) Client-server communication.

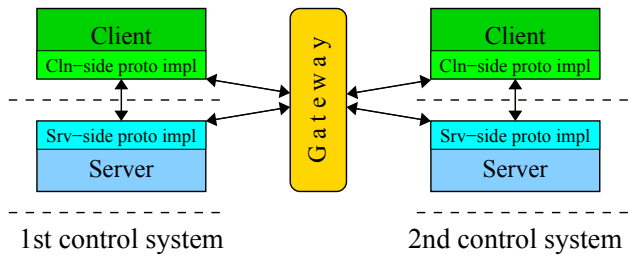


Figure 2: Interaction between different control systems via a gateway.

SOLUTION

In authors' opinion [9, p.3], the key to solution is separation of network protocol specifics and implementation from both client libraries and server. That was done in CX version 4 (see Fig. 3).

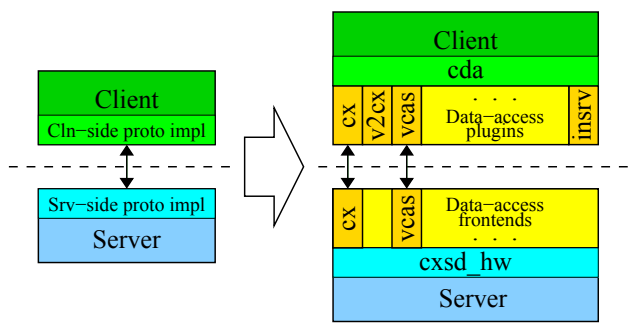


Figure 3: CXv4 modular structure.

Client library doesn't interact with server directly but rather via plugins. The library core provides clients with

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

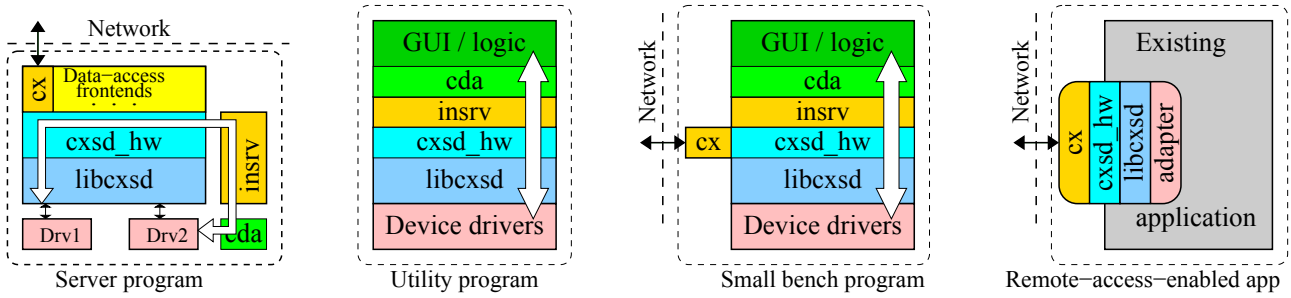


Figure 4: Possible combinations of CX modules.

protocol-agnostic access to data, hence its name "CDA" being Cx Data Access. Data-access plugins are free to implement any communication protocol; this enabled to support old CX versions seamlessly via "v2cx" plugin.

The situation is mirrored on a server side. Server core isn't engaged in network interaction, but just maintains an operating environment for data (channels) and drivers. External access to data is provided by plugins so-called "data frontends". Frontends access data via a simple API provided by `cxsd_hw` module, and are also free to implement any communication protocol.

Server code is implemented as a library (libcxsd), so that the server binary (cxsd) should be just a "conductor" that controls an orchestra of software modules (libcxsd, frontends, drivers, libraries, etc.).

Since most components of client and server sides are modules, these modules can be combined in several interesting ways.

- Device drivers can access channels from neighbor drivers in the same way as clients via "insrv" plugin/frontend (Fig. 4a). I.e., a unified API is used for both local and remote access.
  This gives freedom to place parts of control system (such as calculations or control logic) at either client or server side.
  Besides, data from other servers (and control systems) is accessed in exactly the same way (albeit via a different protocol(s)).
- The same `insrv::` "null-link" can be used to combine GUI and server with drivers into a single simple application without any need for network interaction (Fig. 4b).
  This is handy for small tasks such as a test bench for some type of device.
- A similar setup with a network frontend added (Fig. 4c) is suitable for small facilities where a full-scale 3-layer control system would be an overkill.
  This setup combines simplicity of a monolythic application with ability to access control channels remotely (which is useful for various tools like archivers).
- Implanting `libcxsd` with `cx` frontend to an existing application (such as an "all-in-one" test bench program) is a cheap way to add it remote access abilities (Fig. 4d).

IMPLEMENTATION CHALLENGES

Different Data Paradigms

Approach to data management differs between control systems.[1] Generally the concept of "channels" is used. A channel is a named data entity which can be atomically read and/or written to. However, more variants, such as properties, commands, methods, etc. exist.

To minimize interoperation problems CX uses as simple approach as possible:

- Any object addressable in a control system is a channel.[2] Channels can be either read-only or read-write.
- Commands are presented as channels. Writing to "command channel" triggers an action; value written can be used as parameter.
  This convention is implemented at device driver level and doesn't require any specific handling at other levels.
- Channels are addressed by names at the client side. A name can begin with a "PROTOCOL::" prefix which selects a way to access data and is usually the name of a corresponding protocol or control system (missing prefix taken to be "CX::").
  What goes after PROTOCOL:: is up to data-access-plugin and is passed to it verbatim.

Main Loop Integration

Any non-trivial application or application framework implements some kind of a main loop. Interaction with the main loop can use different models: callbacks in Xt/Motif, signals/slots in Qt, events in LabWindows, select()/poll() in console programs, etc. This diversity of main-loop paradigms presents serious problems when making code which should be able to "live" in any environment.

However, let's look what is required for vital activity of an I/O library or a device driver.

1. To watch for events on a file descriptor (typically readyness for read and/or write).
2. To request a timeout (either after some period or at exact time).

[1] Probably some "ideal way" to operate data in control systems do exist (as well as "idea of data", in platonian sense), but authors are unaware of any definitive work on this subject.

[2] This is similar to *NIX approach "everything is a file".

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

This functionality is present in any main loop implementation. And since this functionality is very limited and simple it is possible to create a formal API with different implementations for different main loops.

Such API was created for CX-server circa 2005. It it is called CXscheduler [10] in a "native" console form and implements the event-driven main loop based on `select()` syscall. API implementations (adapters) for Xt/Motif and Qt exist, libev and LabWindows implementations are under development.

Since all levels and components of CX employ CXscheduler all of them can be used in either console or GUI applications in a uniform way.

Modularity

Since the early days of CX device drivers aren't statically linked into CX server binary but are loaded at run-time, via `dlopen()`.

This approach was extended in v4: data-access modules, frontends, screen instruments in GUI programs are implemented as plugins. Even various configuration-file readers (such as hardware configuration for server, screen description for screen manager) are plugins; this allows us to use external macro-processors (currently m4) or some database instead of files.

Plugins can be either dynamically loaded or linked in statically (for platforms without `dlopen()`); care is taken of this by a dedicated "cxldr" component.

PERFORMANCE

Additional functionality was expected to come at some performance cost.

- Plug-ins management and data routing between plugins and core libraries obviously require some CPU time, which was hard to estimate in advance.
- Initial versions of CX had a limitation for scalar data at device drivers' level to be 32-bit integers only, since this fits most control hardware and significantly simplifies programming. CXv4 removes this limitation, supporting data of various formats (int, float, text) and of different sizes; conversion is performed automatically, if required. This was also expected to cost some performance loss.
- Performance can be especially critical for the most resource-starved pieces of control system computer hardware — CAMAC and CAN intelligent controllers [11] running Linux on 50MHz PowerPC CPU.

However, simple benchmarking shows that neither additional modularity nor more flexible data model cause any performance degradation. And software in intelligent controllers even got several percent performance gain due to optimizations in remote-driver-environment library.

CXv4 DEPLOYMENT

This part of work has been supported by Russian Science Foundation (project N 14-50-00080).

VEPP-5

Drivers for a whole range of hardware used at VEPP-5 were ported from CXv2 to CXv4 during 2014–2015. Control system had switched from v2 to v4 during summer'2015 maintenance stop.

Besides regular data-access plugins, a special "formula scripting" CDA plugin is used. It allows us to perform simple calculations like "return value of $channel1 + channel2 * channel3$" (for read channels), as well as simple sequences like *"put user-input values to channel1 and channel2; pause for 2s; put 1 to channel3"* (for write channels). Availability of this simple scripting in a screen-manager application gives freedom of where to place such macro-actions either at a server level or at a client side, and allows us to easily debug it first in the former case.

VEPP-2000 Interaction

VCAS support has been implemented and tested. Further works on interoperation between VEPP-5 and VEPP-2000 control systems will continue when VEPP-2000 resumes operation in 2016.

Electron Beam Welding Facility

BINP electron beam welding facility is of small scale. It employs a small (10 devices, depending on experiment), but diverse set of control hardware (including CAMAC, CANbus and RS485). Thus, 3-layer control system was used from the very beginning.

However, a full 3-layer software stack is a bit excessive for a small facility. Modular nature of CXv4 enabled to unite GUI and server parts into a single application, as shown on Fig. 4c (and there are several variants depending on experiment being carried out). As this application includes a CX network data-access frontend, benefits of a 3-layer control system are retained.

LIA-2

LIA-2 still runs CXv2 but all required device drivers are ready, and high-level software is under development; switch to CXv4 is scheduled to 2016.

FUTURE AND PROSPECTIVE WORKS

As the key point of CXv4 is its flexibility and ability to interact with other control systems, the main direction of development is expanding the repertoire of data-access plugins and frontends.

EPICS Integration

The nearest target is interaction with EPICS. At present time, client-side access to EPICS is implemented at Python client library; it will be moved to a CDA plugin. EPICS/CA

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

frontend for CX-server is also planned, its implementation will enable both-sided communication with VEPP-4 control system.

NI Integration

BINP uses NI software (LabWindows) and hardware (such as CompactRIO). When interoperation between NI products and CX was required, it was usually implemented via some simple custom protocols.

However, with CXv4 modular structure it seems possible to use CDA in LabWindows and to mate a slightly modified version of CX-server with CompactRIO internals (as shown on Fig. 4d). These projects are currently under development.

CONCLUSION

CX version 4 employs a modular approach with client-server communication separated from both client and server cores and implemented in a plugin fashion.

This model proved to be flexible enabling direct communication with other control system frameworks. Additional flexibility makes CXv4 more suitable for control systems of various scales from small test stands to large facilities.

REFERENCES

[1] A.A.Starostenko et al., "Status of Injection Complex VEPP-5: machine commissioning and first experience of positron storage", IPAC2014, Dresden, Germany, MOPME073 https://inspirehep.net/record/1314333/files/mopme073.pdf

[2] D.A.Starostenko et al., "Results of operating LIA-2 in radiograph mode", Physics of Particles and Nuclei Letters, September 2014, Volume 11, Issue 5, pp 660-664.

[3] D.Bolkhovityanov et al., "Design and Development of a Control System for Intense Source of Radioactive Ions prototype", ICALEPCS'2005, 10-14 November, 2005, Geneva, Switzerland, P1-091.

[4] Yu.I.Semenov et al., "60 KEV 30 KW electron beam facility for electron beam technology", EPAC'08, June 23-27, 2008, Genoa, Italy, TUPP161.

[5] V.E.Blinov, et al., "The status of VEPP-4", Physics of Particles and Nuclei Letters, 2014, Vol.11, No.5, pp. 620-631.

[6] "VEPP-2000 Project", http://vepp2k.inp.nsk.su/

[7] A.Bogomyagkov, et al., "Automation of operations on the VEPP-4 Control System", ICALEPCS-05, 10-14 November, 2005, Geneva, Switzerland.

[8] A.Senchenko et al., "VEPP-2000 Collider Control System", PCaPAC-2012, Kolkata, India, FRCB04 (2012) http://epaper.kek.jp/pcapac2012/papers/frcb04.pdf

[9] D.Yu.Bolkhovityanov et al, "Present Status of VEPP-5 Control System", ICALEPCS2007, October 2007, Oak Ridge, USA, TPPB18.

[10] "CXscheduler: a simple scheduler for event-driven console applications" http://cxscheduler.sourceforge.net/

[11] D.Yu.Bolkhovityanov et al, "PowerPC-based CAMAC and CAN-bus Controllers in VEPP-5 Control System", PCaPAC2005, 22-25 March, 2005, Hayama, Japan, WEB4 http://conference.kek.jp/PCaPAC2005/paper/WEB4.pdf

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A MODULAR APPROACH TO DEVELOP STANDARDIZED HVAC CONTROL SYSTEMS WITH UNICOS CPC FRAMEWORK

W. Booth, B. Bradu, E. Blanco, M. Quilichini, M. Bes, M. Zimny, R. Barillere,
CERN, Geneva, Switzerland

Abstract

At CERN there are currently 200 ventilation air handling units in production, used in many different applications, including building ventilation, pressurization of safe rooms, smoke extraction, pulsion/extraction of experimental areas (tunnel, cavern, etc.), and the ventilation of the computing centre. The PLC applications which operate these installations are currently being revamped to a new framework (UNICOS CPC [1]). This work began 3 years ago, and we are now in a position to standardize the development of these HVAC applications, in order to reduce the cost of initial development (including specification and coding), testing, and long-term maintenance of the code. In this paper we will discuss the various improvements to the process, and show examples, which can thus help the community develop HVAC applications. Improvements include templates for the "Functional Analysis" specification document, standardized HVAC devices and templates for the PLC control logic, and automatically generated test documentation, to help during the Factory Acceptance Test (FAT) and Site Acceptance Test (SAT) processes.

OVERVIEW

The development of an HVAC (Heating, Ventilation and Air Conditioning) control system follows a standard development cycle including data gathering, requirements specification, application development, implementation, and testing, as outlined below.

DATA GATHERING

The data gathering process collects information on the hardware that is, or will be, on-site. The process control overview is shown in the Process & Instrumentation Diagram (P&ID), see Figure 1. The electrical wiring diagram provides the information on the specific signals available in the PLC. Early on in the migration of cooling and ventilation applications to the UNICOS CPC (Unified Industrial Control System: Continuous Process Control) framework, a standard input/output (IO) list format was adopted, which provides all the pertinent information regarding the IO (e.g. unit name, unit number, actuator, position, hardware type, address, etc), based on the electrical wiring diagram, in a convenient form (Excel spreadsheet), see Figure 2. These two documents form the initial inputs for the control system design and requirements specification.

.

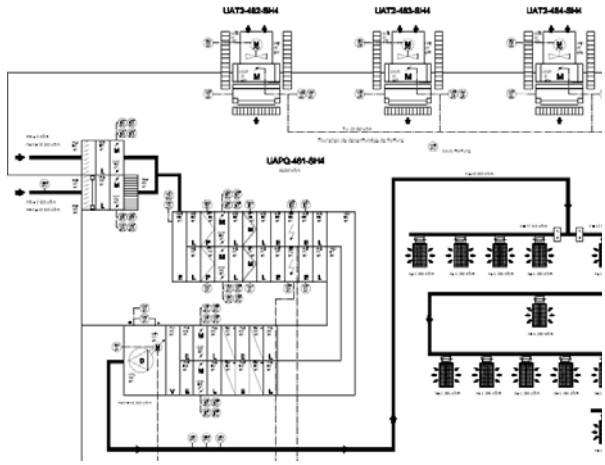


Figure 1: Example Process and Instrumentation Diagram (P&ID)

DeviceIdentification	Unit Name	UAPQ
	Unit Number	481_1
	Actuator	UMRM
	Position	B04
	Generic Name	UBLP
	Special Name	
DeviceDocumentation	Description	DAMPER POSITION EXTERNAL AIR 1 UVUM_B04_481.1
	GMAO	
	Remarks	Attention [2 10] V
FEDeviceIOConfig	Hard. Type	[0 10] V
	Address	PIW128
FEDeviceParameters	Range Min	0
	Range Max	100
	Unit	%

Figure 2: Example IO list format

REQUIREMENTS SPECIFICATION

The requirements of the control system are specified in the functional analysis (FA) document, which defines the functional breakdown of the control system into a hierarchy of individual units as specified in IEC 61512-1 or ANSI/ISA S88 [2]. After 3 years development of Ventilation plants, many different variations were found. Therefore, an analysis of the various ventilation control systems revealed that they are based on two key features: the Stepper (i.e. the state diagram) governing the operation of the individual unit, see Table 1, and the type of temperature regulation, see Table 2.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Overview of AHU Steppers

The result of the analysis of the variation of steppers among the existing applications is shown in Table 1. This demonstrates that 2/3 of the applications can be classified as type 1 through 4. As a result of this analysis, the standard requirements for type 1 through 4 were captured, as a template, that the designer can then use to create FAs in the future.

Table 1: AHU Stepper Type Summary

AHU stepper	Description	#
Type0	No stepper	1
Type1	Start-Up phase	26
Type2	Start-Up+Post-Ventilation	8
Type3	Cold-Start-Up+Post-Ventilation	10
Type4	Post-Ventilation	2
Specific	x	21

The template specification includes the state diagram, or stepper, an example of which is shown in Figure 3, as well as the definition of the operating states, and the table of actuator behaviour, as well as standard interlocks for the unit and the equipment.

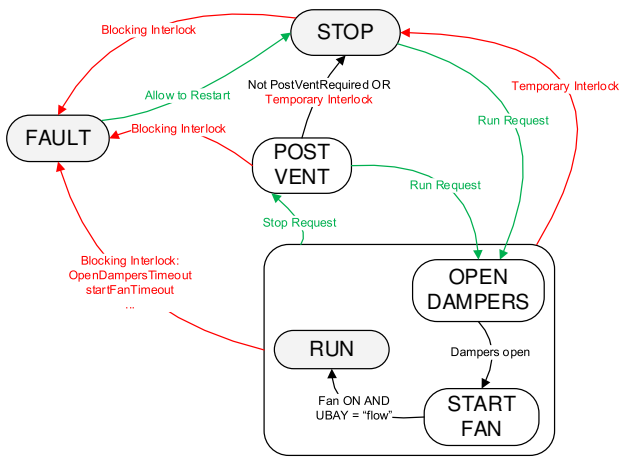


Figure 3 : State diagram for 'Type 2' AHU

Overview of Types of Temperature Regulation

The second key feature of a ventilation control system is the type of temperature regulation. After analysing the applications which have been deployed, it was found that 2/3 can be classified into a few different types, thus allowing us to standardize the implementation.

Table 2: Temperature Regulation Summary

Regulation	Description	#
Type0	No Temperature regulation	7
Type1	1 Cascade for Ambient+Supply with split range with set point filtering on ambient	19
Type2	1 Single Loop for Supply	8
Type3	Type 1 + De-humidification on cooling valve	14
Type4	3 cascades in parallel for supply	1
Type5	3 single loops in parallel for supply	2
Specific	x	17

IMPLEMENTATION

Once the requirements for the ventilation control system have been clearly specified in the Functional Analysis document, the development phase starts. Normally this phase of the project can take up to 4 weeks, depending on the complexity of the project.

The first phase is building the UNICOS CPC specification (spec) defining all the objects in the control system, beginning with the Ventilation Spec Tool as described in the next section.

The next phase is creating the specific logic for the various equipment, per the requirements.

Finally, the PLC program is tested in various phases, initially in simulation on a real PLC in the lab, Factory Acceptance Testing (FAT) and finally on site on the real PLC, Site Acceptance Testing.

Spec Creation with Ventilation Spec Tool

Based on the IO list mentioned above, a tool called the Ventilation Spec Tool has been developed. It takes an IO list and creates an initial version of the UNICOS CPC specification (spec) which defines all the devices in the control system. Note, this is very similar to the Cooling Spec Tool, see [3], except it accounts for the unique naming convention employed in Ventilation systems.

Completing the UNICOS CPC Spec

Once the user has an initial version of the specification, the control engineer completes it by hand, with the control devices needed for the project. These comprise interlocks, regulation loops, operator parameters and calculations, as specified in the Functional Analysis. This is one of the most tedious parts of the control development. There are a few possible ways of automating this, none of which are entirely flawless. The interlocks/alarms could be generated directly from the Functional Analysis, but this is not perfect because of the lack of formal language and signal names which do not match the IO list. Alternatively, a predefined

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

list of alarms for a given equipment could be an alternative, but this tends to be incomplete because the alarms at the unit level (i.e. the master of a group of actuators) cannot be easily generalized.

Once the user has a completed version of his specification, he can then generate all the instances and standard logic for the PLC and create a preliminary version of the control code.

Logic Development Using User Templates

The next step is to create specific logic for the individual units and actuators, based on the requirements. In the baseline UNICOS CPC framework, all 'user defined logic' is, as the name implies, left up to the user to define, either in the IDE (Integrated Development Environment) provided by the PLC supplier (either Unity for Schneider PLCs or Step-7 for Siemens PLCS), or by means of 'templates', which allow more advanced programming capabilities of the PLC logic by means of the use of scripting languages such as Python. However, these templates are completely open for the user to create. In the case of the cooling and ventilation applications, this open approach, resulted in each programmer (there have been up to 10 individuals programming cooling and ventilation PLC applications over the last 3 years) adopting his own approach and the code was somewhat heterogeneous, and more difficult to maintain as a result.

Therefore, more recently, new templates have been adopted for each given type of UNICOS CPC field object. The advantages of these new user templates is shown in Figure 4. They include standard implementations of the various features needed to program ventilation units. For example, work time counters, and "anti restart" protection, which ensures that an equipment cannot be turned back on immediately after being switch off, to avoid overheating. The programmer just has to enable a feature by adding a keyword to his specification file, and in some cases providing a few additional UNICOS objects for storing counters, or temporary variables. This avoids unnecessary work and programming errors; once you have defined a given function, it can easily be re-used elsewhere.

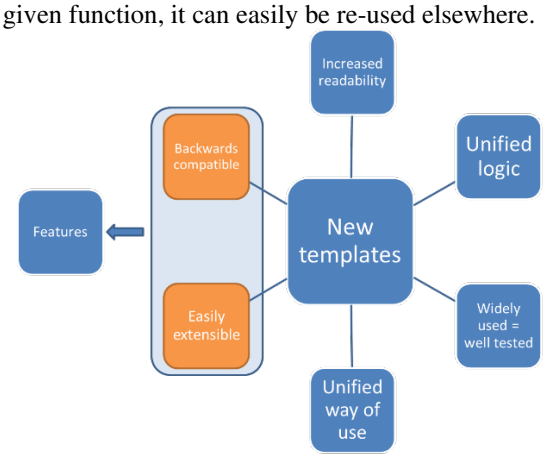


Figure 4: Advantages of new user templates

In addition, the templates make it simple to specify the operation of equipment as a function of the stepper state. As explained above, a significant piece of the functional design, is the state machine for a given unit. Each state has a name, and using the HVAC templates, it is easy to specify in which state a valve should be open or closed, or in which state a fan should be regulating, or at a fixed speed. This connects the programming with the functional specification, and therefore simplifies the overall complexity of the system.

In the future generating the initial logic directly from the Functional Analysis could be possible, once a formal way for designers to specify their requirements is available.

SCADA Synoptics

The final step before testing is the development of SCADA synoptic views, using Siemens WinCC OA, the standard solution adopted at CERN. The synoptics are developed based on the P&ID following internal standards, with additional input from the operators where necessary. The process of generating synoptics from the P&ID could be automated now that the P&IDs are being developed in more modern tools.

In the majority of cases, there is a requirement to develop a local operation facility based on industrial touch panels. The constraint is 100% availability, as the user needs to operate the plant on-site even in the case that network access is not available. This represents significant additional effort and it would be preferable to generate these touch panel views directly from the WinCC OA synoptics, but this is currently not possible.

TESTING

Once the PLC control system logic is implemented and the SCADA synoptics created, the test phase starts. First in a simulation in a lab PLC, a static simulation is built, using a custom-made PLC function to simulate the IO of the actual hardware. This involves modifying the PLC code slightly in order to wrap-around the actuator outputs to the actuator inputs, with some simple dynamics. Also normally-closed fault inputs are artificially forced, and delays are added to other inputs (pressure and flow switches, etc.) as a crude approximation of the dynamics of the installation.

Once a working simulation is built and the code is loaded on a PLC in the lab (this is quick and can be done in less than an hour) the Factory Acceptance Testing using the FAT commissioning file as outlined below is started.

FAT / SAT Commissioning File

During the design, development and test process, the amount of time testing and that of implementing the requirements are approximately the same. Therefore, it is imperative that an exhaustive test document is established, which covers all the requirements. Thus a commissioning file for use during Factory Acceptance Testing (FAT) in the lab and Site Acceptance Testing (SAT) on-site is generated. This file gives an overview of the application, including all inputs/outputs (IO), all actuators (including the parameterization, if any), all parameters which can be

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

modified during plant operation, a list of all the alarms which are sent to the CERN control centre (CCC), as these will be visible to the operators who monitor these applications 24/7, and a list of all the regulation loops.

Also, broken down by individual unit of operation (for example, an extraction unit), this file includes the list of alarms, see Figure 5, the stepper states, the various actuators, the parameters, and the commands, and where possible a comparison of the AF requirement against the actual code implementation (a traceability matrix). Thus the tester (who in most cases is someone other than PLC programmer) can easily navigate the various pieces of the program, in order to test, validate, and ensure compliance with the high-level functional requirements.

Name	UAPQ_481_AL6
Description	PROBLEME TEMPS D'OUVERTURE REGISTRES
FA Condition	Open Dampers Delay 180 s
Code Condition (for reference only)	DB_GRAPH_UAPQ.OPEN_DAMPERS_AR.X OR DB_GRAPH_UAPQ.OPEN_DAMPERS_AN.X
Type	FS
Master	UAPQ_481
Folio	4.10
Threshold	
Delay	UAPQ_481_AL6Dt
CCC Alarm	UAPQ_481_CCC_MAJ
Alarm activation	OK
Alarm Action	OK
Date	22/05/2015
Responsible	
Remarks	

Figure 5: Example of an alarm for a particular unit, with requirements, completed during FAT testing

In addition, the stepper is extracted from the code and displayed visually, in an automated way, in order to validate the implementation against the requirement. This was previously not possible without manually opening each stepper in the suppliers' IDE.

In fact, this commissioning file can be used for any UNICOS-CPC PLC program, and is delivered with the UNICOS-CPC resource package, but it has been developed in close collaboration with the Cooling and Ventilation group, based on their significant testing expertise developed during the migration of their old PLC applications to the UNICOS-CPC framework.

HVAC FOR LHC SURFACE BUILDINGS

For the renovation of the LHC surface building ventilation systems, the control systems of 20 surface ventilation buildings will be re-engineered, beginning with those which contain the surface cryogenics installations.

In order to expedite the control system development, a generic control system and applicable set of templates were developed. These are used to generate the 7 similar installations by replacing generic names with the specific installed equipment codes. This approach ensures that all the control systems are alike, and significantly reduces the time it takes to develop each application. This approach works well for applications which are identical. However, the remaining 13 surface installations, which will be completed in 2016, are slightly different and thus the current approach will have to be adapted. In general, over the past 3 years we have commissioned ~60 ventilation applications, and during the next 5 years, we expect to commission a further ~100, and hopefully these improvements will significantly reduce the development effort.

CONCLUSION

A noticeable improvement of the process of developing HVAC control systems with the UNICOS CPC framework has been done. This comprises improved templates for creating the requirements (Functional Analysis), implementing the PLC code (using user templates), and testing (using the commissioning file). However, there is still room for improvement such as enhancing the process to automatically generate more code, directly from the requirements if possible, and generate the SCADA synoptics directly from the P&IDs. Also the IO list could be generated automatically from the wiring diagram itself, thus simplifying the design process yet further. With these additional improvements, coding errors would be drastically reduced, and the effort required to develop the control system minimized.

ACKNOWLEDGMENT

We would like to thank the CERN Cooling and Ventilation (EN-CV) group and specially the operation and control teams for working closely with us over the last few years and sharing their control system expertise, in order to improve our development process.

REFERENCES

[1] B. Fernandez et al., UNICOS-CPC6: Automated code generation for process control applications" ICALEPCS'11, Grenoble, October 2011.

[2] IEC 61512-1 or ANSI/ISA S88. Batch Control - Part 1: Models and terminology. 1995.

[3] B. Bradu et al., "Re-engineering Controls Systems Using Automatic Generation Tools and Process Simulation: The LHC Water Cooling Case", THPPC076, ICALEPCS'13, San Francisco, USA (2013)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

APPLICATION OF PyCDB FOR K-500 BEAM TRANSFER LINE*

Pavel Cheblakov#, Sergey Karnaev,
Oidin Khudayberdieva, BINP SB RAS, Novosibirsk, Russia

Abstract

The new injection complex for VEPP-4 and VEPP-2000 e-p colliders is under construction at Budker Institute, Novosibirsk, Russia. The double-direction bipolar transfer line K-500 of 130 and 220 meters length respectively will provide the beam transportation from the injection complex to the colliders with a frequency of 1 Hz. The designed number of particles in the transferred beam is 2*10^10 of electrons or positrons, the energy is 500 MeV. K-500 has dozens of types of magnets, power supplies and electronic devices. It is rather complicated task to store and manage information about such a number of types and instances of entities, especially to handle relations between them. This knowledge is critical for configuration of all aspects of control system. Therefore we have chosen PyCDB to handle this information and automate configuration data extraction for different purposes starting with reports and diagrams and ending with high-level applications and EPICS IOCs' configuration. This paper considers concepts of this approach and shows the PyCDB database structure designed for K-500 transfer line. An automatic configuration of IOCs is described as integration with EPICS.

INTRODUCTION

Every large scientific facility is a complicated construction with a huge amount of hardware, electronic equipment and software. Hardware is controlled by built-in electronics in some cases, it requiring dedicated or universal standalone controllers in other cases. Anyway it results in using of a wide spectrum of electronics of different types from different vendors. Moreover it can rapidly increase the amount of physical protocols types (field busses), interconnections, network equipment, CPU and OS platforms and etc. Consequently, software should provide support for diversity of electronics and thereby hide its complication from user.

The largest facilities (e.g. ITER [1]) include hundreds of local systems and dozens of subsystems. These local systems consist of a set of controllers, interfacing the actuators and sensors, and connected together via network switches to the plant operation network. The total amount of computers can reach one thousand, the total number of signals (wires) and the total number of process variables is estimated at 100.000 and 1.000.000 accordingly.

* This work has been supported by Russian Science Foundation (project N 14-50-00080).
# P.B.Cheblakov@inp.nsk.su

PROBLEMS

To maintain so large and diverse system it is necessary to collect and store all relevant information: not only lists of systems, subsystems, network equipment, controllers, instrumentation and control devices but relationships between them as well. We need to know which controller is responsible for corresponding power supply or specialized gauge. We need to know how that controller is connected to corresponding computer and which switch is used. Also we need to know what types of cables are used for interconnection.

Strictly speaking we can continue this list further and further adding more and more information about our facility. This information is vital for broad range of specialists: from storekeepers, designers and electricians to software developers, operators and physicists. Having this data we can create a list of equipment of particular type for record keeping purposes. Or we can simply generate actual cable schedule. Furthermore we can get configuration data for software involved in control.

This information is mostly distributed along many spreadsheets and databases which are weakly connected. This fact is likely to cause data duplication and denormalization so we can get data inconsistency induced by human factor. As a result it is very difficult to change and analyse distributed data both for human and computer.

There are different approaches to store and handle this information. First of all we divide the information into two parts: the first one is used for software configuration and the second one includes everything else. Handling the former part of data is more formalized and there are some practices to operate with it. For example the paper [2] distinguishes centralized and decentralized approaches covering almost all cases and proposes hybrid technique. Even though it turns to a laborious work for every facility [3, 4, 5].

As regards the latter part of the data it is so unstructured and non-formalized that it becomes a great challenge to build up a data-storing system from scratch or even deploy any specialized software system [6].

KNOWLEDGE BASE

To deal with heterogeneity and complexity of data we propose to use a centralized storage which is able to store and handle with ease not only data and its properties but relationships as well. Storing and retrieving relationships is a key aspect for successful functioning of many systems: from equipment procurement and registering to cable scheduling, from GIS-data to software configuration

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

system. Let us use a name *knowledge base* for our storage instead of extensively used term *database* to distinguish the former from classic relational database [7]. We need to store not just tables with numbers and strings but pointers to other objects which in turn have additional pointers.

Therefore our knowledge base represents facts about scientific facility in a structured form that suitable for software processing and analysis. An object model often called *ontology* can be used for representation of knowledge base [8]. Ontology is a formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular domain.

GRAPH DATABASE

Closely connected conception to knowledge base is a *semantic network*. It is a network which represents semantic relations between concepts. It is a directed or undirected graph consisting of vertices, which represent concepts, and edges [9]. Applying ideas of semantic network to our problem domain - scientific facility - we can create *facility informational model.* Since semantic network is a graph we assume that using a graph database is a good choice.

Graph databases stay far away from relational databases and they differ markedly from most NOSQL databases. They have some drawbacks which are mostly not relevant to our problem. But advantages make graph databases a preferable solution. One compelling reason for choosing a graph database is the sheer performance increase when dealing with connected data. Graphs allow structure and schema to emerge in tandem with our growing understanding of the problem space, rather than being imposed upfront, when we know least about the real shape and intricacies of the data. Graphs are naturally additive, meaning we can add new kinds of relationships, new nodes, new labels, and new subgraphs to an existing structure without disturbing existing queries and application functionality [10].

K-500 TRANSFER LINE

K-500 is a transfer line for electrons and positrons which is under construction in Budker Institute of Nuclear Physics, Novosibirsk, Russia. The source of particles is Injection Complex (VEPP-5) and it will supply both BINP colliders: VEPP-4 and VEPP-2000. Right now we are working on the part from Injection Complex to VEPP-4 and it contains 5 DC power supplies and 15 pulsed power supplies of 5 types in all. 24 electronic devices of 8 types are used to control power supplies and measure their parameters.

PyCDB [2] was chosen to operate with graph data. This is a specially designed graph database engine to operate over facility informational model. The data model remains almost the same as developed for the booster of the NSLS-II [11] while we tried to make it more flexible (see Fig. 2).

The special tool was developed for editing and viewing graphs. It is a graphical web-application which allows to observe and manipulate with data in a graph stored in PyCDB. It is shown on the Fig. 1. The main window has three parts. The left side contains a data model for corresponding facility, the right side (largest one) is a graph itself and the upper part allows us to edit properties and “look and feel” of selected entity or relationship.

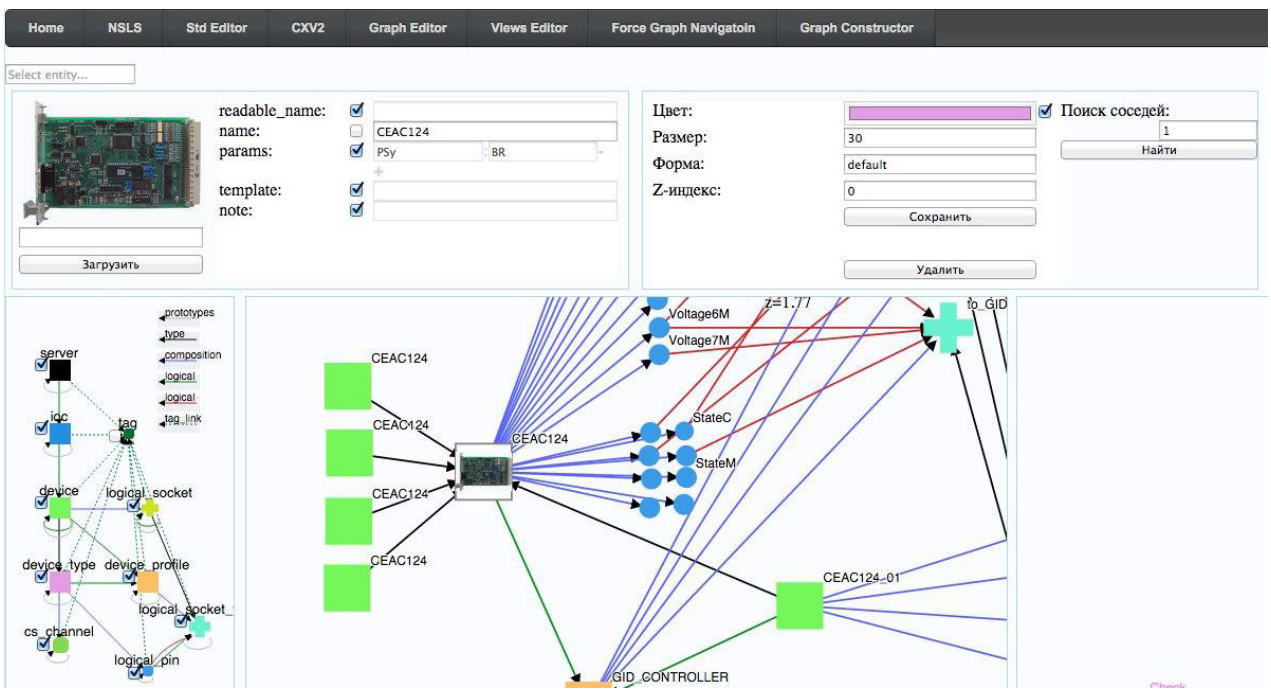


Figure 1: The Graph Editor’s main window.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

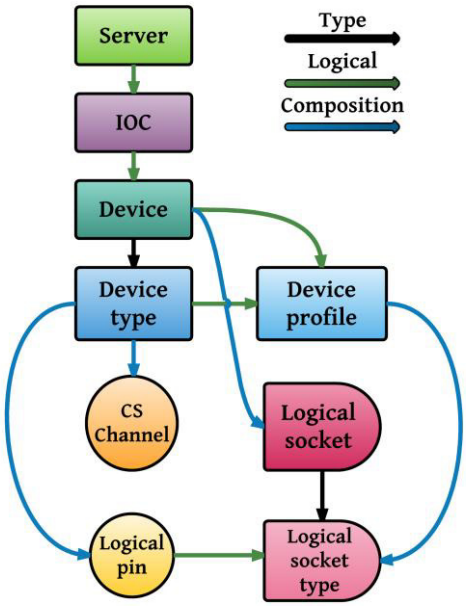


Figure 2: The conceptual data model for K-500's Power Supply System.

Control Systems Integration

One of the obvious and desired possibilities for applying facility informational model is to automatically configure EPICS IOC Database [12]. Special Python applications are able to fetch data from the database and generate IOC's start-up scripts. These scripts are loaded into IOC and define a part of distributed EPICS database.

CONCLUSION

Using technologies initially developed in a scope of artificial intelligence, particularly for expert systems and knowledge systems, may meet requirements of modern scientific facilities. Complexity and scale of recent ones makes handling all relevant information to be a quite complicated task. Appearing and fast growing of modern technologies like graph databases gives a new life for old ideas. Applying a graph database for creation and handling facility information model for K-500 transfer line is in progress. A high necessity of such model is shown, a graphical editor for a model was developed and made some steps towards control system integration.

REFERENCES

[1] A.Wallander et al., "News From ITER Controls – a Status Report", proceedings of ICALEPCS2011, Grenoble, France (2011).
[2] A.Makeev et al., "Centralized Software and Hardware Configuration Tool for Large and Small Experimental Physics Facilities", proceedings of ICALEPCS2013, San Francisco, CA, USA (2013).
[3] R. Bakker et al., "Significance of a Comprehensive Relational Database System for Modern Accelerator Controls", ICALEPCS97, Beijing, October 1997, p. 239.
[4] Z. Zaharieva et al., "Database Foundation for the Configuration Management of the CERN Accelerator Controls Systems", proceedings of ICALEPCS2011, Grenoble, France (2011).
[5] D. A. Dohan et al., "Integrated Relational Modeling of Software, Hardware, and Cable Databases at the APS", proceedings of ICALEPCS2003, Gyeongju, Korea (2003).
[6] D. Stepanov et al., "Integrated Approach to the Development of the ITER Control System Configuration Data", proceedings of ICALEPCS2011, Grenoble, France (2011).
[7] Krishna, S. Introduction to Database and Knowledge-base Systems. Singapore: World Scientific Publishing, 1992. ISBN 981-02-0619-4.
[8] Oberle, D., Guarino, N., & Staab, S. "What is an ontology?". In: "Handbook on Ontologies". Springer, 2nd edition, 2009.
[9] John F. Sowa (1987). "Semantic Networks". In Stuart C Shapiro. Encyclopedia of Artificial Intelligence.
[10] Ian Robinson, Jim Webber, and Emil Eifrem, "Graph Databases". 2nd edition. 2015.
[11] P.Cheblakov et al., "Configuration System of the NSLS-II Booster Control System Electronics", proceedings of ICALEPCS2013, San Francisco, CA, USA (2013).
[12] R.Keitel, "Generating EPICS IOC Data Bases from a Relational Data Base – A Different Approach", proceedings of ICALEPCS'01, San Jose, CA, USA (2001).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MANAGING A REAL-TIME EMBEDDED LINUX PLATFORM WITH BUILDROOT

J. Diamond#, K. Martin, FNAL*, Batavia, IL 60510, U.S.A.

Abstract

Developers of real-time embedded software often need to build the operating system, kernel, tools and supporting applications from source to work with the differences in their hardware configuration. The first attempts to introduce Linux-based real-time embedded systems into the Fermilab accelerator controls system used this approach but it was found to be time-consuming, difficult to maintain and difficult to adapt to different hardware configurations. Buildroot is an open source build system with a menu-driven configuration tool (similar to the Linux kernel build system) that automates this process. A customized Buildroot [1] system has been developed for use in the Fermilab accelerator controls system that includes several hardware configuration profiles (including Intel, ARM and PowerPC) and packages for Fermilab support software. A bootable image file is produced containing the Linux kernel, shell and supporting software suite that varies from 3 to 20 megabytes large – ideal for network booting. The result is a platform that is easier to maintain and deploy in diverse hardware configurations.

INTRODUCTION

Scientific Linux [2] is a Linux distribution produced by a collaborate effort between Fermilab, CERN and other members of the global Linux community. Scientific Linux has been used successfully as a target platform for data acquisition in the Fermilab accelerator control system but only on commodity PC hardware and not with real-time extensions [3]. When attempting to adopt Scientific Linux to a real-time embedded application some road-blocks were encountered. The first was that Linux is not designed to be a hard real-time operating system. There are however, several options for making Linux real-time and all of these options require a custom kernel be built. At Fermilab, one route chosen was to use the Real-Time Application Interface (RTAI) [4]. The next road-block was the minimum foot-print of the installed system. The minimum Scientific Linux install profile requires 1.5GB of storage. This is a concern because it is desirable to network boot the targets. The requirement for a network boot capability drastically reduces the acceptable footprint size as the entire boot image needs to be transferred over the network. Finally, Scientific Linux is a binary distribution targeted for x86 architecture, thus it would not be possible to use as a platform for ARM and PowerPC targets. These concerns drove the decision to use a custom Linux platform built from the ground-up.

* Operated by Fermi Research Alliance, LLC under Contract No. De-AC02-07CH11359 with the United States Department of Energy

jdiamond@fnal.gov

LINUX "FROM SCRATCH"

The Linux from Scratch website [5] was chosen as the model for the attempt to build a custom Linux platform from source. Linux from Scratch is an on-line resource that explains in step-by-step fashion how to build an entire Linux operating system from source. First the user partitions the target's disk on the build system. Then the necessary source packages (about three dozen) and patches (another dozen) are downloaded onto the build system and unpacked into a special "chroot environment". Then the toolchain packages are built and installed – not a trivial task in its own right. The toolchain is then used to build a complete Linux system from inside of the chroot environment - kernel, boot-loader, shell, etc. Once the system is in place the boot-scripts are installed and configured. Finally, the boot-loader is installed on the target CF disk and an attempt is made to boot the target (things rarely work on the first try).

As illustrated above, the process is exhausting and can take several days to complete. It should be noted that the target platform was x86 so a cross-compile toolchain was not attempted (ARM targets came later). Once developed the root filesystem could be used to boot any target with a similar hardware architecture.

The root filesystems produced with this method were in the range of two to three hundred megabytes. One significant drawback is that a boot image of this size is still too large to network boot. The decision was made to store the kernel, boot-scripts and network configuration on a Compact Flash (CF) disk installed with each target node and the root filesystem on a common network filesystem that could be mounted by each target at boot time. The network filesystem also contains the application programs, libraries and data areas.

One issue that was noticed early on was the failure rate of the CF disks. CF disks are susceptible to failure after many write-cycles. As noted above, application data is written to a network filesystem instead of the CF disk. To reduce the CF write cycles the /tmp and /log partitions were moved to a RAM disk and the system log was redirected to a syslog server. These efforts were able to mitigate the failure rate of the CF disks.

Another significant challenge with this platform was maintainability. Installing new software packages in the root filesystem was easy but deploying to targets already in the field meant physically replacing the CF disks. Furthermore, the root filesystem installed on the network filesystem would creep away from its original state over time without version control system in place.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONTROL SYSTEM INTEGRATION

At first the support libraries available for integrating with the Fermilab accelerator controls system (ACNET) [6] [7] only supported VxWorks. Package Exchange Protocol (PEP), a light weight UDP protocol was developed to bridge communication between Linux targets and VxWorks targets (Fig. 1). Support software that ran on a VxWorks target was deployed to bridge communication between ACNET and Linux targets deployed in the field. This method was successful and supported many, but not all of the features expected from an ACNET front end node.

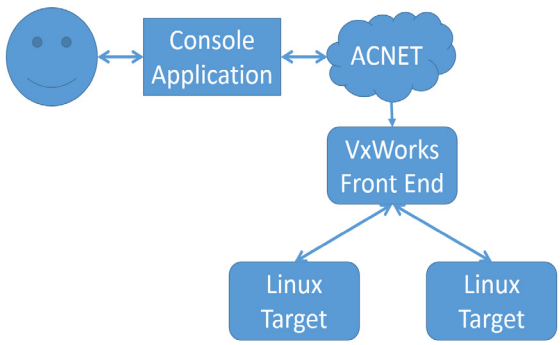


Figure 1. Method for using PEP to bridge ACNET request to Linux Targets.

Later, a suite of software was developed for interfacing Linux front end nodes with ACNET [8]. This ACNET support software is written in Erlang and runs inside of the Erlang virtual machine as its own process (Fig. 2). Using this software the user can write data acquisition drivers in Erlang or using a C++ API. The challenge in this model is how to attach the data acquisition driver, running in its own process to the real-time application running in another process. The decision was made to again utilize the PEP protocol to forward ACNET requests to the real-time application process. In this architecture the communication was done with local sockets instead of over Ethernet.

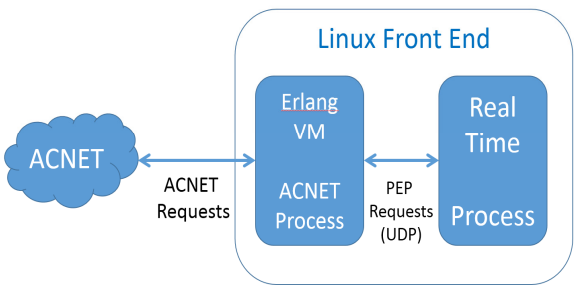


Figure 2. ACNET process utilizing PEP for inter-process communication.

Using this architecture the existing applications using the bridge method were adapted to the new Linux ACNET platform without much effort (Fig. 3). The interface remained the same and the bridge software was moved into the Erlang framework. An alternate method was developed that replaced the PEP protocol with RTAI message queues and shared memory. This method allows large amounts of data to be shared between the ACNET process and the data acquisition process without transferring through a UDP socket.

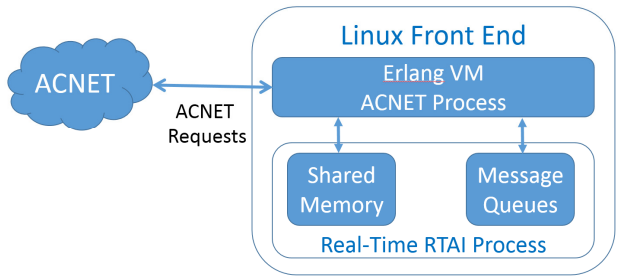


Figure 3. Replacing PEP with RTAI IPC methods.

LOOKING TO IMPROVE

Out of the experience of building a real-time embedded Linux platform from scratch came three desired improvements:

1. A smaller footprint operating system that could be packaged into a network-deliverable boot-image
2. An automated system for building kernel and root filesystem that could support multiple target architectures and hardware configurations
3. An integrated system for building and deploying application software

To address these improvements an effort was begun to look to the open source community for a “best-practice” approach to deploying an embedded Linux platform.

BUILDROOT

Buildroot is an open source build system with a menu-driven configuration tool (similar to the Linux kernel build system) that completely automates this process (Fig. 4). It supports ucLibc [9], a low-footprint alternative to the GNU standard C library and Busybox [10], which combines many of the standard UNIX utilities and a shell into a single low-footprint executable.

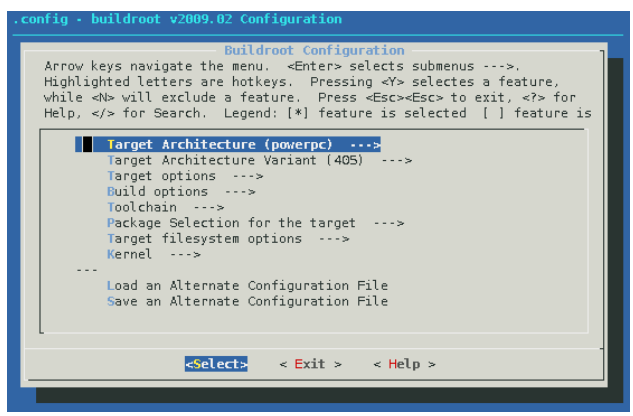


Figure 4. Buildroot configuration menu.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

In addition to providing a kernel, shell and basic utilities, hundreds of software packages are supported. The user simply selects which software package to include in the boot-image and the build system downloads, unpacks, configures, compiles and installs it. The Buildroot configuration system includes an option to combine the kernel and root filesystem into a single bzImage at the end of the build process. A boot-loader and this bzImage file are the only components needed to boot an embedded Linux target.

The first build usually takes a couple of hours, depending on the processing power and network bandwidth of the build machine. This is primarily because of the sheer size of the tool-chain and kernel builds. Once these packages are built, subsequent builds can be done in a matter of seconds.

The footprint depends in large part on the choice of software packages and system libraries. Using ucLibc, Busybox and a streamlined kernel the resulting footprint was 3.5 megabytes – a 99% reduction in size from the existing Linux platform. When the build configuration was expanded to include Erlang, ACNET support libraries and application software the footprint grew to 20 megabytes. With this footprint size, the entire kernel and filesystem could be downloaded by the boot-loader and loaded into a RAM disk on start-up. In order to store the network configuration locally, the /etc partition was all that was left to store on the CF disk. The boot-time of one target was clocked at under nine seconds from power-on to login prompt.

REVISION CONTROL

The Buildroot project is maintained in a public Git repository. A Fermilab clone of the master branch was made and is merged with the project's repository when upgrades or new packages are desired. Custom Buildroot and Kernel configurations for each target are maintained in the repository. Custom application packages are maintained in the repository as well.

APPLICATION PACKAGES

Buildroot supports hundreds of software packages out of the box as well as custom packages. A Buildroot package consists of two parts: a menu description file and a Makefile that is included as part of the build process. Buildroot uses variables defined in the Makefile to find dependencies, to find out where to download the software from and what specific configuration and build options to use.

Dozens of custom software packages were developed for applications and support libraries specific to Fermilab. Buildroot downloads the software directly from a version control server using a specific tag or branch identified in the Makefile.

NETWORK BOOT

The GRUB [11] boot-loader supports downloading the bzImage file from a network filesystem at boot time. GRUB must be built with the support for the network adapter on the target. The network configuration is stored in GRUB menu.lst file, which resides on the flash disk's boot partition.

APPLICATION BUILD ENVIRONMENT

All software that runs on the target must be built with the toolchain that Buildroot builds as the first step in the build process. A set of Makefiles was developed that could be included in a developer's C/C++ project that builds using this toolchain. Deploying to remote targets is supported using scp and ssh or by mounting a remote filesystem accessible from the target and the build machine.

ARM SUPPORT

Recently, Buildroot was utilized to deploy a real-time embedded Linux platform for the ARM Cortex A-9 Hard Processor System on the Altera Cyclone V FPGA. Cross compiling was done by specifying the pre-built Linaro ARM toolchain [12] as an external toolchain. A community kernel configuration file for the Cyclone 5 FPGA SOC, available through RocketBoards.org [13], was utilized to build the Linux kernel.

One notable difference between ARM and x86 targets is that they use different boot loaders. The x86 targets use GRUB and the ARM targets use U-Boot [14]. Like GRUB, U-Boot is a flexible boot loader that allows for network booting. Another difference due to U-Boot though is that it is unable to boot a kernel image that is integrated with a root filesystem. Thus, Buildroot was configured to output the kernel and the root filesystem as separate files. Also, ARM based Linux systems require a device tree file. Fortunately a device tree file for the Cyclone 5 FPGA SOC target was also available pre-made from RocketBoards.org.

RESULTS

Buildroot has been successfully used to deploy real-time embedded Linux systems in the Fermilab accelerator control system for several projects running on x86 and ARM hardware with applications such as magnet quench protection, power supply regulation and beam instrumentation. An effort is currently underway to extend support for legacy PowerPC targets such as the MVME 5500. We have confidence that adding support for additional targets can be done with minimal effort thanks to the efforts of the open-source embedded Linux community.With Buildroot we were able to reduce the operating system footprint and introduce version control to the platform build process. We have found that managing multiple architectures and hardware configurations much easier than building platforms from source would be.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ACKNOWLEDGEMENTS

The ACNET Erlang framework was implemented by Rich Neswold, Dennis Nicklaus, and Jerry Firebaugh from the Femilab's Controls Group.

REFERENCES

[1] http://buildroot.uclibc.org/
[2] https://www.scientificlinux.org/
[3] C. Briegel, J. Diamond, "Acsys Camera Implementation Utilizing an Erlang Framework to C++ Interface", ICALEPCS (2013); San Francisco, CA, USA.
[4] https://www.rtai.org/
[5] http://www.linuxfromscratch.org/
[6] J. Patrick, "ACNET Control System Overview," Fermilab Beams-doc-1762-v1, http://beamdocs.fnal.gov/AD-public/DocDB/ShowDocument?docid=176
[7] K. Cahill, L. Carmichael, D. Finstrom, B. Hendricks, S. Lackey, R. Neswold, J. Patrick, A. Petrov, C. Schumann, J. Smedinghoff, "Fermilab Control System," Fermilab Beams-doc-3260-v3, http://beamdocs.fnal.gov/AD-public/DocDB/ShowDocument?docid=326
[8] D. Nicklaus, "An Erlang-based Front End Framework for Accelerator Controls," ICALEPCS (2011); Grenoble, France.
[9] http://uclibc.org/
[10] http://www.busybox.net/
[11] https://www.gnu.org/software/grub/
[12] https://www.linaro.org/
[13] http://www.rocketboards.org/
[14] http://www.denx.de/wiki/U-Boot

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

LOCAL MONITORING AND CONTROL SYSTEM FOR THE SKA TELESCOPE MANAGER: A KNOWLEDGE-BASED SYSTEM APPROACH FOR ISSUES IDENTIFICATION WITHIN A LOGGING SERVICE*

M. Di Carlo, INAF Osservatorio Astronomico di Teramo, Italy
M. Dolci, INAF Osservatorio Astronomico di Teramo, Italy
G. M. Le Roux, SKA South Africa, Cape Town
R. Smareglia, INAF Osservatorio Astronomico di Trieste, Italy
P. S. Swart, SKA South Africa, Cape Town

Abstract

The SKA Telescope Manager (SKA.TM) is a distributed software application aimed to control the operation of thousands of radio telescopes, antennas and auxiliary systems (e.g. infrastructures, signal processors, ...) which will compose the Square Kilometre Array, the world's largest radio astronomy facility currently under development. SKA.TM, as an "element" of the SKA, is composed in turn by a set of sub-elements whose tight coordination is ensured by a specific sub-element called "Local Monitoring and Control" (TM.LMC).

TM.LMC is mainly focussed on the life cycle management of TM, the acquisition of every network-related information useful to understand how TM is performing and the logging library for both online and offline sub-elements. Given the high complexity of the system, identifying the origin of an issue, as soon as a problem occurs, appears to be a hard task. To allow a prompt diagnostics analysis by engineers, operators and software developers, a Knowledge-Based System (KBS) approach is proposed and described for the logging service.

INTRODUCTION

A log message is the simplest possible storage abstraction which says what happened and when. It is an append-only, totally-ordered sequence of records timestamped. So, a log is not all that different from a file or a table. A file is an array of bytes, a table is an array of records, and a log is really just a kind of table or file where the records are sorted by time.

Since it is a very simple concept, developers tend to underestimate the logging system but logs record *what happened and when* and, for distributed data systems, this can be the only way to find out the origin of an error.

Usually log files are written in a natural language (human readable) and, even if it is very common, this is not the best way to store informations: it does not allow to reason programmatically** about those information.

Building a logging system with a declarative language (for instance prolog) can give the possibility to reason about facts of the communications and operations with an inference engine. For the specific case this document is aimed to, with the adoption of a knowledge-based system approach, TM.LMC could give to SKA engineers, operators and software developers the possibility to ask high level questions to the system in order to understand how a failure came up or simply understand how the system is working.

Information to Log

In order to understand which are the informations to log it is important to make some consideration. In a distributed environment:

- the entities which make up an application are active (processes or agents);
- the interactions are based on message exchange mechanism;
- the process life time is connected to the application life time; the life time of an agent are usually independent from life time of a specific application;
- the logical architecture can be set by different patterns: client-server, peer to peer, etc.
- the middleware realize the physical and logical connection between entities (subsystem, service, object, component, process, agent, etc.).

From an high level point of view the kind of applications like SKA TM define a logical network of interactions which it is composed by different nodes that interact each other and some of them act as coordinator or controller (at least for the online part of the system). So it is important to log:

- *Node*† *identification,*
- *Node signal*‡ *declaration,*
- *Node interactions,*
- *Actions and loops.*

* Work supported by the Italian Ministry of University and Research (MIUR)

† In a network, a node is a connection point, either a redistribution point or an end point for data transmissions. In general, a node has programmed or engineered capability to recognize and process or forward transmissions to other nodes.

‡ An information usable by a node in order to control the behaviour of another one or its behaviour in function with the one from another node.

** Doing something programmatically means that you can do it using source code, rather than via direct user interaction or a macro

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

KNOWLEDGE-BASED SYSTEM APPROACH

A knowledge-based system is a computer program that reasons and uses a knowledge base to solve complex problems. It is composed by two types of sub-systems: a knowledge base (facts[§] about the world) and an inference engine (logical assertions and conditions about the world). To build it, it is possible to use a formal language like Prolog.

A *fact* must start with a *predicate* (which is an *atom*[**]) and end with a *fullstop*. The predicate may be followed by *one or more arguments (*separated by commas) which are enclosed by *parentheses*. The arguments can be atoms (in this case, these atoms are treated as constants), numbers, variables or lists. The formalism can be summarized in this way: *predicate(arg1, arg2, ..., argN).*

Besides, the prolog language can be seen as a language for database queries; in fact, in a relational database a *tuple* is a generic element of a relation with attributes.

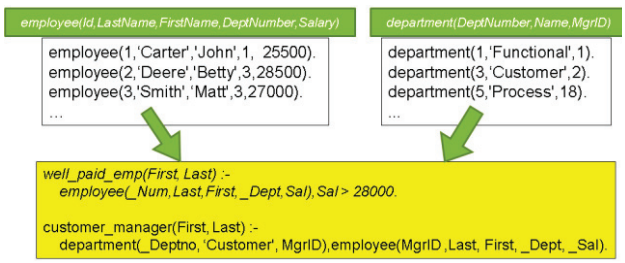


Figure 1: Prolog as database.

In Figure 1, it is shown how it is possible to represent two tables with the prolog language (the table *employee* and the table *department*) and two possible queries: the first one which give the names of all employees earning more than 28000 and the second one which give the name of the manager of the department Customer.

Interface

In order to allow software developers to use such a formalism without actually knowing it, a possibility is to look at the symbolic representation and try to generalize it following some general rule in order to finally get a *prolog theory*.

Based on the definition of a fact, a very simple way to log programmatically informations can be the following one:

logger.log(level, keyword-predicate, arguments);

where *level* is an enumerator, the *keyword-predicate* is a string and the *arguments* are an array of strings. This method can translate the informations into a Prolog fact, avoiding the developers to study and use the Prolog language. The only thing they have to do is indeed to choose a list of words representing the informations needed to store. So it is important to have a strong analysis based on UML or other modelling language. This is because the words used to log must have a sense and must not be chosen casually. Anyway, this is only a method to store informations and not a mechanism to query them. There is still a need to learn the Prolog language in order to gain full advantage from the use of the declarative approach (for simple query it is possible to make a generic tool, for example for level, date). Besides it will be always possible to search for a string directly in the text file just opening it with a notepad tool.

Compared to an old logging system (using for example the natural language), the proposed one aims to formalize every phrase in a sort of database (a Prolog database) where tables are dynamically built on the predicates chosen. Let's take the following log line:

Attaching appender named [CONSOLE-REQUEST] to Logger[TangoClientRequests]

The Prolog translation could be the following:

attaching(appender('CONSOLE-REQUEST'), logger('TangoClientRequests')).

Working in this way will give us the advantage to easily search for 'appender' or 'logger' (because we defined it as predicates) or directly searching for the predicate 'attaching'.

TANGO EXAMPLE

Tango[1] has been chosen by SKA Organization as a common middleware for communication and development. In this work it is used the same framework to show a proof of concept concerning a declarative approach based on the Prolog language.

A Counter Device

A counter device is a software that implements the following tango commands:

- *PlusOne*: increase the counter by one;
- *MinusOne*: decrease the counter by one;
- *Read*: read the value of the counter;
- *SetMaxValue*: set the max value for the counter (the counter will start from 0 and will never reach the maximum value but only the max – 1);
- *SetCounterConsumer*: set the name of the device which will use it (for logging purpose);
  Reset: set the counter value to 0.

The device will throw an event every time its value changes (a Tango change event) and every time the value is reset (when the value is equal to zero a tango user event

§ A fact is a truth about the real world that can be represented as a symbol, in order to be easily manipulated by programs.

** An atom, in Prolog, means a single data item like a string or a symbol like likes, john, and pizza, in likes(john, pizza).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

is thrown). Based on the analysis on the informations to log, a possible representation can be the one shown in Table 1.

Table 1: Log Information Expected from the Counter

Information	Prolog formalism	Counter example
Node identification	entity(entity-name, entity-type).	entity(counter, 'LRU').††
Node signal declaration	declares(entity-name, signal).	declares(counter, '+') declares(counter, '-') declares(counter, reset). declares(counter, read).
Node interactions	relation-word(emitter-name, receiver-name, signal, …).	interaction(consumer, counter, '+').

Based on Table 1, it is possible to imagine a real log like the following:

```
entity(date(2015,5,15,10,50,0,0,-,-),counter, 'LRU').
declares(date(2015,5,15,10,50,0,0,-,-),counter, '+').
declares(date(2015,5,15,10,50,0,0,-,-),(counter, '-').

entity(date(2015,5,15,10,50,0,0,-,-),clock, 'LRU').

interaction(date(2015,5,15,10,50,0,10,-,-), clock, counter, '+').
...
interaction(date(2015,5,15,10,50,0,90,-,-), clock, counter, '+').
interaction(date(2015,5,15,10,50,0,100,-,-), clock, counter, '-').
interaction(date(2015,5,15,10,50,0,110,-,-), clock, counter, '-').
interaction(date(2015,5,15,10,50,0,130,-,-), clock, counter, '-').
interaction(date(2015,5,15,10,50,0,140,-,-), clock, counter, '+').
interaction(date(2015,5,15,10,50,0,160,-,-), clock, counter, '+').
interaction(date(2015,5,15,10,50,0,170,-,-), clock, counter, '+').
...
event(date(2015,5,15,10,51,0,0,-,-), counter, minute).
```

Figure 2: Real Log Example.

With this log file it is possible to query it for retrieving all the messages prior a certain date:
interaction(S,X,Y,Z),S@<date(2015,5,15,10,50,0,31,-,-).

But it is also possible to make complex query like counting '+ signal' and '- signal' to check if the difference is what we expect:
findall([], interaction(D,X,Y,'+'), L), length(L, N), findall([], interaction(D, X,Y,'-'), L1), length(L1, N1), Tot is N-N1.

†† An LRU in SKA terminology means " line replaceable unit".

An Emom Device

In order to demonstrate the power of a declarative approach for the logging service, it is helpful to introduce an example that allows to generate a meaningful log.

Emom, in the context of a gym, means "every minute on the minute" and it is a technique for training for which a gymnast has to make an exercise every minute in less than a minute. Usually in an even minute is an exercise while in an odd minute is another one. To help people with this practice many app have been created (usually for smartphone) which indicate with a different colour whether the current minute is odd or even (see Figure 3: Emom Gui Even/Odd).

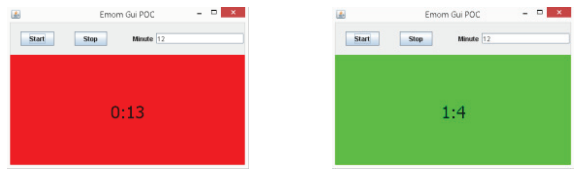


Figure 3: Emom Gui Even/Odd.

Based on the counter example, it is possible to build the Emom device with the help of two Counter Devices, one for the seconds and the other one for the minutes. In the Emom device, there will be a thread which will send a command every second to the seconds counter and, in case of the reset event, it will send a "PlusOne" command to the minute counter. When a minute changes an event is raised from the device: an "Odd" event if the minute is odd, an "Even" event if the minute is even.

The commands owned by the device are:

- *Start*: start the emom process;
- *Stop*: stop the emom process;
- *Reset*: reset the two counters;
- *SetMinuteCounterDevName*: set the name of the device for minute counting;
- *SetSecondCounterDevName*: set the name of the device for second counting.

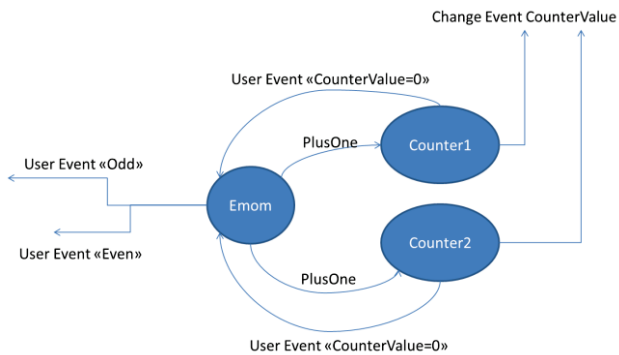


Figure 4: The Emom device.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The emom process is summarized in Figure 4:

1. The emom device sends a «PlusOne» command every second to the first counter (for second);
2. An event is generated where the counter second reset its value;
3. When happens a «PlusOne» command to the second counter (for minute);
4. An event is generated where the counter minute reset its value;
5. If minute is «odd», the emom throw the corresponding event otherwise an «even» event.

Log4Prolog: A Tango LogViewer Branch

To be able to take advantage of the declarative approach with Prolog log message, it is necessary to modify the standard Tango LogViewer creating a software branch. The purpose is to create a plugin architecture so that a developer can create a specific filter for his development or simply use a logic console editor to ask some particular or specific question to the engine.

Every filter is a custom class that has to implement a specific interface called *IFilter*. When starting the application the application instantiates the filter present in the configuration file (an xml file) and adds the plugin in the application. A filter has the ability to show a new popup (with a title and size) or directly add a new control inside the main control panel of the user interface. The running application is shown in Figure 5: The Log4Prolog Application where it is shown six plugin: four of them are simple controls added to the main user interface (which emulate the standard filters of the old Tango LogViewer app) and the other two are define two dialogs. The first one is a prolog console where an expert can ask high level question while the second one (the upper one in the figure) is a specific plugin created for the emom example (called emom log analyser).

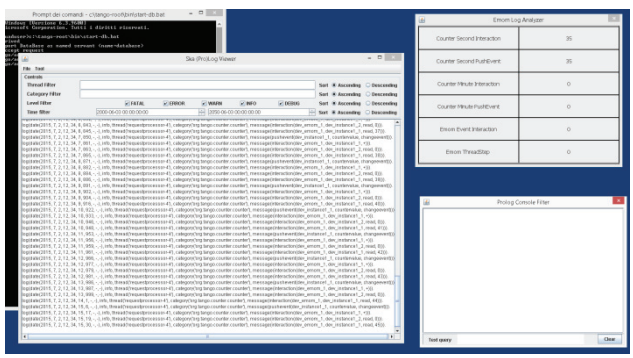


Figure 5: The Log4Prolog Application.

In specific the Emom log analyser define the logic query described in Table 2.

Table 2: Query Explanation

Title	Log query explanation
Counter second interaction	Find all interaction messages sent *to* the second counter device then count them.
Counter second pushEvent	Find all push-event[‡‡] messages sent *from* the second counter device then count them.
Counter minute interaction	Find all interaction messages sent *to* the minute counter device then count them.
Counter minute pushEvent	Find all push-event messages sent *from* the minute counter device then count them
Emom event interaction	Find all event-interaction messages sent *to* the emom device then count them.
Emom threadStep	Find all thread-step messages sent *from* the emom device then count them.

CONCLUSION

In this document a declarative approach in a logging system has been described. A good way to develop software is thinking in term of test and unit test. In addition to these techniques it is desirable to think in terms of logging so that the log file can be a way to read what the software is doing. If it is chosen to use a declarative language it is possible to work with files ready to be queried because of the formalism used.

The main disadvantage is related to the declarative language itself. However, by thinking the language as aimed at creating artificial intelligence or as a *sophisticated database language* (more NoSQL-like than most NoSQL approaches), it appears just like any other tool in the collection of a software architect that can help in solving specific problems.

REFERENCES

[1] JM. Chaize, A. Goetz, WD. Klotz, J. Meyer, M. Perez, E. Taurel, P. Verdier, “The ESRF TANGO control system status”, ARXIV , November 2011

‡‡ A push event in Tango occurs when the developer wants to force an event to be thrown.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

DRAMA 2 - AN EVOLUTIONARY LEAP FOR THE DRAMA ENVIRONMENT FOR INSTRUMENTATION SOFTWARE DEVELOPMENT

T. Farrell, K. Shortridge, Australian Astronomical Observatory, Australia

Abstract

The DRAMA Environment provides an API for distributed instrument software development. It originated at the Anglo-Australian Observatory (now Australian Astronomical Observatory) in the early 1990s, in response to the need for a software environment for large distributed and heterogeneous systems, with some components requiring real-time performance. It was first used for the AAOs 2dF fibre positioner project for the Anglo-Australian Telescope. DRAMA is used for most AAO systems and is or has been used at various other observatories looking for a similar solution. Whilst DRAMA has evolved and many features were added, the overall design has not changed. It was still a largely C language based system, with some C++ wrappers. It did not provide good support for threading or exceptions. Ideas for proper thread support within DRAMA have been in development for some years, but C++11 has provided many features that allow a high quality implementation. We have taken the opportunity provided by C++11 to make significant changes to the DRAMA API, producing a modern and more reliable interface to DRAMA, known as DRAMA2.

INTRODUCTION

The DRAMA API [1] remains the AAO's primary tool for constructing complex instrumentation systems and has been/is being used by various other observatories. With an approach based on the older Starlink ADAM Environment [2], it implements a tasking model; with each named task responding to named messages of a number of different types. In a DRAMA "System", tasks can run across different hosts in a heterogeneous environment. DRAMA was implemented from about 1992 and was designed to be highly portable at a time before ANSI C was available on all machines of interest. It has been run on many flavours of UNIX/Linux, VMS, VxWorks and MS Windows, and provided the ability to write soft[1] real-time applications and with good performance on, for example, 30Mhz 68020 CPUs. The flexibility allowed systems as complex as the AAO's 2dF system [3] to be implemented, making use of the most appropriate hardware for each job across a distributed system.

Most work is a DRAMA task is done in response to "Obey" messages – in effect, command messages; implementing "Actions". The design approach implements co-operative multi-tasking; multiple actions can be running at the same time but must deliberately return control to the DRAMA message reading loop between events to allow other actions to run and for the action itself to be "Kicked" – sent a message to change its behaviour in some way (typically, but not always, to cancel the action cleanly). The approach has worked well and a strongly objected-oriented task design approach was implementable for tasks written in C.

Attempts were made starting about 1994 to implement C++ interfaces for DRAMA, but the results were relatively poor and various different approaches were tried. One of the early issues was the poor portability of early C++ compilers, some features such as templates and exceptions were not reliably implemented and were not portable. Another was that we were still learning the best approaches to use.

Whilst DRAMA tasks using threads of various types have been implemented over the years, DRAMA itself has not supported using threads, with its own co-operative multi-tasking technique sufficient in most cases being more portable then threads were. In the C API, task authors must work around DRAMA when using threads; but in recent times, many libraries for component control have presumed threads are available and thread support has become widespread and is presumed to be available by most software engineers. We had been working on designs for proper thread support for DRAMA over some years, but had not yet implemented it.

C++11 [4] was a major revamp to the C++ language: Threads are now supported using a well thought out approach, by the compilers and standard libraries; Many new features are provided by C++11 that assist library implementers to construct quality interfaces; compilers of interest (GCC and Clang in particular) have implemented the full feature set on machines of interest (Linux and Mac OS X). We have taken advantage of the upgrade of C++ to implement DRAMA2, which will simplify writing and maintaining complex DRAMA tasks.

BASIC DRAMA

To understand DRAMA2, a quick introduction to DRAMA is needed. A DRAMA task executes a message receive loop that dispatches control to event/message handlers when messages arrive. A "Path" is a connection to another task, which is opened as required by application specific code and then used to send messages. There are various types of messages sent between tasks, which may be running across various machines on the network.

1 "Soft" Real-Time: Don't bet a life on an interrupt response, but good enough for ground based astronomy.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Application specific code is provided to handle the "Obey" and "Kick" message types. The "Obey" message type causes application specific code implementing an "Action" of a specified "Name" to be run. "Kick" messages are used to communicate with a running Action of the "Name" in the message. The implementation of an action of a given "Name" is provided by application specific C language routines that are invoked in response to messages by the DRAMA event loop.

A Parameter system allows a task to publish information about itself, which can be used for enquiries and to display information on GUIs. Get/Set/Monitor type messages containing the parameter "Name" are used to work with parameters in other tasks.

Most messages can have an "Argument" attached, which provides a way of moving data across machines. These arguments are implemented using Self Defining Structures (SDS). These can be of any size and are designed to allow large amounts of data to be sent efficiently between tasks, both locally and across heterogeneous networks configurations. They are used for simple command arguments, complex structures and large image transfers.

Internally, a single global variable (a large structure) is used to maintain all the DRAMA details – in retrospect, possibly a mistake, but this allows a simpler API that did not need a DRAMA task pointer to be passed to all DRAMA API calls.

THE APPROACH TO DRAMA2

The basic approach used in DRAMA2 is based on work previously done to implement DRAMA GUIs with JAVA. In this work, the JAVA Native Interface (JNI) was used to allow JAVA to invoke and be invoked by the DRAMA API. The approach was directed to implementing GUIs but allowed many of the concepts that are required for DRAMA2, to be developed. Four particular areas drive the design:

Implementation as Wrapper around the C API

The DRAMA JAVA interface proved that dramatic changes to the C level interfaces to DRAMA are not required for thread and exception support. By implementing DRAMA2 as a set of wrappers around DRAMA C APIs, compatibility with the large set of existing tasks can be easily maintained and DRAMA2 could be implemented quickly.

Only One Thread Reading DRAMA Messages

There is one and only one thread that actually blocks for and reads DRAMA messages from the underlying message queue. Most of the "DRAMA" internal processing is done within this thread. Other threads can send DRAMA messages (and make other DRAMA API calls) but cannot actually read the messages directly. Much potential complexity is removed and no changes are required to the DRAMA C language internals when using this approach. If another thread needs to wait for a DRAMA message to occur, it must wait on a C++ condition, which is notified by the DRAMA thread when the message arrives.

Locking Access to DRAMA Structures

Locking is important to get right! Systems with multiple locks help to avoid lock wait delays, but significantly increase the complexity of the design required to ensure avoiding deadlocks. Since DRAMA2 is an API available to implement applications, it is much harder to avoid deadlocks when using multiple locks. As a result, only one lock is used and it must be taken by most methods that invoke the DRAMA C API. Use of the lock is normally internal to the DRAMA2 methods, but it can be used by application specific code to access any DRAMA C API not yet available or for application specific locking. Use of the DRAMA2 lock as the only lock in the application would avoid deadlock. The DRAMA2 lock is safe for recursive use – a thread that has already taken the lock will not deadlock if it attempts to take it a second time.

The DRAMA design allows the DRAMA2 message read thread to block waiting for a message without taking the lock. That thread only takes the lock when processing a message. Since the lock is free any time the DRAMA2 message read thread is waiting for a message, there is plenty of opportunity for application threads to lock access to DRAMA and send messages themselves.

Status and Error Reports vs. C++ Exceptions

The DRAMA C API uses an inherited status convention. Most functions have a "status" argument, which is a pointer to an integral type. Functions are expected to check the value pointed to is zero on entry. If it is not, they return immediately. If an error occurs, status is set to a non-zero value. The Inherited status convention neatly avoids the long series of nested if statements typical in the use C APIs that returns a value to indicate if they failed. The integer status value is passed as the result of DRAMA messages, allowing other tasks to determine if an Action has failed and to interpret the status value. An Error Reporting System (ERS) enables extra contextual information to be added when errors occur.

In C++, it is natural to replace the inherited status approach and ERS by exceptions. An exception class is provided which is a sub-class of std::exception. Any DRAMA2 method invoking a DRAMA C API must check the status returned and, if bad raise an exception. The DRAMA2 exception class stores the integer status value and information about the context and location of the exception.

At any point where the DRAMA C API must invoke a DRAMA2 method, there must be an interface function. That has an inherited status argument. This function must catch any exception thrown by DRAMA2. If the exception is the DRAMA2 exception, the original status value will be available and can be returned to the C API as the status of the call, otherwise another status value

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

will be returned. Any extra context available in the exception will be reported using ERS.

TASK STRUCTURE

A Simple Task

Example 1, below, shows "Hello World" in DRAMA2. This program implements a task named "TASK1", which has just one Action – named "HELLO". Sending an Obey message with the name "HELLO" will result in the message "Hello World" being output and the task then exiting. The action is implemented by sub-classing the abstract class "MessageHandler" providing an implementation of "MessageReceived()". Any number of actions can be added in a similar way and they don't normally cause the task to exit, and may be invoked multiple times in sequence.

```
#include "drama.hh"
using namespace drama;
// Action definition.
class Action1 : public MessageHandler{
private:
  Request MessageReceived() override {
    MessageUser("Hello World");
    return  RequestCode::Exit;
  }
};
// Task Definition
class ExTask : public Task {
private:
  // actions
  Action1 Action1Obj;
public:
  ExTask(const std::string &taskName)
:
    Task(taskName) {
    Add("HELLO",
        MessageHandlerPtr(&Action1Obj,
                          nodel()));
  }
};
// Main program.
int main() {
  CreateRunDramaTask<ExTask>("TASK1");
  return 0;
}
```

Example 1: "Hello World" in DRAMA2.

Threaded Actions

In Example 1, the "HELLO" action is running in the main DRAMA2 thread. Whilst it can "Reschedule", in the traditional DRAMA way to return control to the message thread, the intent of DRAMA2 is to support running actions in threads. The class "thread::TAction" is an abstract sub-class of "MessageHandler". The user of this class must provide the method "ActionThread", which is invoked within a thread when an Obey message of the specified name is received. When the thread completes, DRAMA2 is informed and the action is marked as completed. Importantly, all details of thread creation; joining the thread etc. is hidden by DRAMA2. Any exception thrown by the thread is reported via DRAMA2 as an action failure – the task does not abort. Example 2 below shows a simple implementation of a thread action.

```
// Action definition.
class Action1 : public
thread::TAction{
public:
  Action1(std::weak_ptr<Task>
theTask):
    TAction(theTask) {}
private:
  void ActionThread(const sds::Id &)
override {
    MessageUser("Hello World - from a
thread");
  }
};
```

Example 2: Threaded "Hello World" in DRAMA2.

Action threads can create their own sub-threads, which can interact with DRAMA, but the implementer is then responsible for handling creation, joining the thread, dealing with exceptions in the thread, etc.

An important requirement for implementing access to DRAMA APIs from threads is to get the DRAMA "Context" right. In a normal DRAMA task, with only one thread runing, the DRAMA API has been able to presume that certain components of the DRAMA Global structure indicate which action is running, and other information about that action. When a threaded action is started by DRAMA2, this information is captured just before the thread is started. Later, any call to a DRAMA C API from the threaded action must first lock access to DRAMA, save the current DRAMA context and then enable its own DRAMA context. This must be undone when the call to the DRAMA C API is complete. An object of a particular DRAMA2 class is used to wrap this up using RAII (Resource Allocation Is Initialisation) to ensure it is undone correctly even is an exception is thrown. This is done transparently by the DRAMA2 API, but is available to application code if access to other DRAMA C API's is required.

Kicking Threaded Actions

A DRAMA Action can be "Kicked", which provides a method for other tasks to communicate with a running action. Often used for Action cancellation, Kick messages are flexible and a system design may use them to update a running action. The "WaitForKick()" method and related methods allow a thread to wait for a kick message to be received.

Alternatively, a "KickNotifier" object may be created before say entering a CPU intensive loop. These objects create a thread that waits for a kick message. The caller can ask the object if a kick was received and respond correctly.

Sending Messages

A "Path" class is provided to enable sending DRAMA messages to other tasks. In traditional C DRAMA,

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

message sending does not block and an action must explicitly reschedule to message handle replies. In DRAMA2, message sending is only possible from a threaded action. The thread, but not the task, is blocked to await replies. As a threaded action can have child threads, they may have any number of messages outstanding at any time. Example 3 shows how to send an Obey message to a server. In this case, the action name is "HELLO".

```
Path server(…)
…
server.Obey(this, HELLO);
```

Example 3: Sending an Obey message.

There are various message sending methods, including the ability to monitor for changes to the values of parameters in other tasks. By default, the methods will block until the subsidiary action completes, but there are features allowing overriding of the default processing of the various possible replies to a message.

OTHER FEATURES

SDS

The sds::Id class provides access to DRAMA's SDS objects, used to send data between tasks. Action implementations can access any structure sent to them and can send such structures as arguments in any message they send. Some complexities of the underlying SDS system made writing a clean C++ interface hard prior to C++11. In C++11, the move assignment and move copy operators proved liberating, allowing an effective and relatively clean interface to be constructed.

The "sds::Id" class is extensive, providing methods allowing the building and accessing of complex structures, as well as providing simple ways of building and accessing typical command line arguments.

The sds::Id class allows SDS structures to be written to and read from files, buffers of various forms (e.g. for sending in messages) and for details of such structures to be listed to streams and other locations, for debugging purposes. Simple and highly efficient access to large arrays is provided.

Accessing Command Arguments

All arguments to actions are sent in an SDS structure, but there is a standard approach to command arguments, which allows simple generic programs to be used send to obey messages to any task. Various simple methods are provided by the sds::Id class to construct such arguments. In the action receiving the message, sds::Id class methods can be used, but there is also an alterative interface – via the "gitarg" namespace. These are a series of classes that create sub-classes of standard types initialised from an SDS structure. For example, a gitarg::Bool uses an SDS structure to initialise a Boolean type, accepting for example string values "YES" and "NO" to indicate the value.

GUIs

DRAMA provides a number of GUI toolkits, Java and Tcl/Tk based GUIs being commonly used at this point. These will continue to work with DRAMA2 tasks. Additionally, new toolkits will be constructed using DRAMA2, with Python likely to be the first.

The support for working with threaded systems easily ensures DRAMA2 can be used with many other modern systems. The first DRAMA2 task implemented outside the package itself was a GUI using the Qt widget set, an extensively threaded environment.

Documentation and Regression Testing

An important part of the implementation of DRAMA2 was to ensure the documentation was created with the package, rather then the tradition of being tacked on later. The "doxygen" tool was chosen as the interface documentation tool and all interfaces have been documented as the code was written.

A 130-page manual has been generated, working through all the many features of DRAMA2. The manual includes a large number of code examples, all of which is available as compilable code. Generation of the detailed manual and the required examples quickly highlighted various flaws or unnecessary complexities in the initial interfaces, allowing them to be revamped before release.

As example programs were generated to demonstrate and test features they have been added to our regression test facilities. As a result, any change to DRAMA2, or the underlying DRAMA software, is automatically subject to extensive testing.

CONCLUSION

DRAMA2 has quickly modernized the development of DRAMA tasks. It is well documented and allows reliable tasks to be written quickly. It allows sequenced code to be written for sequenced jobs, but with all the efficient non-blocking DRAMA messaging facilities available. Much of the (potentially risky) complexity of creating threaded distributed applications is hidden from programmers using DRAMA2, in the most common cases.

REFERENCES

[1] T.J. Farrell, K. Shortridge, J.A. Bailey, "DRAMA: An Environment for Instrumentation Software," Bulletin of the American Astronomical Society, Volume 25, No 2 (1993).

[2] Allan P. M., "The ADAM software environment," Astronomical Data Analysis Software and Systems I, 126 (1992).

[3] Taylor K., et al., "Anglo-Australian Telescope's 2dF Facility", Proc. SPIE 2871 (1997).

[4] ISO/IEC 14882:2011 "Information technology -- Programming languages -- C++" (2011).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

A MODULAR SOFTWARE ARCHITECTURE FOR APPLICATIONS THAT SUPPORT ACCELERATOR COMMISSIONING AT MedAustron

M. Hager*, M. Regodic#, EBG MedAustron, Wiener Neustadt, Austria

Abstract

The commissioning and operation of an accelerator requires a large set of supportive applications. Especially in the early stages, these tools have to work with unfinished and changing systems. To allow the implementation of applications that are dynamic enough for this environment, a dedicated software architecture, the Operational Application (OpApp) architecture, has been developed at MedAustron. The main ideas of the architecture are a separation of functionality into reusable execution modules and a flexible and intuitive composition of the modules into bigger modules and applications. Execution modules are implemented for the acquisition of beam measurements, the generation of cycle dependent data, the access to a database and other tasks. On this basis, Operational Applications for a wide variety of use cases can be created, from small helper tools to interactive beam commissioning applications with graphical user interfaces. This contribution outlines the OpApp architecture and the implementation of the most frequently used applications.

INTRODUCTION

The heart of MedAustron is a synchrotron-based accelerator. The accelerator provides the possibility to generate a large range of different ion beams, for example Proton and Carbon beam with different beam sizes and hundreds of different energies. A backside of synchrotron-based accelerators is the high number of components and the complex control of the beam. The commissioning of such an accelerator is a laborious task that requires support by software to be executed efficiently.

One challenge in the development of such applications is the agility of the commissioning process. While the commissioning progresses, new components are integrated, existing ones change and the understanding of the accelerator and the beam behavior deepens. Software that supports the process has to be easily adaptable to the evolving environment. Previous architectural concepts turned out to be too restrictive for these demands. Therefore, a new, modular architecture has been developed at MedAustron, the OpApp architecture.

APPROACH

The OpApp framework is connected to the MedAustron Accelerator Control System (MACS), [1]. With the connection to MACS, OpApps can: A) Send commands to the systems and devices of the accelerator B) Apply device settings and retrieve state information. C) Request beam cycles D) Receive measurements and timing information.

The framework is also connected to a database. OpApps use the database to store data related to the beam generation as well as acquired measurements and accelerator configuration. OpApps also retrieve and analyze stored data.

Functionality

Main domain of Operational Applications is the beam commissioning. OpApps can compute settings, like currents and voltages, for all accelerator devices based on the optical setup of the accelerator and the desired beam characteristics. OpApps can also apply the settings, request beam cycles and measure the characteristics of the generated beam. With this, OpApps can automatize many aspects of the measurement-based beam commissioning workflows. The role of Operational Applications in the workflow is depicted in Figure 1.

OpApps contribute to the processing and analysis of measurements but usually leave complex analysis tasks to dedicated tools. To provide data to other tools, OpApps can generate files in a variety of different formats.

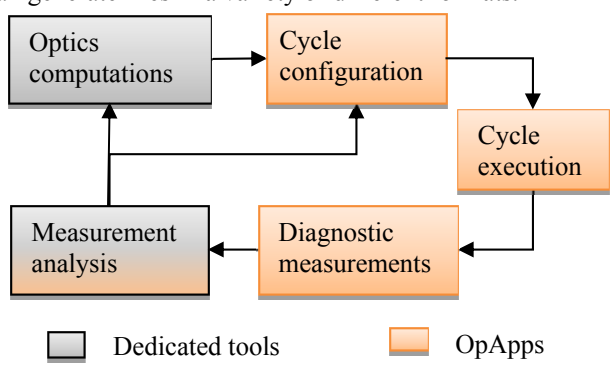


Figure 1: The measurement-based beam commissioning workflow

Although the OpApp concept was developed with beam commissioning in mind, OpApps are by far not limited to it. Operational Applications can also be used for:

- Quality Assurance (QA) - OpApps that acquire measurements and compare them with stored reference data can be used for a regular QA of the beam characteristics
- Configuration Management - With a set of database related execution modules OpApps can, for example, help the user to import device specifications into the database or export data for the Control System
- Accelerator and Beam Monitoring - OpApps can be used to acquire accelerator and beam data in the

* markus.hager@medaustron.at
# milovan.regodic@medaustron.at

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

background and log this data into the database. Additionally OpApps can analyse the stored data and generate reports, for example of the accelerator performance.

OPAPP ARCHITECTURE

Most beam commissioning activities involve the execution of the same core tasks, for example: "Request an accelerator cycle" or "Take a measurement with a beam diagnostic device". Often the combination of some core tasks builds an activity that is executed as part of other commissioning activities. A trajectory measurement, for example, requires a series of cycle requests and beam position measurements. The trajectory measurement itself is used in a trajectory steering where it is combined with the application of device settings, i.e.: apply settings, measure the trajectory, (if necessary) apply better settings, and so on. Based on this realization, the OpApp architecture enforces a separation of the core commissioning tasks into dedicated modules and defines a mechanism that allows a flexible composition of the different modules.

Core Execution Modules

The core execution modules of the OpApp architecture are separated into two different layers. On layer, represented by *Executors*, is specific to devices and data structures. The other layer contains *Repositories* that encapsulates the interaction with connected systems, like MACS or the database. The repositories in this layer work with generic data structures.

An example for the separation into the two layers is the execution of beam diagnostic measurements. For each group of beam monitors an own Executor is implemented that handles the specifics of the measurement, for example a beam intensity measurement or a profile measurement. All measurement executors, however, use the same measurement repository. The measurement repository implements the generic access to the measurement interface provided by MACS. Figure 2 shows the main measurement executors and the repositories they use.

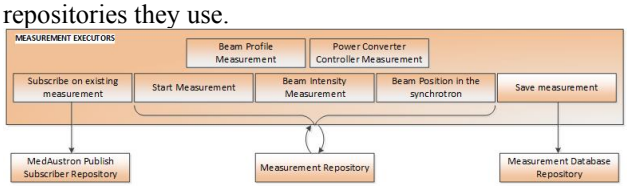


Figure 2: Measurement Executors and the Repositories they use

Changes in the interface of the connected systems only affect the repository layer but are not reflected into the OpApp business logic, which results in highly maintainable code. The centralization of the accessing logic also has another big advantage: It provides a way of testing operational applications without being connected to the real system. To test OpApps, "Demo" repositories are implemented that simulate the response of the different systems. The OpApp framework can switch, with a simple flag in the code, from the "live" repositories to the "demo" repositories. After the switch to demo mode the OpApps work as before but on simulated connections.

Module composition

In the OpApp architecture, all execution modules get registered in the OpApp framework. All modules also have access to this registry, to enable access from every module to every module. This allows a flexible composition of modules into bigger modules and applications.

OpApp Language

Operational Applications are usually developed by the Controls team on request of domain experts. Often the requested functionality is an automatization of simple but time consuming routines, like the acquisition of an extensive set of measurements. If domain experts could write these routines themselves, they wouldn't have to request their development and wait for the implementation.

To allow domain experts to contribute to the development, the OpApp architecture specifies the implementation of an own OpApp language. Via the language the different execution modules can be retrieved from the OpApp framework and called in an intuitive way, similar to a natural-language. The trajectory measurement module, for example, can be executed with the following line of code:

MeasureTrajectory.In(*beamLine*).

Figure 3 shows the concept of the OpApp language and the composition of the execution modules.

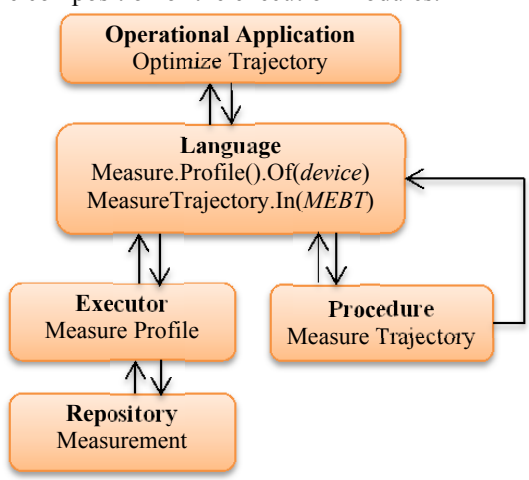


Figure 3: Execution modules and their composition via the Language

Accelerator Models

OpApps require information about the accelerator. They must know which elements are available and which properties of these elements they can influence. OpApps also require information about beam optical parameters, for the computation of device settings. To include this

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

information into the framework, the OpApp architecture uses several models. The code of all models is generated from the accelerator configuration, to allow an adjustment of the code when changes in the configuration occur.

One of the models is the accelerator model. This model contains the elements of the accelerator and their controllers, for example:

- S1-01-000-MBH (CS-03-031-PCC)
- S1-00-000-MCX
  - S1-00-000-MCH (CS-03-211-PCC)
  - S1-00-000-MCV (CS-03-246-PCC)

From the model, single elements can be retrieved but also all elements that belong to a certain part of the accelerator or a certain device group. The model information is available in the Language, which allows commands like: SetCurrentOf(S1-01-000-MBH).To(20);

IMPLEMENTATION

Operational Applications are developed in a Microsoft .Net environment with C#.

Language

The OpApp Language is implemented with a fluent API. A fluent API is a programming interface that employs mechanisms like method chaining to allow the creation of highly readable code.

Figure 4 shows an example for the implementation of the fluent API. In the example an OpApp sets the current for a magnet. The OpApp calls:

SetCurrent.Of(*magnet*).To(*current*).

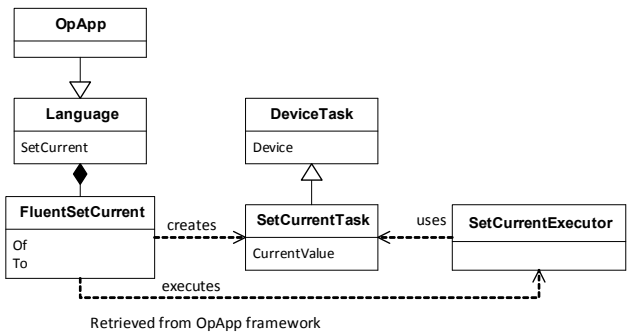


Figure 4: Implementation of the fluent API

The OpApp framework manages all execution modules in a Dependency Injection container. For the execution of the SetCurrent command, the Language retrieves the SetCurrentExecutor from this container. The name of the magnet and the current to be applied are stored in a SetCurrentTask object. This object is forwarded to the executor in the To() method. The executor uses an according repository to apply the current to the device via MACS.

User Interfaces

Graphical User Interfaces (GUI) for Operational Applications are developed with the .Net WPF framework.

In applications that require a high interaction with the user, the GUI can get quite complex. This means that a lot of interaction logic must be implemented. In addition, GUI-based OpApps require asynchronous mechanisms to interact with the execution modules. GUI-based OpApps are, therefore, developed by software engineers.

OpApps that execute simpler routines only require some input values but not a complex user interface. To allow the creation of simpler, language-based applications by domain experts, a common user interface has been implemented together with a library of visualization modules for different input parameters. OpApps authors mark the required input parameters in the code with special attributes. The user interface reads the attributes and automatically displays the according fields. Figure 5 and Figure 6 show an example of the code attributes and the resulting display in the user interface.

```
[ElementListParameter(AllChecked = false,
    DisplayName = "Monitors", Class = new ElementClass[] {
    ElementClass.QIM, ElementClass.ORB, ElementClass.QPM,
    ElementClass.SFX, })]
public List<BDElement> Monitors { get; set; }
```

Figure 5: Example of parameter attribute

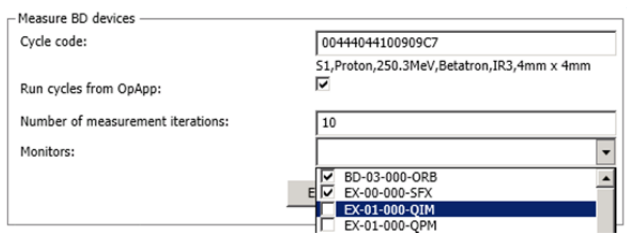


Figure 6: Automatically generated user interface

RESULTS

Around 1.5 man years of work have already been spent on the development of Operational Applications. During this time the framework has been developed, together with about 25 *Atomic* OpApps, one *Complex* application and one *Monitoring* application.

Atomic OpApps

Atomic OpApps are started via a common, generic user interface. Atomic OpApps use the OpApp language and can potentially be written by domain experts. Examples for routines executed by Atomic OpApps are trajectory measurements, kick response measurements and checks for incorrect device settings. Figure 7 shows an example for a trajectory measurement written with the OpApp language.

```
var trajectory = CreateEmptyTrajectory();
foreach (var monitor in ProfileMonitors.AllIn(transferLine))
{
    var beamProfile = MeasureProfile.With(monitor);
    var position =
        CalculateBeamPosition
          .WithBias(percent: 15).From(beamProfile);
    trajectory.AddPosition(postion);
}
```

Figure 7: Example for a fluent trajectory measurement

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Complex OpApps

Complex OpApps are interactive applications that are operated via their own user interfaces. This type of applications uses the execution modules but not necessarily the language.

One Complex application has already been developed, the Beam Scan OpApp. Main functionality of the OpApp is the application of device settings in a given range and a subsequent measurement of the beam. In this way, the OpApp allows the determination of the setting that results in the best beam characteristics.

The Beam Scan OpApp has turned out to be extremely helpful. The application supports the scan of many optical parameters and a wide range of device, monitor combinations. 8 shows a scan of a synchrotron RF system setting with a beam current measurement.

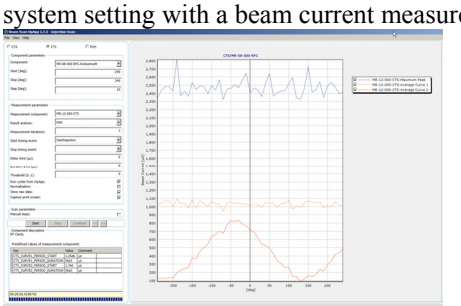


Figure 8: Screenshot of the Beam Scan OpApp

Monitoring OpApps

Monitoring OpApps continuously collect and store machine and beam related data. They run on dedicated machines and don't require any user interaction.

The Particle Logger is the first Monitoring OpApp. The application is composed of two parts. One part parasitically acquires measurements of the beam current in the synchrotron and logs this information together with beam cycle data into the database. The second part is an analysis application that retrieves this data, displays it and generates performance indicators. One indicator for the accelerator performance is the number of cycles in a given period in that more than a certain number of particles were measured in the beam. Figure 9 shows an example of a chart that contains the number of particles in the cycles generated over 24 hours.

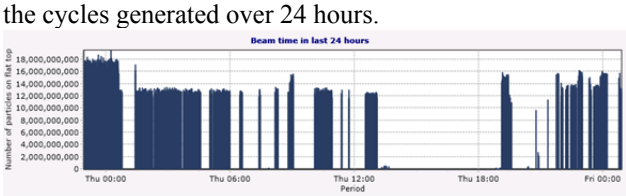


Figure 9: Screenshot of the Particle Logger Analysis application

Operational Applications have also been presented at the IPAC 2015. The according contribution, [2], contains further OpApp examples.

OUTLOOK

The OpApp development at MedAustron will continue. The OpApp framework will be extended and improved, for example, with a synchronization mechanism between asynchronously executed tasks.

Despite of the availability of the fluent language, OpApps have so far only been developed by software engineers. In order to encourage the participation of domain experts, an out-of-the-box development environment and user guides will be prepared.

Upcoming OpApps will focus on advanced beam commissioning tasks and on aspects of the accelerator operation, mainly: Quality Assurance, Configuration Management and Accelerator Monitoring.

With the integration of dosimetric measurement equipment and the Treatment Control System, OpApps could in the future also be used for treatment-related QA procedures and allow the tuning of the accelerator from a treatment perspective.

CONCLUSION

The OpApp architecture has proven to be an excellent basis for the development of software solutions that support the commissioning and operation of the MedAustron accelerator. The modular design, enforced by the architecture, has shown to allow a very quick adaptation and development of applications.

OpApps already build an indispensable set of tools for the commissioning and operation of the MedAustron accelerator – and their importance will continue to grow, as there are many supportive applications waiting to be developed.

ACKNOWLEDGEMENTS

We would like to thank A. Wastl (MedAustron) for his important contributions to the OpApp development and K. Fuchsberger (CERN) for his help in working out the initial OpApp concept. We would also like to thank J. Junuzovic (MedAustron) for her support.

REFERENCES

[1] J. Gutleber, et al., The MedAustron Accelerator Control System, Proceedings of ICALEPCS 2011, Grenoble, France

[2] A. Wastl, M. Hager, M. Regodic, Operational Applications – A Software Framework used for the Commissioning of the MedAustron Accelerator, Proceedings of IPAC 2015, Richmond, VA, USA

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SOLVING THE SYNCHRONIZATION PROBLEM IN MULTI-CORE EMBEDDED REAL-TIME SYSTEMS

F. Hoguin, S. Deghaye, CERN, Geneva, Switzerland

Abstract

Multi-core CPUs have become the standard in embedded real-time systems. In such systems, where several tasks run simultaneously, developers can no longer rely on high priority tasks blocking low priority tasks. In typical control systems, low priority tasks are dedicated to receiving settings from the control room, and high priority real-time tasks, triggered by external events, control the underlying hardware based on these settings. Settings' correctness is of paramount importance and they must be modified atomically from a real-time task point of view. This is not feasible in multi-core environments using classic double-buffer approaches, mainly because real-time tasks can overlap, preventing buffer swaps. Other common synchronization solutions involving locking critical sections introduce unpredictable jitter on real-time tasks, which is not acceptable in CERN's control system. We present a lock-free, wait-free solution to this problem based on a triple buffer, guaranteeing atomicity no matter the number of concurrent tasks. The only drawback is potential synchronization delay on contention. This solution has been implemented and tested in CERN's real-time C++ framework.

FROM SINGLE-THREADED TO MULTI-THREADED EMBEDDED SYSTEMS

In typical control systems, accelerators settings are modified either manually by operators using graphical user interfaces or by high-level systems computing hardware settings from high-level values. These settings are then sent by the high-level applications to the computers in charge of the hardware real-time control. This control is done by real-time tasks in a limited time frame following an external trigger. A real-time task is nothing more than a piece of code executed with real-time priority. These tasks typically perform computations based on settings and drive the hardware with the values obtained. Hardware settings can be interdependent and modifications must be applied in a single operation. The real-time tasks must work with consistent sets of settings; from their point of view, settings must be modified atomically.

In single core embedded systems, developers can rely on the determinism of a real-time scheduler to guarantee consistency of settings with a simple double buffer system; one for active values (accessed by real-time tasks) and another one for pending values i.e. just modified values not yet accessible to real-time tasks. When the modification is done, the set of pending values is consistent and buffers may be swapped. A low priority task swaps pointers to the buffers atomically; the pending buffer becomes active, and vice versa. This is guaranteed to be safe provided no real-time task is ever in a waiting state in the middle of its execution, ensuring that the buffer swapping task can never be executed while a real-time task is being executed.

As multi-core embedded systems become the norm, such assumptions cannot be made anymore. At CERN, most of the control system embedded systems use 2-core processors. In such setups, the buffer swapping task can be executed concurrently to a real-time task. As a result, swapping pending and active buffers without further checks could lead to a real-time task reading inconsistent settings. An example is shown in table 1.

Table 1: Example of Naive Buffer Swap with Multi-core CPU

Real-time task	Buffer swapping thread	Buffer 1	Buffer 2
Reads voltage => 200V	Swap triggered…	V = 200, A = 100	V = 10, A = 1000
Busy computing…	Swaps buffers	V = 200, A = 100	V = 10, A = 1000
Reads current => 1000A		V = 200, A = 100	V = 10, A = 1000
Tells power supply: 200V, 1000A		V = 200, A = 100	V = 10, A = 1000

EXPLORED AND ABANDONED SOLUTIONS

In order to solve this problem, several solutions were explored. The goal was to implement an algorithm that fulfils the following requirements:

1. The solution shall be real-time compliant (in particular, no dynamic memory allocation is allowed).
2. Real-time tasks shall read consistent setting values throughout their execution.
3. Time interval between a trigger and the execution of the corresponding real-time task shall be constant and below 5 milliseconds (a jitter of 10% is acceptable).
4. Pending setting values shall be made available to real-time tasks within a reasonable time frame.

Snapshot of the Setting Values

A possible solution is to make a snapshot of the setting values just as a real-time task is about to start. This snapshot is private to that execution of the task. While this would work, it is not real-time compliant as memory would be dynamically allocated to copy the setting values. Even a fixed size memory pool could be exhausted given enough simultaneous real-time tasks. Also, the jitter between the trigger and the task execution

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

depends on the size of the setting values. This solution fails requirements #1 and #3.

Reader-Writer Lock Mechanism

In this implementation, real-time tasks acquire a reader lock just before starting their execution. Swapping settings buffers is done when the corresponding writer lock can be acquired. While this solution looks like it could work, there are unavoidable cases where a real-time task would be delayed beyond acceptable limits. This is typically the case when two real-time tasks overlap (see table 2 for an example).

Table 2: Reader-writer Example Preventing RT Task Execution

Real-time task A	Real-time task B	Buffer swapping task
Starts. Acquires reader mutex.		
Executing		Swap required. On hold waiting for writer mutex.
Executing	Starts. Blocks trying to acquire reader mutex because of "Buffer Swapping" task.	On hold
Executing	Blocked	On hold
Finishes. Releases reader mutex.	Blocked	Acquires witer mutex and swaps buffers.
	Acquires reader mutex.	Releases writer mutex.

In this situation, task B is blocked for as long as task A is executing, which is not acceptable as it fails requirement #3. To avoid real-time tasks blocking, readers could have priority over writers. But in this case, overlapping real-time tasks would prevent settings from ever being updated; the buffer swapping task would never acquire the writer mutex as, at any time, one of RTA or RTB would hold a reader mutex. This fails requirement #4.

A LOCK-FREE, WAIT-FREE SYNCHRONIZATION SOLUTION

Requirement #3 means that real-time tasks should never block and never wait. We demonstrated earlier that these requirements cannot be satisfied with a traditional pointer swapping implementation, and that using any form of locking before executing a real-time task is impossible.

We introduce a solution that, by the use of an additional setting values buffer, guarantees settings consistency without using any locking mechanism in real-time tasks. Furthermore, we guarantee that the delay between an event and the execution of the corresponding real-time task is fixed (requirement #3) by ensuring a fixed set of operations is executed in-between.

Instead of using an active settings buffer and a pending settings buffer, we use a reference settings buffer and two real-time settings buffers. The reference buffer contains the latest setting values, as modified by operators. This buffer is never accessed by real-time tasks. The real-time settings buffers are copies of the reference buffer at a certain point in time. From now on, we'll refer to the real-time buffers as buffer A and buffer B. They can be in one of the following four states:

- Current: the buffer can safely be accessed and contains the current settings.
- Obsolete: the buffer can safely be accessed and contains old settings.
- Modifiable: the buffer is not in use and cannot be accessed. It is waiting for an update of setting values.
- Updating: new setting values are being copied in the buffer.

Buffers A and B are always in different but related states. Transition between states for a single buffer is presented in Fig. 1.

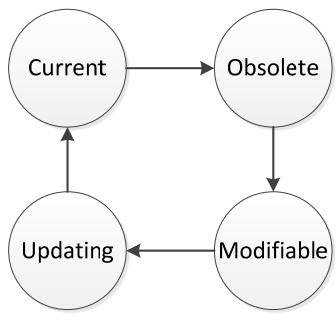


Figure 1 Buffer's states.

Overview of the Behaviour

The following simplified description focuses on buffer A; buffer B follows the same pattern. Relations between buffer states are described in depth in the next section.

When a real-time task reacts to an event, it becomes a reader of the *Current* buffer (e.g. buffer A). Whenever newer values are available in the other buffer (buffer B), buffer A becomes *Obsolete*. Buffer A is guaranteed to have its readers count eventually reduced to 0 since it cannot receive new readers. As soon as buffer A has no longer any readers, it becomes *Modifiable*. At some point in time, new settings will be available and will be copied from the reference buffer to buffer A; it goes to the state *Updating*. When the copy is done, buffer A becomes *Current* at the same time as buffer B becomes *Obsolete*, coming back to the initial state. Operations available on buffers depending on their state are listed in table 3.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Table 3: Buffer Access Rights

	Can be accessed by readers	Can get new readers	Can be modified
Current	Yes	Yes	No
Obsolete	Yes	No	No
Modifiable	No	No	Yes
Updating	No	No	Yes

States Transitions

A state machine representing this algorithm is presented in Fig. 2.

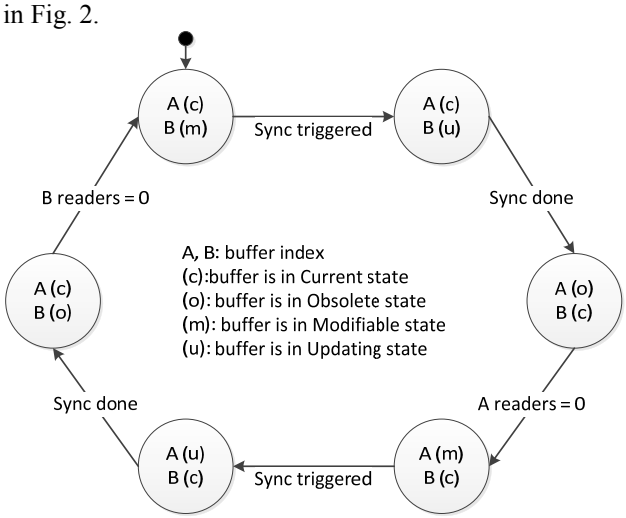


Figure 2: Complete two-buffer state machine.

At start-up, the content of the reference buffer is copied to both real-time buffers A and B, and buffer A is in *Current* state while buffer B is in *Modifiable* state. Whenever the reference buffer is modified and the modification operation committed, a synchronization is triggered ("Sync triggered" transition). Since buffer B is in the *Modifiable* state, setting values are copied by a low priority task (the buffer synchronizer) from the reference buffer to buffer B; the latter goes to the state *Updating*. If no buffer is *Modifiable*, the copy operation is put on hold. To ensure consistency of settings, the buffer synchronizer holds a reader lock on the reference buffer during the copy. This time interval depends on the number of modified settings, their size and the bandwidth of the computer memory. On modern systems, it is typically very short, between a few microseconds to a few milliseconds. During that period of time, the reference buffer cannot be modified and the threads writing the settings coming from the high-level application are blocked. This delay is acceptable as the transfer of settings uses the Ethernet network, which is not real-time. In addition, the buffer synchronizer keeps a list of just-copied settings, the "settings to replicate", which will be useful when the next synchronization occurs. Once the copy is done ("Sync done" transition), buffer A becomes *Obsolete* and buffer B becomes *Current*. This latter transition must be atomic to ensure that at any time, one and only one buffer is in the *Current* state. When buffer A has no readers any more, it becomes *Modifiable* ("A readers = 0" transition). The next time a synchronization will be triggered (second "Sync triggered" transition), the modified settings will be copied from the reference buffer to buffer A. This time though, this will not be sufficient as, at this point, buffer A is not up-to-date with respect to the settings that were copied earlier to buffer B. The buffer synchronizer needs to use the "settings to replicate" and copy them from buffer B to buffer A. When the copy is done (second "Sync done" transition), buffer A and buffer B atomically change state, going to *Current* and *Obsolete* respectively. When buffer B has no readers any more ("B readers = 0" transition), it goes back to the state *Modifiable*, which is the initial state.

IMPLEMENTATION

This algorithm has been implemented in CERN's real-time C++ framework. We present details of our implementation.

Atomic Operations

Some operations need to be carried out atomically to ensure proper functioning of the algorithm. The list of required atomic operations is as follows:

- Fetch and increment or increment and fetch on 32 bits (can be reduced to 8 bits)
- Decrement on 32 bits (can be reduced to 8 bits)
- Compare and swap on a pointer (optional, for validation purposes only, can be replaced by a write operation)

As explained earlier, it is required to always have one and only one buffer in the *Current* state. This is achieved by using a pointer to the *Current* buffer, whose value can be changed atomically (supported by all modern CPU architectures [1][2]). Instead of a simple write, an atomic compare and swap is used in our implementation to ensure that the pointer value is as expected before modifying it; this is for validation purposes only and, in production, only the write is required.

When a new reader requests access to the *Current* buffer, its readers count is incremented, and its index is retrieved and assigned to the reader. Since the *Current* buffer pointer can be changed at any time, this sequence (increment and read) needs to be atomic as well. Our implementation uses a structure that can be modified atomically. It contains the buffer index in the most significant part and the number of readers in the least significant part. We use 8 bits for the buffer index and 24 bits for the reader index (see Fig. 3). This structure allows us to use a "fetch and increment" that will atomically increment the number of readers and read the buffer index. This primitive is again supported by all modern CPU architectures [3][4]. Note that on architectures with limited resources, the number of bits can be reduced from 8 to 1 and from 24 to 7 bits respectively, while still allowing a maximum of 128 readers.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

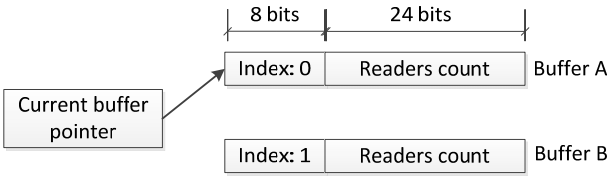


Figure 3: Internal structure for buffer management.

When a reader no longer needs access to a buffer, its readers count must be decremented. This also needs to be atomic because other readers may simultaneously modify this count. However, in this case, there is no need to retrieve the buffer index as it was retrieved when requesting access to the buffer. Therefore, an atomic decrement is sufficient and, again, this operation is supported by all modern architectures [3][4].

Our implementation uses GCC atomic built-ins __sync_fetch_and_add, __sync_fetch_and_sub, and __sync_bool_compare_and_swap [5], requiring no hand-written assembly at all. Such intrinsics exist on other systems [6] and, as of C++11, in the language itself [7].

Synchronization Trigger

The synchronization of setting values is triggered when the reference buffer contains new consistent setting values. However, the synchronization can only happen if a real-time buffer is in the *Modifiable* state. Figure 2 shows that a real-time buffer is in the *Modifiable* state if it is not the *Current* buffer and it has no readers.

The buffer synchronizer thread (BST) ensures that a synchronization happens as soon as possible by waiting for the *Obsolete* buffer to go to *Modifiable*. Once a *Modifiable* buffer is available, the BST checks continuously whether a synchronization is needed. This is achieved in several steps. First, the BST deduces from the *Current* buffer pointer which buffer is *Obsolete*. The *Current* buffer pointer is guaranteed not to change since the BST is the only thread that can change it. To avoid a busy wait when the number of readers of the *Obsolete* buffer is greater than 0, we use a condition variable which is signalled by readers when a buffer's readers count reaches 0. A synchronization request is indicated to the BST via a boolean. Again, an expensive busy wait is avoided thanks to a condition variable which is signalled whenever the boolean is set to true.

CONCLUSION

This synchronization algorithm fulfils all the requirements:

- Real-time compliant (no memory allocation)
- Consistent setting values (real-time buffers cannot be modified while they have readers)
- Constant jitter between event and task execution thanks to a fixed flow of execution and an O(1) algorithm
- Settings made available as soon as possible

Nevertheless, implementing the solution is not free and two major drawbacks have to be mentioned. First, there is an obvious additional memory consumption; a third buffer is required compared to the simpler double-buffer approach. In modern systems, this is probably not an issue as the amount of settings is typically small compared to the amount of available RAM. The second drawback is the lack of control on the delay between the synchronization request and the actual availability of the new settings to the real-time tasks. As the synchronization cannot occur before all the *Obsolete* buffer's readers have completed their execution, a slow real-time task can delay the synchronization. Therefore, it is possible to have new real-time tasks executions not using the latest settings. In our case, this is not a problem as we consider the sending of new settings a slow and non-deterministic operation. If this limitation is incompatible with the system to be controlled, the possible evolution based on the usage of additional real-time buffers is detailed in the next section.

Possible Evolution

Our implementation uses two real-time buffers, but in practice, for busy real-time application with many tasks and frequent changes of settings, one can use as many buffers as the available memory allows. This would reduce the likelihood not to have any *Modifiable* buffer on synchronization request and therefore reduce significantly the delay between settings modification and settings availability. The management of *Modifiable* buffers would need to be adapted so that the first available *Modifiable* buffer can receive a copy of the newest setting values. In practice, two real-time buffers should be sufficient for most if not all real-time applications.

REFERENCES

[1] Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3A, Section 8.1.1, Guaranteed atomic operations; http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architecture-software-developer-vol-3a-part-1-manual.pdf

[2] ARM Synchronization Primitives, Section 1.2, Exclusive accesses; http://infocenter.arm.com/help/topic/com.arm.doc.dht0008a/DHT0008A_arm_synchronization_primitives.pdf

[3] Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3A, Section 8.1.2.2, Software Controlled Bus Locking; http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architecture-software-developer-vol-3a-part-1-manual.pdf

[4] ARM Synchronization Primitives, Section 1.3.3, Implementing a semaphore; http://infocenter.arm.com/help/topic/com.arm.doc.dht0008a/DHT0008A_arm_synchronization_primitives.pdf

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

[5] Atomic Builtins – Using the GNU Compiler Collection (GCC), http://gcc.gnu.org/onlinedocs/gcc-4.1.0/gcc/Atomic-Builtins.html
[6] InterlockedIncrement function (Windows), https://msdn.microsoft.com/en-us/library/windows/desktop/ms683614(v=vs.85).aspx
[7] std::atomic::fetch_add, C++ reference, http://en.cppreference.com/w/cpp/atomic/atomic/fetch_add

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

EPICS V4 EVALUATION FOR SNS NEUTRON DATA*

K.U. Kasemir, G.S. Guyotte, M.R.Pearson, ORNL, Oak Ridge, TN37831, USA

Abstract

Version 4 of the Experimental Physics and Industrial Control System (EPICS [1]) toolkit allows defining application-specific structured data types (pvData) and offers a network protocol for their efficient exchange (pvAccess). We evaluated V4 for the transport of neutron events from the detectors of the Spallation Neutron Source (SNS) to data acquisition and experiment monitoring systems. This includes the comparison of possible data structures, performance tests, and experience using V4 in production on a beam line.

MOTIVATION

On SNS beam lines, each neutron event consists of a pixel ID that identifies the location of the detector where a neutron was observed, and a time-of-flight measurement that describes when the neutron was detected relative to the most recent beam pulse. Depending on the beam line and its specific configuration, event rates can reach a few million events per seconds.

This neutron event information needs to be transferred from detectors to processing stages that provide users of the experiment with visual feedback, accumulate information that allows for the automation of the experiment, and finally stream the events into a data collection pipeline for long-term storage of the experiment data.

The original SNS beam line data acquisition software used a locally developed UDP/IP-based network protocol to transmit neutron event information [2]. The limited performance and reliability of this protocol necessitated an update of the overall data acquisition software [3].

EPICS V4

Based on a proven track record for control of the SNS accelerator, EPICS had been chosen as a toolkit for the beam line control system upgrade. While Channel Access [4], the original EPICS network protocol, has a well defined and functional set of data objects, this set is fixed to types suitable for describing a single data point, for example a temperature reading or voltage set point.

*This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan(http://energy.gov/downloads/doe-public-access-plan).

EPICS V4 [5] is an addition to the EPICS toolkit that introduces an alternative to the existing EPICS V3 data types and network protocol.

pvData

pvData is the EPICS V4 library for structured data. It can describe such data in an operating-system independent way, hold it in memory, and copy complete or partial data containers. The data can include time stamps, numeric values, enumerated data, text and alarm information. Data can also be assembled into arrays and structures.

Normative Types

pvData allows clients to define nearly arbitrary data structures, which is ideal for packaging site-specific information. At the same time it limits the interoperability of applications. *Normative Types* are a set of agreed-upon pvData types that all implementers of V4 applications are encouraged to support. All original EPICS V3 data types are described as Normative Types, allowing for an eventual transition from V3 to V4. In addition, data types like N-dimensional images or statistical samples that are often used in higher-level control system applications are available as Normative Types.

pvAccess

pvAccess is the V4 network protocol that allows for the exchange of pvData. It is conceptually similar to V3 Channel Access, using UDP/IP for channel name resolution, and then establishing a TCP/IP connection between pvAccess servers and clients to exchange data. It supports basic read and write access. A subscription mode efficiently updates clients on changes in the data by only transferring the modified structure elements. Finally, a combined write/read mode supports remote service calls by atomically sending parameters, awaiting the execution of the remote service, then returning the result.

Both pvData and pvAccess have been implemented in C++ and Java, with additional bindings for Python [6].

SNS NEUTRON DATA

SNS neutron data consists of a list of pixels and time-of-flights as already described, combined with a sequential pulse number and the proton charge of the accelerator pulse that generated these neutrons.

The following pvData structure would be a direct representation:

```
// Sequential pulse number
uint64   pulse
```

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

```
// Proton Charge
double   proton_charge
// Event array, each element is
// time-of-flight & pixel
struct
{
   uint32  time_of_flight
   uint32  pixel
} events[]
```

Each neutron event naturally combines the affected detector pixel and the time of flight when the event occurred. The above data structure always provides each such event as a tuple of { time_of_flight, pixel }.

Some consumers, however, require only a subset of this tuple. A time-of-flight histogram only needs to inspect the time_of_flight elements, and a spatial X/Y histogram only the pixel elements. With the above structure they need to subscribe to the "events" array and thus always receive both the time of flight and pixel information.

The following data structure holds the same information, but allows clients to subscribe to just the information of interest:

```
// Time stamp for everything in this
// structure.
// timeStamp.userTag holds
// sequential pulse number
time_t timestamp

// Proton Charge
NTScalar proton_charge
  double value

// Time-of-Flight values for N neutron events
NTScalarArray time_of_flight
   uint[]  value

// Pixel IDs for N neutron events
NTScalarArray pixel
   uint[]  value
```

With this optimized data structure, all producers and consumers agree that the "time_of_flight" and "pixel" arrays always contain the same number of elements, because corresponding array elements constitute one neutron event.

A tool that accumulates the X/Y histogram can now subscribe to just the "pixel" element, receiving only this data and thus reducing the network traffic. Tools that require the complete information can still subscribe to the whole pvData structure.

In addition, this updated structure packages the sequential pulse number into the 'user' element of the normative time stamp type, and bases the proton charge, time of flight and pixel elements on Normative Types, allowing for compatibility with generic V4 client tools.

pvData allowed us to package the events as either an array-of-structures or a structure-of-arrays, and we chose the latter to optimize network traffic for the various use cases. For certain detector types or operating modes, the above structure can be extended with additional data elements, for example to transmit internal detector counts, which are used during calibration.

PERFORMANCE TESTS

We implemented a V4 server that emits data of the above format with sequential pulse numbers as well as a V4 client that subscribes to this data, counting the received elements and specifically detecting missed pulse numbers [7].

Figure 1 shows network traffic results from executing the demonstration server and client on a 1 gigabit Ethernet link. The server was sending 100 updates per second, varying the number of events in each of these updates. When each packet contains 150000 events, 100 times a second, this would equate transmitting 15 million SNS neutron events per second. Up to about 15 million events per second, the measured network traffic scaled linearly. There is very little overhead on the expected network traffic based on the underlying data size, proving that pvAccess efficiently serializes the pvData.

As we increased the event rate beyond 15 million SNS neutron events per second, the network traffic asymptotically approaches 95 MB per second. The client starts to indicate lost pulse updates. CPU loads of the server and client were only about 30%, indicating that we reached the limit of TCP on 1GigE.

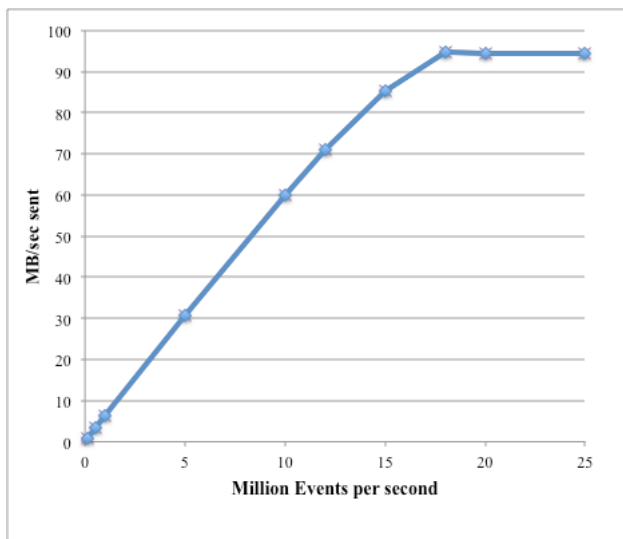


Figure 1: Network traffic on 1GigE network when sending various amounts of neutron events packaged into 100 updates per second.

On a 10GigE test setup, the simulated SNS neutron event rate could be increased to about 100 million events per second before reaching CPU load limits on the server.

GENERIC V4 TOOLS

EPICS V4 includes generic command line tools. The "pvinfo" command displays the IP address of the V4

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

server and the pvData structure of a V4 channel. This is useful for testing if a V4 server is online, and to check if the data contains the expected elements.

The "pvget" command can display the complete structure of a current value, or subscribe to selected elements. For example, the following command would show updates to the pixel array of the data sent by the V4 server, on a V4 channel named "neutrons", used in the performance tests:

```
pvget –m –r neutrons.pixel
```

USE OF V4 AT SNS BEAM LINES

As part of the update to the SNS detector control software to EPICS, the nED software [8] was developed to provide a pluggable framework for interfacing to the various detector configurations found at different beam lines. One nED module is a V4 server that publishes the SNS neutron data in the format we described.

There are two primary network clients for this data. One is a streaming data acquisition system that writes all neutron events to files for later analysis. The other is ADnED, an EPICS Area Detector driver which provides user displays and information for automation [9].

Ideas from the example server code [7] were used to create the V4 server in nED. Similarly, the example server was useful to create test data during the development of ADnED, allowing independent development and testing of these tools.

CONCLUSION

SNS neutron data can be packaged in pvData. By comparing different packaging options, we were able to optimize the network traffic based on the expected types of network clients.

The performance of pvAccess easily meets our requirement of about 10M events/sec on 1GigE and exceeds it on 10GigE.

While the original SNS beam line data acquisition software was limited to Microsoft Visual C++ on Windows, the EPICS V4 libraries for pvData and pvAccess are available on Linux, Mac OS and Windows, for C++, Java and Python. This allowed us to implement nED and ADnED in C++ on Linux to obtain the required performance, while test and calibration tools are often implemented in Python, offering more flexibility.

While the original SNS beam line neutron event data server and clients had no additional network test tools, we can now use the generic EPICS V4 command line tools to test if a server is online, or to monitor the data on the network.

At the time of writing, the SNS beam lines USANS, CORELLI, HYSPEC, VISION and SEQUOIA have been updated to use V4 pvData and pvAccess, along with nED and ADnED, for the critical first stages of neutron data transfer. Operation has been very reliable, especially considering that pvData and pvAccess are new developments. The SNS is the first facility to utilize these technologies in production systems on operating beam lines.

ACKNOWLEDGMENT

We thank Matej Sekoranja, Marty Kraimer and David Hickin for their assistance while learning about V4, their help when implementing the performance test code, and their fast response whenever we found problems in pvData and pvAccess.

REFERENCES

[1] http://www.aps.anl.gov/epics/
[2] R.E. Riedel, "Overview of Data Acquisition at the SNS", NOBUGS 2004, http://lns00.psi.ch/nobugs2004/papers/paper00055.pdf
[3] S.M.Hartman, "SNS Instrument Data Acquisition And Controls", ICALEPCS 2013, San Francisco, CA, USA.
[4] http://www.aps.anl.gov/epics/docs/CAproto.html
[5] L.R. Dalesio et al, "EPICS V4 Expands Support to Physics Application, Data Acquisition, and Data Analysis", ICALEPCS 2011, Grenoble, France.
[6] http://epics-pvdata.sourceforge.net
[7] K. Kasemir, EPICS V4 Example Server and Client for SNS Neutron Data, https://github.com/kasemir/EPICSV4Sandbox
[8] G.Guyotte, "nED – EPICS-based Neutron Data Acquisition and Detector Control Software", EPICS Meeting, FRIB, MSU, Lansing, MI, 2015.
[9] M. Pearson, "ADnED – V4 Neutron Event Data in areaDetector", EPICS Meeting, FRIB, MSU, Lansing, MI, 2015. https://github.com/areaDetector/ADnED

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CCLIBS: THE CERN POWER CONVERTER CONTROL LIBRARIES

Q. King, K. Lebioda, M. Magrans De Abril, M. Martino, R. Murillo, A. Nicoletti,
CERN, Geneva, Switzerland

Abstract

Accurate control of power converters is a vital activity in large physics projects. Several different control scenarios may coexist, including regulation of a circuit's voltage, current, or field strength within a magnet. Depending on the type of facility, a circuit's reference value may be changed asynchronously or synchronously with other circuits. Synchronous changes may be on demand or under the control of a cyclic timing system. In other cases, the reference may be calculated in real-time by an outer regulation loop of some other quantity, such as the tune of the beam in a synchrotron. The power stage may be unipolar or bipolar in voltage and current. If it is unipolar in current, it may be used with a polarity switch. Depending on the design, the power stage may be controlled by a firing angle or PWM duty-cycle reference, or a voltage or current reference. All these cases are supported by the CERN Converter Control Libraries (CCLIBS). These open-source C libraries include advanced reference generation and regulation algorithms. This paper introduces the libraries and reviews their origins, current status and future.

INTRODUCTION

The CERN converter control libraries are a collection of libraries written in C, available under the GNU Lesser General Public License from https://github.com/cclibs.

The libraries were started in 2010 by extracting the measurement calibration, function generation and current regulation code from the operational software used to control the power converters in the CERN LHC. This software was originally deployed in the LHC magnet test facility in 2000 [1,2], so the libraries built on ten years of experience with the high-precision control of current in magnet circuits [3].

Figure 1 shows an overview of the architecture of a power converter controller based on CCLIBS. The long-term objective is to move all the re-usable generic software components into CCLIBS and to leave the application with just the hardware specific components.

In 2012, the new libraries [4] were deployed into operation on two different hardware platforms [5] and a third platform was added the following year.

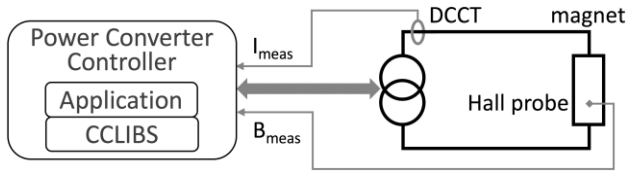


Figure 1: The role of CCLIBS in converter control

In 2013, a second version of the libraries was started with the objective to support regulation of magnetic field measured by a hall probe. Hall probes have more noise than current transducers (DCCTs), so more advanced anti-aliasing filtering is needed. This in turn requires more advanced algorithms for the computation of the coefficients for the RST regulator, to accommodate the additional delay from the filters.

Then in 2015, a new reference manager library, libref, was created to support the three different operating modes used with magnet circuits at CERN. This is well advanced and will be deployed into operation in 2016, along with two more libraries, libsig and liblog. Several other features will also be added, including voltage regulation for thyristor converters.

THE LIBRARIES

Taken together, the Converter Control Libraries have too many features to cover them all in this short paper. Furthermore, the first versions of libfg and libreg were described in [4] so this paper will report some of the improvements made to libfg and libreg as well as the main features of the new libraries and test programs.

Libcal: Calibration

The calibration library was part of CCLIBS from the start. It provides a three-point calibration scheme for ADCs and DCCTs, with second order temperature compensation. This accommodates an offset and separate positive and negative gain errors. The library also supports three-point DAC calibration and a simple scaling for hall probes and voltage probes (i.e. no offset and just one gain that applies to both positive and negative measurement values). The library will be optimised for better performance in 2016.

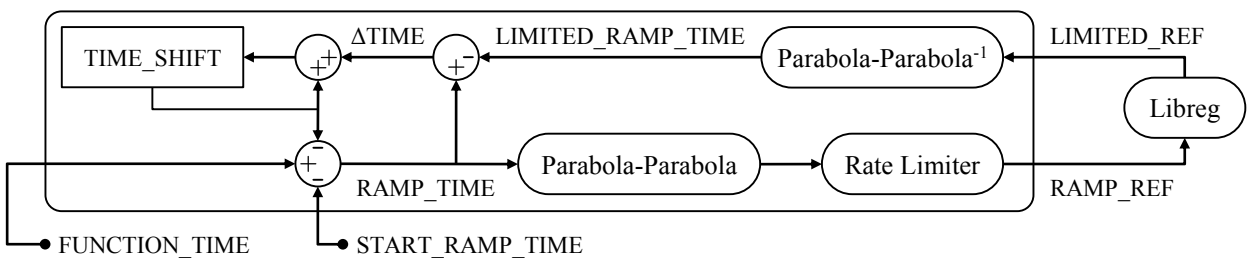


Figure 2: The RAMP function algorithm

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Libfg: Function Generation

The new version of libfg has a simplified API and now supports twelve reference functions, including a new RAMP function with special powers. It can start with a non-zero rate of change, which allows it to take over smoothly from a running function. The start and end of a RAMP are always smooth because it is based on two parabolic segments. What makes RAMP uniquely powerful is the way it can respond to the clipping of the reference, either by RAMP's rate limiter or by libreg.

Figure 2 shows how the LIMITED_REF value returned from libreg is used to calculate the TIME_SHIFT variable, which is then subtracted from the FUNCTION_TIME to generate the RAMP_TIME. It uses the classic formula to invert the quadratic equations of the parabolas:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \rightarrow LIMITED_RAMP_TIME$$

The difference between LIMITED_RAMP_TIME and the actual RAMP_TIME is accumulated as TIME_SHIFT. In this way the parabolic functions are distorted by effectively slowing down time while the reference is being limited.

Libreg: Regulation

The new version of the regulation library builds on the original and has many new features and improvements; the API has been simplified to ease integration, new algorithms have been added to calculate and validate the RST coefficients, a two-stage sliding average anti-aliasing filter is available for measurement signals and field regulation is now supported as well as dynamic switching between current, field and voltage regulation modes.

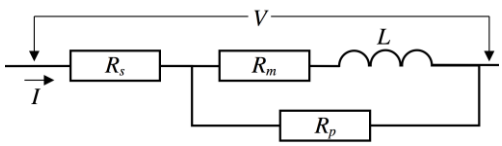


Figure 3: First order magnet circuit supported by libreg

The new algorithms for calculating the RST coefficients implement an internally stable pseudo-deadbeat regulator for the first-order magnet circuit shown in Fig. 3. They can accommodate up to 2.4 periods of loop delay (compared to only 0.4 periods in the first version of libreg). As before, the RST coefficients can also be calculated externally and in both cases libreg checks the stability of the regulator using Jury's test. For an internally generated regulator, the modulus margin is calculated to check the vulnerability of the regulator to errors in the model.

The new library also implements an open-loop controller for the first-order magnet circuit shown in Fig. 3. The circuit current $I(t)$ respects the differential equation:

$$(R_p + R_m)V(t) + L\frac{dV}{dt}(t) =$$
$$(R_p + R_s)L\frac{dI}{dt}(t) + \left(R_s(R_p + R_m) + R_pR_m\right)I(t)$$

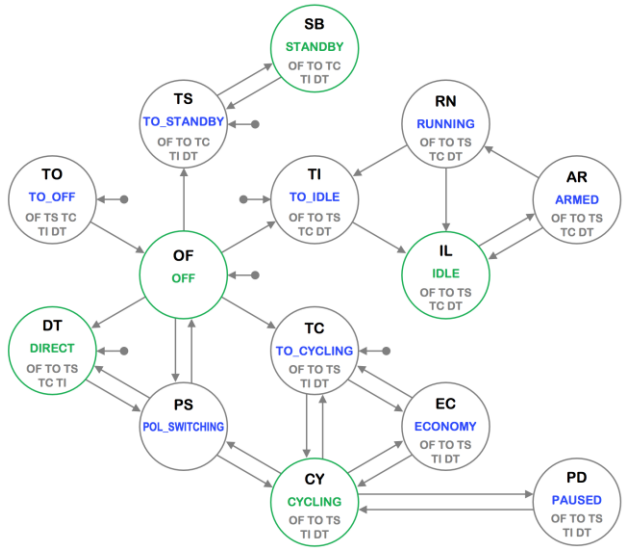


Figure 4: The Reference State Machine

This is implemented in libreg with a forward Euler approximation. It can run forwards (ref→actuation) or backwards (actuation→ref) in a similar way to the closed-loop RST algorithm. Libreg will automatically use the open-loop controller for one and two-quadrant power converters when the current (or field) is below the closed-loop threshold. This is the level below which the voltage source gain is non-linear and closed-loop regulation becomes unreliable. This feature makes starting and stopping of one and two-quadrant converters very easy to implement in the new reference manager library.

Another new feature of libreg helps with the reality that there is rarely enough time to commission accelerator circuits in depth because of the pressure to get an accelerator into production as quickly as possible. With a rapid cycling machine, sometimes the operators can allocate one cycle per super-cycle without beam, to allow the continued refinement of the regulation parameters in parallel with operation. On these cycles, libreg can switch automatically to use a set of test RST coefficients.

Libref: Reference Management

The reference manager library uses libfg and libreg to provide a full range of advanced features to the converter control application. Its behaviour is controlled by the Reference State machine, which is illustrated in Fig. 4. This supports three operating modes, associated with the states DIRECT, CYCLING and IDLE (see below).

Libref uses the new RAMP function together with the new open-loop regulator in libreg to implement the smooth starting and stopping of one and two-quadrant converters. This is an elegant solution that makes the handling of one and two-quadrant converters equivalent to that of four-quadrant converters.

Libref manages the switching between the three regulation modes supported by libreg (field, current and voltage). It allows the application to prepare to run a reference function for a given regulation mode. This is known as "arming" the function.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Libref also implements polarity switch management. This allows a low-cost unipolar converter to be used with a bipolar circuit. Switching can be on-demand, or automatic in DIRECT and CYCLING states, based on the polarity of the reference function.

THE TEST PROGRAMS

One of the motivations for the creation of the converter controls libraries was to make it possible to test these critical software components outside of the embedded systems in which they run operationally. This is because debugging is much easier on a desktop computer, where resources are abundant. A growing portfolio of test scripts is run automatically as part of the group's continuous integration system, both nightly and following commits to CERN's git repository (this is not github).

CCLIBS currently has two different test programs, cctest and ccrt, but by 2017, ccrt will be able to cover all the testing requirements for CCLIBS and support for cctest will be dropped.

Single-Threaded Test Program: cctest

Cctest, was created in 2010 at the same time as the first libraries. It reads commands interactively from standard input or from files when running in batch mode. It only tests libfg and libreg and it runs simulations of a power converter with an inductive load for a specified series of reference functions. A large number of digital and analog signals are logged and then written to files in HTML, JSON or CSV formats. Examples can be seen on the project web site: https://cern.ch/cclibs.

Cctest, like the libraries, is written in standard C and has very few dependencies. It compiles and runs under Linux, MacOS and MinGW on Windows.

Multi-Threaded Test Program: ccrt

In 2015, libref was created to manage reference generation. The library is designed to work with a real-time thread (or interrupt) and a background thread. This requires a multi-threaded test program, so ccrt was created by forking cctest.

Ccrt tests libref, libfg and libreg, and it will be extended in 2016 to test libsig, libcal and liblog. It implements some important features that are missing from cctest, including parameter assertions (equals, not equals, less than, greater than and approximately equals), as well as synchronisation commands to sleep, wait for a reference state or wait for a function to complete. This allows detailed use cases to be tested rigorously without the need for a human to review simulation results graphically (as is the case with cctest).

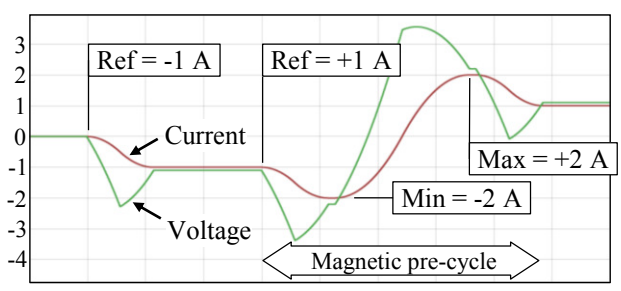


Figure 5: Example of an UPMINMAX pre-cycle

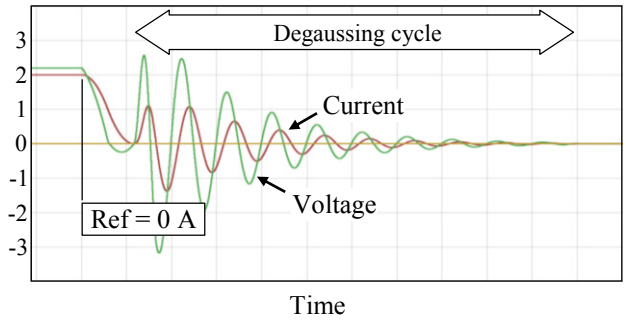


Figure 6: Example of a degaussing cycle

There are still a few features of libfg and libreg that can only be tested in cctest, but in 2016, these missing features will be added to ccrt so that in 2017, support for cctest will be dropped. Ccrt uses Posix threads and currently runs only under Linux, but it will be adapted to run under MacOS. In principle, it should then also work under MSYS2/MinGW-w64 on Windows.

DIRECT, CYCLING AND IDLE

Libref will cover all the operational use cases of power converter control present in CERN's accelerators. These divide into three operating modes that are managed by the DIRECT, CYCLING and IDLE states.

DIRECT for DC Circuits

The DIRECT state was added to support circuits which operate with a stable DC reference (which defines the field, current or voltage according to the regulation mode). The reference value may need to be modified from time to time, but the form and synchronisation of the ramp to the new reference value is not important to the accelerator operators. So libref is responsible for this change and it uses the RAMP function. The acceleration, deceleration and linear rate limit are taken from default parameters. If the ramp speed is important, the rate limiter can be disabled and the ramp will be limited only by the available voltage.

Libref implements three advanced features in DIRECT state to cover particular use cases at CERN: magnetic pre-cycling, automatic degaussing and automatic polarity switch control.

Magnetic pre-cycling improves the reproducibility of the field in a magnet by always approaching the new reference following the same magnetisation curve. Two options are supported: DOWNMAXMIN and UPMINMAX.

With UPMINMAX, the reference will always approach the new value from above. Thus, when going down, no pre-cycle is needed, but when going up, the reference must first visit the minimum and maximum values. In each case, the reference will pause for a time chosen to allow the eddy currents to decay. This is illustrated in Fig. 5, where the initial ramp is from 0 A to -1 A. This is followed by a request to go up to +1 A, which triggers the MINMAX pre-

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

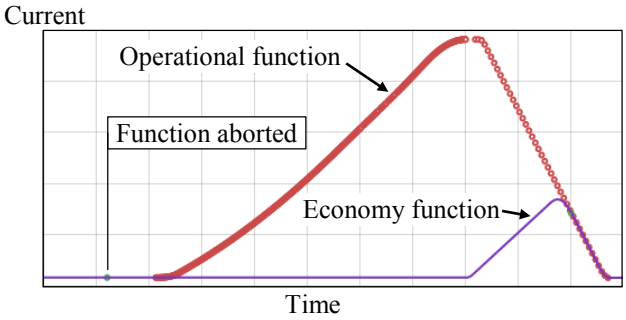


Figure 7: A dynamic economy cycle from the CERN SPS

cycle. The behaviour of DOWNMAXMIN is simply the mirror image of UPMINMAX.

If the remanent magnet field must be minimised, a degaussing cycle can be enabled. In this case, a ramp to zero amps (or zero gauss if regulating the field) will trigger a degaussing waveform. This is an exponentially decreasing sine wave, windowed to have a smooth start and end. The amplitude and period can be selected according to the needs of the magnet. Figure 6 shows an example of a degaussing waveform, triggered by setting the current reference to zero while at 2A.

CYCLING for Rapid Cycling Machines

The accelerators that collectively inject particles into the LHC all operate periodically under the control of a sophisticated timing system. Super-cycles typically lasting thirty seconds are assembled from cycles with particular uses. The identity of the next cycle (the "cycle selector") is broadcast in real-time over a dedicated timing network to all the equipment in the accelerator, which must be ready to execute whichever cycle is specified. Libref supports this by allowing the application to arm a different function and regulation mode for every cycle selector. The timing events must be passed to the library, giving it the cycle selector and absolute start time of the next cycle. The library then takes care of executing the selected function at the correct time.

Function generation can be suspended and resumed by timing events. When suspended the state machine will be in the PAUSED state. The active function can be replaced by another function while paused and if the reference changes, the PAUSED state will use a RAMP function to change smoothly to the new value.

If a polarity switch is in use with a unipolar converter, then the CYCLING state is able to switch polarity between cycles based on the polarity of the next reference function (which must be all positive or all negative).

In large non-super-conducting circuits, the resistive losses can be significant. If beam for a given cycle selector is not available, then it is desirable to suppress the execution of the function in real-time. However, this disturbs the magnetic field of the following cycle because the eddy currents that would normally be induced by the ramp down are not present. This is addressed by the dynamic economy feature. It allows a running function to be aborted if there is no beam. The reference ramps down to the minimum RMS level and then ramps up to join the end of the original function smoothly at a user-specified time. This can be at as little as 25% of the maximum reference value, but the remaining ramp down excites the eddy currents that are expected at the start of the next cycle. The result is a major reduction in resistive losses. This behaviour is provided in the ECONOMY state. Figure 7 shows an example of a dynamic economy cycle from the CERN SPS accelerator.

IDLE for Single-Use Reference Functions

The LHC cycles aperiodically with every ramp triggered by the operators. Although a ramp in energy might be repeated from cycle to cycle, when this is converted into a ramp in current for a given super-conducting circuit, it is always different because it depends upon the powering history of that circuit. So functions are only ever used once and are then disarmed automatically.

This functionality is provided by the states IDLE, ARMED and RUNNING. A new function can only be armed in the IDLE state, and must begin from the actual reference. Once a function has been armed, the state changes to ARMED and the function can be started synchronously with a timing event. The function will be generated in the RUNNING state, and upon completion, it is disarmed and the state machine returns to IDLE.

The running function can be aborted before the end with another timing event, changing the state to TO_IDLE. This will smoothly decelerate and return parabolically to the reference value that was latched at the moment of the abort event. The state then returns to IDLE.

CONCLUSION

The CERN converter controls libraries build on more than fifteen years of experience with accurately regulating the current in magnet circuits. They will form the heart of CERN's converter controls software for years to come and are freely available under the GNU Lesser General Public License.

REFERENCES

[1] Q. King et al, "The All-Digital approach to LHC power converter current control", ICALEPCS'01, THBT004

[2] F. Bordry, H. Thiesen, "RST Digital Algorithm for controlling the LHC magnet current", CERN/LHC Project report 258

[3] Q. King, "Status of the LHC power converter controls", ICALEPCS'09, MOB003

[4] Q. King et al., "Function generation and regulation libraries and their application to the control of the new main power converter (POPS) at the CERN CPS", ICALEPCS'11, WEPMN008

[5] D. Calcoen, Q. King, P. Semanaz, "Evolution of the CERN power converter function generator/controller for operation in fast cycling accelerators", ICALEPCS'11, WEPMN026

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

MULTI-HOST MESSAGE ROUTING IN MADOCA II

T. Matsumoto#, Y. Furukawa, K. Okada, JASRI/SPring-8, Hyogo, 679-5198, Japan

Abstract

MADOCA II is the next generation of the Message and Database-oriented Control Architecture (MADOCA) that was implemented in the control systems of the SPring-8 and the SPring-8 Angstrom Compact Free Electron Laser (SACLA) data acquisition (DAQ) systems in 2013 [1]. In 2014, SACLA was equipped with a second beamline (BL2) to increase the capacity of experiments. Messaging over a firewall network is required for beamline control, and an ad hoc socket application was used for this because MADOCA II does not have the requisite functionality. However, this caused problems and increased the complexity of the control system. In this paper, we propose multi-host message routing in MADOCA II to get rid of the socket application In SACLA DAQ systems, multi-host message routing is used for sophisticated access control. We introduce a master server that mediates control messages between operator workstations and equipment management servers. Since access control can be centralized to the master server, reliable operation was achieved by avoiding the effects of the accidental modification of DAQ settings by end users. We implemented multi-host message routing in the SACLA DAQ system in September 2014, and it has been stably operating since then.

INTRODUCTION

As the next generation of the Message and Database-oriented Control Architecture (MADOCA), MADOCA II was successfully implemented to control systems for SPring-8 and the SPring-8 Angstrom Compact Free Electron Laser (SACLA) data acquisition (DAQ) system in 2013, as reported at the last meeting of the International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS) [1]. Like its predecessor, MADOCA II is based on a message-oriented control scheme. However, messaging in this architecture has been made more flexible by replacing the messaging protocol with the ZeroMQ socket library [2] and reconfiguring the messaging scheme. In MADOCA II, data of varying length, such as image data can be attached to messages, and control applications can be used in the Windows operating system environment. Such features were utilized in several control applications in SPring-8 [3] [4].

However, MADOCA II needed to be extended when a second beamline (BL2) was equipped to the SACLA system in 2014. There are several network layers in the control system for experimental users, data acquisition, and the beamline equipment. We sometimes need control over the firewall network, which requires messaging via an intermediate host, as shown in Figure 1.

MADOCA II is designed to facilitate communication between hosts. To implement message routing among more than two hosts, we used an ad hoc socket application to mediate messages. However, we found that the application stalled at times. This was because the application processes messages sequentially, and is affected when a messaging procedure takes a while. Moreover, the application renders the control procedure complex, making it difficult to understand when and why control troubles occurred. To solve these problems, we developed multi-host message routing in MADOCA II.

In the following section, first, we provide an account of the implementation of multi-host routing in MADOCA II and its application to the SACLA DAQ system. We conclude with a summary of our findings.

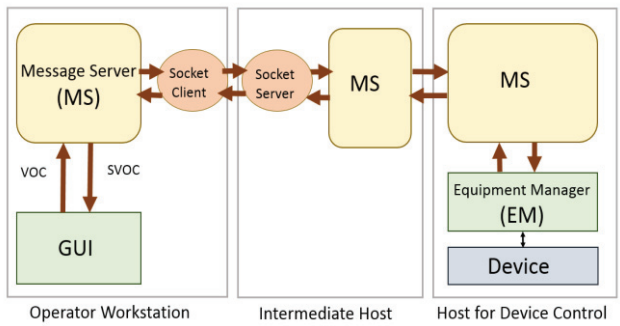


Figure 1: Messaging via an intermediate host in MADOCA II. An ad hoc socket application is used to mediate a message.

IMPLEMENTATION OF MULTI-HOST MESSAGE ROUTING IN MADOCA II

In this section, we describe the implementation of multi-host message routing in MADOCA II. We first explain the messaging procedure between two hosts, followed by an account of how the scheme is extended to multi-host message routing. Some technical issues, such as access control, prevention of messaging loops, and RTT (round-trip time) during messaging, are also addressed.

Message Routing Between Hosts

Figure 2 shows an example of messaging between hosts. A message command is exchanged between a graphical user interface (GUI) on host-b and an equipment manager (EM) on host-a through a Message Server (MS) in each host. In MADOCA II, messages are based on text and composed of a character string in subject/verb/object/complement (S/V/O/C) syntax. In case of messages from a GUI, “put” (V) represents the

#matumot@spring8.or.jp

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

control action, "objectA" represents the equipment to be controlled, and "start" (C) represents the value to describe the content of the action. The subject (S) is composed of process number, application name, account name, and host name, and is determined by the control framework as "123_matumot_testgui_host-b." The response is obtained from an EM and read as "objectA/put/123_matumot_testgui_host-b/ok" (in the response, S and O are interchanged).

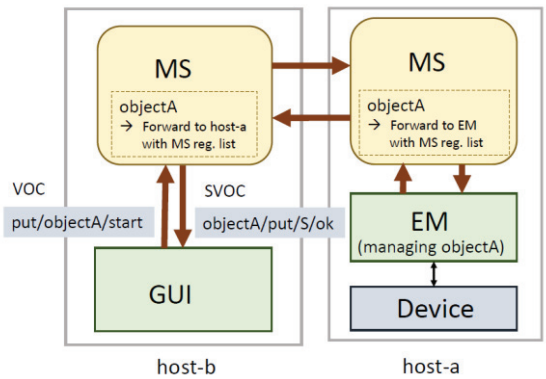


Figure 2: An example of messaging between hosts in MADOCA II. A GUI sends a VOC command, and the response is returned in SVOC format from an EM (equipment manager). Object information registered in an MS (Message Server) is used for message routing.

To implement message routing, we use the object name (O) to determine the route to an EM and the subject name (S) to identify the route back for the message to the GUI because S contains information regarding the GUI, such as application name. In an MS, the following information is registered through related sockets to handle message routing among applications:

1. S defined in the application, such as a GUI or an EM in the same host.
2. Hostnames for connecting or connected MS(s).
3. Object names tagged with the S of an EM in the same host. Object information is also registered to a connected MS.

When an MS on host-b is connected to one on host-a, "objectA" is registered in an MS on host-b. Therefore, a message can be transferred from a GUI to an EM by referring to the registered information in the MSs.

Message Routing Through Multiple Hosts

In MADOCA II, object information is used to determine the route of the message. However, this information is only shared directly with the connected MSs. Therefore, multi-host messaging involving more than two hosts is not possible in MADOCA II. To overcome this limitation, we developed an extension in MADOCA II to forward a message to other hosts even if the object in the SVOC message is not registered in an MS.

Table 1: An example of the Access Control List (ACL) in MS

Object name	Management method	account@hostname
sr_ms_serve	MS	*@*
sr_ms_manage	MS	control@localhost
object_fwd1	MS:host1	*@*
object_fwd2	MS:host1,host2	*@*
object_ip1	MS	*@172.14.12.15
object_ip2	MS	*@172.14.12.0/24

The message forwarding scheme was implemented by extending the Access Control List (ACL) used to set the MS for each host. Table 1 shows an example of the ACL. ACL is composed of three columns. The first column, "object name," is the target object name for access control. Using this column, objects with forward-matched names can be specified. The second column, "management method," defines the method for message forwarding. The management method is set to "MS" if the object is managed using the normal method. When a message is forwarded to host1, we define the management method as "MS: host1" even if the relevant object is not registered in the relevant MS.

We can also define the management method as "MS: host1, host2." In this case, the message is forwarded to host2 when message forwarding to host1 fails. Thus, we can set a priority for message forwarding.

The third column, "account@hostname," defines an account name and a hostname in access control. The details are described in the next section

Figure 3 shows an example of multi-host message routing using our proposed extension in MADOCA II. A GUI sends a message, "get/objectA/start," and an EM responds by finding an intermediate host. The MSs have connections between host-b and host-m, and host-m and host-a. To implement multi-host message routing, we define the message forwarding rule by setting the ACL to host-b. The management method of the ACL is set to "MS: host-m" for "objectA." We can then forward the message from host-b to host-m. Following this, the message is routed to an EM because the intermediate host-m is connected to host-b through an MS, where the EM is located, and information regarding "objectA" has already been transferred from an MS on host-a to one on host-m.

To return the message from the EM to the GUI, however, only the subject name (S) in the message is not sufficient because we also need to know the message route. Therefore, we attach the message route (msg_route) while forwarding the message. When the message is forwarded from host-b to host-m, we set msg_route as "host-b: host-m." The msg_route is then updated to "host-b: host-m: host-a" when the message is forwarded from host-m to host-a. With msg_route, we identify the message route and can use it to return the message from an EM to a GUI.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

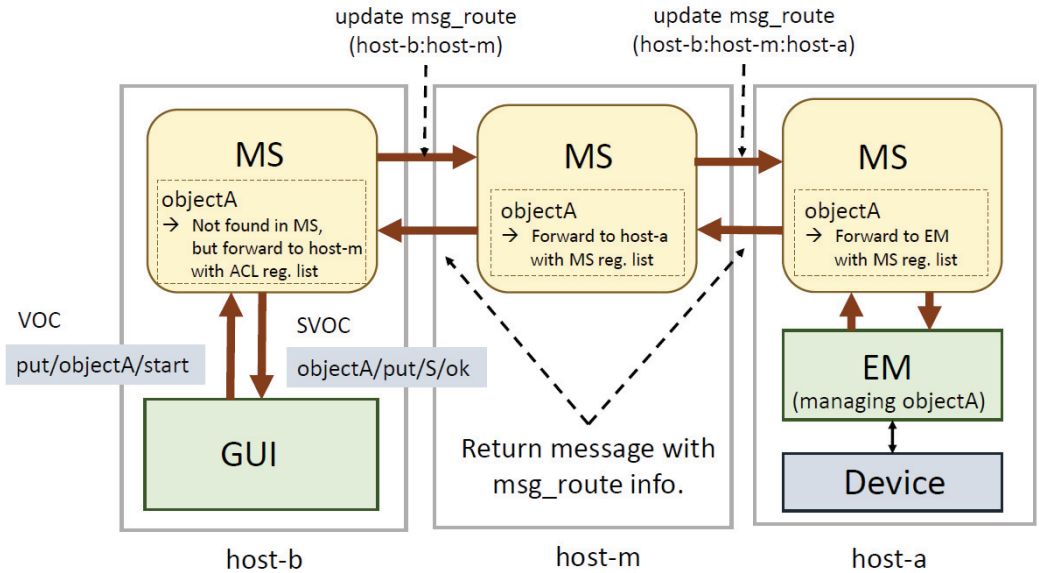


Figure 3: An example of messaging with multi-host routing in MADOCA II. A message is forwarded from host-b to host-m by setting the forwarding rule in the ACL. The message route is used to return the message to a GUI.

Figure 3 shows multi-host message routing through three hosts. In principle, the messaging scheme can be applied to multi-host message routing through more hosts.

Access Control

Access control in MADOCA II is implemented by referring to column "account@hostname" in ACL. Here, we define the account name and the hostname to allow control access. In Table 1, the object "sr_ms_manage" is allowed access with a control account to a local host only because the object is used for management purposes. However, there is no restriction on access to object "sr_ms_serve."

In the SACLA DAQ system, we need to apply control access at specific network segments because some end user PCs are connected through the Dynamic Host Configuration Protocol (DHCP) and we cannot identify these hostnames. Therefore, the ACL is extended to manage access control using IP addresses with subnet masks, as shown in Table 1. If the requirements of access control in the ACL are not satisfied, a message with un-authorized error is returned.

Preventing the Messaging Loop

In multi-host message routing, messaging loops should be avoided because they increase network traffic use and cause control problems. To prevent message loops, we use msg_route attached to the message while forwarding the message. The messaging loop can be detected by checking for the duplication of the hostname in msg_route. If a message loop is detected, an error message is returned.

RTT

In Multi-host message routing, we expect longer RTTs between a GUI and an EM because the messages are mediated by hosts.

We measured the RTT with varying configurations on the hosts, as shown in Table 2. When we performed message routing through three hosts, RTT of approximately 3 ms were observed. RTT delays of approximately 1 ms were observed in comparison with the case with message routing involving two hosts.

In the SACLA DAQ system, we found that the delay in RTT was not a problem because we used the messaging for slow control.

Table 2: RTT of a messaging between a GUI and an EM with several configurations. We used an EM, which simply returns a clock value. The test was performed using PCs with Intel Core i5-425U processors and a 1 GbE network switch under a local network. RTTs were averaged over 10,000 samples.

Case	RTT
A GUI and an EM in the same host	0.40 ms
A GUI and an EM in different hosts	1.89 ms
A GUI and an EM in different hosts with an intermediate host.	2.75 ms

APPLICATION OF MULTI-HOST MESSAGE ROUTING TO SACLA DAQ SYSTEM

In the SACLA DAQ system, shot-by-shot information synchronized with an X-FEL (X-ray Free Electron Laser) beam was stored at 60 Hz at the maximum repetition rate [5]. The challenge for multi-beamline operation was handling massive image data from experiments in each

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

beamline. To enhance the data handling capacity, the data stream and the network layer were upgraded to process separately [6].

We also needed reliable control in the face of complexity in multi-beamline operation. To satisfy these requirements, we redesigned the control architecture as shown in Figure 4.

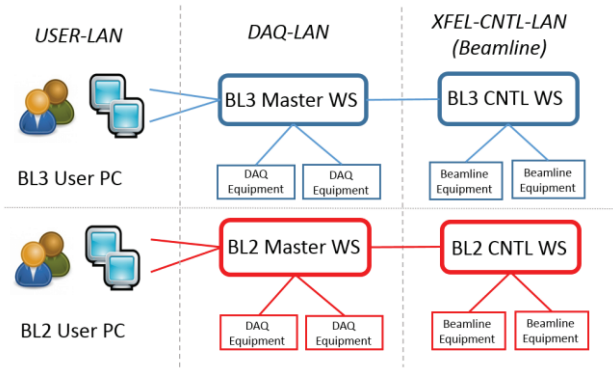


Figure 4: Control architecture of multi-beamline operation in the SACLA DAQ system. Access control was centralized through BL-master WSs and secured with settings in VLAN.

We split the network into three layers: DAQ-LAN was the network that controlled the DAQ equipment, XFEL-CNTL-LAN controlled beamline equipment, and USER-LAN controlled experimental end users as well as the equipment for DAQ and the beamlines. Direct access to XFEL-CNTL-LAN from USER-LAN was forbidden. We implemented multi-host message routing for the controls between USR-LAN and XFEL-CNTL-LAN.

For messaging control, a BL-master workstation (WS) was introduced to mediate all commands between user PCs and applications for equipment management. The user-PC could control all equipment in DAQ and the beamline simply by connecting to an MS in the BL-master WS. By passing BL-master WS in the messaging, we could monitor the message log for user PCs thorough the BL-master WS. We were also able to centralize the access control settings at the BL-master WS, hence avoiding the influence of missed configurations by experimental end users. As an example, it was possible to limit the use of some motor axes to facility usage. In USER-LAN, PCs were available for the experiments. We also forbade access to beamline equipment for user PCs through the BL-master WS. Such access control can be dynamically updated by modifying the ACL setting of the BL-master WS according to the experimental conditions. Therefore, access controls can be more flexibly adjusted than settings in the firewalls of networks.

We implemented multi-host message routing on multi-beamline operations in the SACLA DAQ in September 2014. In general, there were two user operator PCs in each experimental hatch, for a total of six. The operation thus far has been limited to slow controls, and no major problem has been reported during the one-year multi-beamline operation.

In 2015, an additional beamline (BL1) was added to SACLA. The adoption was effected by simply scaling up using the same control architecture.

SUMMARY

In this paper, we proposed multi-host message routing for MADOCA II and its application to the SACLA DAQ system for sophisticated access control under multi-beamline operation. The multi-host message routing was implemented by forwarding specific messages to other hosts by setting the ACL. With multi-host messaging, we can achieve flexible access control by centralizing access controls in the BL-master WS. The control scheme was successfully implemented to the SACLA DAQ system and has operated stably for a year. The same control scheme was applied to a new beamline (BL1) in SACLA. We also plan to adapt multi-host message routing to the SPring-8 beamline, where access control over the firewall network layer is required.

REFERENCES

[1] T. Matsumoto et al., "Next-generation MADOCA for SPring-8 control framework," Proceedings of ICALEPCS 2013, San Francisco, USA, (2013) p. 944.

[2] http://zeromq.org/

[3] A. Kiyomichi et al., "Development of MicroTCA-based Image Processing System at SPring-8," Proceedings of ICALEPCS 2013, San Francisco, California, USA, (2013) p. 78.

[4] Y. Furukawa et al., "MADOCA II Interface for LabVIEW," Proceedings of ICALEPCS 2013, San Francisco, California, USA, (2013) p. 410.

[5] M. Yamaga et al., "Control and data acquisition system for X-Ray free-electron laser experiments at SACLA," IEEE Nuclear Science Symposium and Medical Imaging Conference 2012 Record (NSS/MIC), Anaheim, USA. (2012).

[6] K. Okada et al., "Upgrade of SACLA DAQ system adapts to multi-beamline operation," PCaPAC 2014, Karlsruhe, Germany, (2014) p. 22.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

FLOP: CUSTOMIZING YOCTO PROJECT FOR MVMExxxx POWERPC AND BEAGLEBONE ARM

L. Pivetta, A.I. Bogani, R. Passuello
Elettra Sincrotrone Trieste, Trieste, Italy

Abstract

During the last fifteen years several PowerPC-based VME single board computers, belonging to the MVMExxxx family, have been used for the control system front-end computers at Elettra Sincrotrone Trieste. Moreover, a low cost embedded board has been recently adopted to fulfill the control requirements of distributed instrumentation. These facts lead to the necessity of managing several releases of the operating system, kernel and libraries, and finally to the decision of adopting a comprehensive unified approach based on a common codebase: the Yocto Project. Based on Yocto Project, a control system oriented GNU/Linux distribution called Flop has been created. The complete management of the software chain, the ease of upgrading or downgrading complete systems, the centralized management and the platform-independent deployment of the user software are the main features of Flop.

INTRODUCTION

Since the last decades, technology evolution and component obsolescence are driving a renewal that involves also particle accelerator control system platforms. Even industrial grade products well-known for the long lifetime support, such as VME, suffer these issues. Moreover, the evolution of application software, especially when dealing with new compiler features or when requiring new system library functionalities, often involves an update of the operating system software. Several models of PowerPC-based VME single board computers are in use at Elettra, running different distributions/releases of the GNU/Linux operating system [1–3]. Furthermore, the BeagleBone, a low cost embedded board based on the ARM microprocessor, has been recently introduced, as well as a low-power high-performance INTEL Bay Trail Celeron based Soc platform to interface a specific instrument via USB.

To address all these platforms in an effective and straightforward way, ensuring the consistency of the operating system, the device drivers and the application software, a comprehensive approach is desirable.

GNU/LINUX

The usual approach to GNU/Linux consists in the installation of one of the many available distributions. Yet, most of them do not support all the required architectures and instruction sets and, very often, rely on a generic instruction set to achieve improved compatibility at the expense of the performance.

Using distinct distributions for different architectures, which is the current scenario at Elettra, entails a number of possible issues, among which different releases of the system libraries, of the system initialization, e.g System V init rather than upstart or systemd, and of the filesystem hierarchy. Furthermore, diverse bugfix policy is often in place, making the system and platform adimistration quite onerous in the medium/long term.

Sticking to a specific GNU/Linux distribution release, conversely, exposes to the risk of outdated software, especially concerning the kernel, the system libraries and the compiler. Also, recent hardware is usually poorly supported, or even unsupported, by old distributions without a big effort in back-porting.

THE YOCTO PROJECT

The Yocto Project [4] is an open source collaboration, aimed at providing templates, tools and methods to create custom GNU/Linux based systems for embedded hardware, regardless of the architecture. Established in 2010, it involves many hardware manufacturers, open-source operating system vendors and electronics companies. The Yocto Project provides a complete embedded Linux development environment, with tools, metadata and documentation for free, including core system component recipes provided by the OpenEmbedded project. Specific platform support is provided by the Board Support Package (BSP), for which a standard format has been developed. In addition to the very effective command line tools, the Yocto Project provides an Eclipse IDE plugin and the Hob graphical user interface.

FLOP

Starting from the Yocto Project release 1.8, a control system oriented GNU/Linux distribution named Flop has been created. The current Flop release is 0.5, based on the 3.14 Linux kernel, glibc 2.21, and systemd 219; the compiler in use is gcc release 4.9.2. The TANGO Controls framework is also included, featuring OmniORB 4.1.6, ZeroMQ 3.2.4 and TANGO 8.1.2.

Flop Recipes

In the Yocto Project, the key mechanism that allows to define a specific platform support is referenced as recipe. Each recipe specifies the procedures to select and compile software packages. Recipes can be grouped in layers, and multiple layers can be specified to introduce dependencies. Recipes, usually written as shell scripts o Python scripts, make use of common variables that can be accessed by

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

different recipes or from configuration files. These features make possible to specify and build systems enabling only very specific functionalities of the required subsystems, whereas the standard distribution package based approach usually enables all the functionalities. Thus, useless, redundant or unwanted features can be easily excluded, improving the system reliability and, as a side effect, decreasing the footprint. As an example, the recipe written for TANGO is listed below.

```
DESCRIPTION = "TANGO is an object oriented distributed control \
  system using CORBA (synchronous and asynchronous \
  communication) and zeromq (event based communication)"
HOMEPAGE = "http://www.tango-controls.org"
LICENSE = "LGPLv3+"
LIC_FILES_CHKSUM = "file://COPYING.LESSER; \
  md5=6a6a8e020838b23406c81b19c1d46df6"

PR = "r0"

DEPENDS += "zlib bash omniorb zeromq"
RDEPENDS_${PN} += "bash"

S = "${WORKDIR}/tango-${PV}"

SRC_URI = "http://download.sourceforge.net/project/tango-cs/
  tango-${PV}c.tar.gz"

SRC_URI[md5sum] = "3dbcc2cf34f8c9395ee72f4ee5ae05dc"
SRC_URI[sha256sum] = "0149e797e5745b1dd8d5d39260889b6da31c8
  4c75c272372255ae8ca3507a116"

inherit autotools pkgconfig systemd

EXTRA_OECONF_append = " --disable-jpegmmx --disable-static \
  --disable-java --without-java --disable-dbserver \
  --disable-dbcreate --enable-stdcxx11=no"
CXXFLAGS_prepend = "-std=gnu++98 "

SYSTEMD_SERVICE_${PN} = "starter.service stopper.service"

do_configure_prepend() {
  ( cd ${S}; ${S}/bootstrap )
}

do_install_append() {
  install -d ${D}${systemd_unitdir}/system
  install -m 0644 ${WORKDIR}/starter.service \
    ${D}${systemd_unitdir}/system
  install -m 0644 ${WORKDIR}/stopper.service \
    ${D}${systemd_unitdir}/system
}
```

In addition to defining some descriptive keywords, such as `DESCRIPTION` or `HOMEPAGE`, the recipe contains the checksums of the specified licence and source files as well as the dependencies toward other tools or libraries. Extra configuration directives for the autotools and the compiler are also specified.

Everything that needs to be included in an embedded system distribution based on the Yocto Project, and therefore in Flop, have to be specified in a recipe. Accordingly, the whole system, built on the basis of the recipes, is univocally defined and characterized by the release number.

Multiple Platform Support

The first advantage of the Yocto Project approach is that Flop support includes all the embedded systems and microprocessor architectures used at Elettra: four generations of PowerPC based VME single board computers manufactured by Artesyn, formerly Emerson/Motorola, the MVME5110, MVME6100, MVME7100 and MVME2500, as well as the ARM based BeagleBone embedded system and the INTEL Bay Tray based Jetway JBC311U93 Soc. Flop provides specific platform support for mathematical coprocessors, vector accelerators, such as the Altivec engine or the SSE4, and optimized intruction sets. The above is especially true for the MVME2500 single board computer P2010 e500 v2 core QorIQ processor, which single and double precision embedded scalar floating point unit is somehow different from the classic floating point unit and uses the integer register file.

The availability of board support packages (BSP) deserves to be analyzed. Each BSP usually supports a very specific kernel release and, even within a single family of boards made by one manufacturer, often applies to a different kernel release for different board models. This makes the development and the management of kernel code, such as device drivers or kernel modules, more complicated. To solve this problem a back/forward porting of the BSP to the current release of Flop has been done, with the request to the community to include it in the mainline. Similarly, the required device drivers have been ported to the current release of Flop. As an example, some changes to the existing Tsi148 PCI-VME bridge device driver, aimed at adding a more transparent and flexible support whenever several different slave VME boards are installed, have been proposed.

The use of different platforms and different distributions sometimes leads to system software and application software fragmentation. The centralized system software management allows to keep the software aligned and up to date for all the platforms. Flop, being based on the Yocto Project, does not make use of a package based approach. On the contrary, all the required components are listed on a recipe, and the whole system compiled from source based on it. This single source approach, although sacrifices the flexibility of the package based software, guarantees the consistency of Flop with respect to all the architectures, as well as the very same patchlevel.

Development Tools

Interesting embedded boards exist which are low resources and low computing power. Moreover old boards are in use at Elettra, such as the MVME5110, which can be considered low performance with respect to modern systems. Native mode development on these platforms, meaning that the tools and the compiler run on the target system, is often quite slow and sometimes limited, or prevented, by the shortage in memory and the inappropriate storage devices. The Yocto Project allows to generate the required cross-compiler tools for the host architecture, typically fast and cheap INTEL-based workstations or servers. Ubuntu 14.04 LTS, running on a modern INTEL-based 64 bit hardware, has been selected as the operating system for the host platform for Flop. The repetitive tasks connected to software development are therefore made on powerful multiuser systems with plenty of memory and fast storage, with an

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

explicit benefit for the developer convenience and time savings.

Moreover, some special subsystems may require very specific tools that are often not included in standard GNU/Linux distributions. An example is the programmable real-time unit and industrial communication subsystem (PRU-ICSS) of the BeagleBone. This engine consists of dual 32-bit RISC cores with memory, interrupt controller and internal peripherals. In addition to the supported real-time industrial protocols, such as EtherCAT [5], PROFIBUS and PROFINET [6], EtherNet/IP [7], Ethernet Powerlink [8] and SERCOS III [9], the PRU subsystem has been exploited for some special applications at Elettra Sincrotrone Trieste. These include driving piezo-electric actuated mirrors in real-time for the Fermi seed laser trajectory feedbacks and interfacing the new power supply controllers for the Elettra storage ring [10]. Therefore, the PRU C compiler has been included in the Flop SDK, ensuring the straightforward availability as well as the versioning of the tool with Flop.

On-board Storage

Several partitions are foreseen for the on-board storage; as described hereafter, the current release of Flop requires at least four partitions: one for the boot loader, two for the system image, one for data when in standalone operation without networking.

One of the key features of Flop, as appears from the considerations in the previous sections, is the mandatory versioning which applies to the whole SDK and, of course, to the target platform support. Strict versioning means that systems with the same version must be identical. This aspect can be a limitation whenever a large number of hosts is involved and the application software, even limited to configuration files, is subject to change quite often.

To overcome this restriction the root filesystem of Flop relies on two physical partitions and on the overlay filesystem [11]. The overlayfs allows one, usually read-write, directory tree to be layered onto another read-only directory tree. All modifications go to the upper, writable layer. In Flop, the "lower" directory tree is used for the read-only system image. The "upper", used read-write, stores the changes with respect to the versioned image and is mounted over the previous one exploiting the overlayfs.

Flop also provides a simple and reliable mechanism for system update, with rollback capabilities. Two partitions are foreseen for the Flop system image, one of which is active. A major update can be performed, on a running system, writing the new Flop image to the inactive partition and reconfiguring the system to boot from that one. On next reboot the boot loader will load the updated system. It's worth noting that the specific application and configuration data, stored in the read-write partition by means of the overlayfs, are preserved because not affected by the update procedure. Should it emerge any issue with the updated system, the rollback to the previous image is straightaway and just implies the reset of the boot parameter and the system reboot. The local storage layout and the interactions with the network share, described in the next section, are shown in Fig. 1 for the network operating mode case.

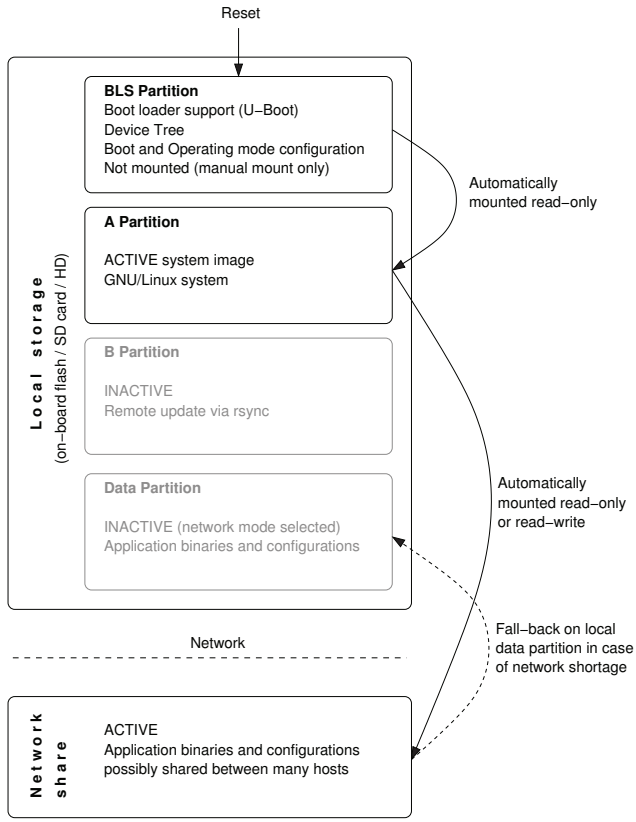


Figure 1: Local and network storage layout and interactions.

Standalone/Network Operation

Flop has been designed to support both network and network-less, or standalone, operation, where a network enabled setup is not limited to the configuration of the IP address but also includes DHCP, NFS, NIS, NTP, SSH, HTTP and FTP support. Both operating modes have to be addressed in a simple and effective way, minimizing the complexity from the user point of view. Configuration files, binary executables and, possibly, kernel modules, especially when related to specific user applications, can reside in different locations of the filesystem, depending on the selected operating mode. The solution found provides an additional partition to store all the configuration files, required by the boot loader, in order to specify and select the operating mode. If the standalone mode is selected, Flop makes use of a local partition for the data storage; if, conversely, the network mode is selected a part of the filesystem can be shared over a network. Currently, the NFS is not compatible with the overlay mechanism in use; the most suitable network protocol is, at the moment, 9P [12].

When in network mode, essential services, such as IP configutation, DNS, routing and NTP, can be configured via DHCP. In addition, an approach based on DHCP custom options is under evaluation to add the support for a number of

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

parameters, not included in the DHCP specification, that will allow a complete centralized management of the system configurations. Precision Time Protocol (PTP) support is also available and, even not included by default, can be exploited if requested.

Moreover, not excluding the possible adoption of advanced remote management systems such as Puppet or Chef, an HTTP/FTP based interface, aimed at system configuration, is under development. Whenever required, a light http server can be deployed to provide a simple web page exposing the main system configuration parameters and, possibly, any configuration for specific user applications.

CONCLUSION

The increasing number of different embedded systems in use at Elettra, as well as the need of comprehensive system software updates, lead to the need of a unified approach for embedded platform management. Based on the Yocto Project, Flop, a control system oriented GNU/Linux distribution hase been designed. Flop fulfils many of the requirements of a modern embedded operating system, provides a user friendly development environment and rationalizes the support for different platforms and architectures, simplifying the administration and ensuring the consistency.

REFERENCES

[1] D. Bulfone et al., "New front-end computers based on Linux-RTAI and PPC", ICALEPCS'03, Gyeongiu, Korea (2003).

[2] C. Scafuri, L. Pivetta, "The evolution of the Elettra control system", ICALEPCS'07, Knoxville, USA (2007).

[3] M. Lonza et al, "The control system of the FERMI@Elettra free electon laser", ICALEPCS'09, Kobe, Japan (2009).

[4] Yocto Project website: `https://www.yoctoproject.org`

[5] EtherCAT Technology Group website: `https://www.ethercat.org`

[6] PROFIBUS and PROFINET website: `http://www.profibus.com`

[7] EtherNet/IP website: `https://www.odva.org/Home/ODVATECHNOLOGIES/EtherNetIP.aspx`

[8] Ethernet Powerlink website: `http://www.ethernet-powerlink.org`

[9] SERCOS III website: `http://www.sercos.com/technology/sercos3.htm`

[10] S. Cleva, L. Pivetta, P. Sigalotti, "BeagleBone for embedded control system applications", ICALEPCS'13, San Francisco, USA (2013).

[11] Linux kernel overlay filesystem: `https://www.kernel.org/doc/Documentation/filesystems/overlayfs.txt`

[12] 9P website: `http://9p.cat-v.org`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PHYSICS APPLICATION INFRASTRUCTURE DESIGN FOR FRIB DRIVER LINAC*

G. Shen#, E. Berryman, Z. He, M. Ikegami, D. Liu, D. Maxwell, V. Vuppala,
FRIB, Michigan State University, East Lansing, 48864 USA

Abstract

FRIB (Facility for Rare Isotope Beams) is a new heavy ion accelerator facility to provide intense beams of rare isotopes, and currently under active construction at MSU (Michigan State University). Its driver linac accelerates all stable ions up to uranium, and targets to provide a CW beam with the energy of 200MeV/u and the beam power of 400 kW. A new software infrastructure for high level control and physics application is under development to support the beam commissioning and operation, which is a 3-tier structure consisting of upper level, middle layer, and low level respectively. The detailed design and its current status are described in this paper.

INTRODUCTION

FRIB [1] is a new project under cooperative agreement between US Department of Energy and MSU. It is under construction on the campus of MSU and will be a new national user facility for nuclear science. Its driver accelerator is designed to accelerate all stable ions to energies > 200 MeV/u with beam pow er on the target up to 400 kW as shown in Table 1 [2].

Table 1: FRIB Driver Accelerator Main Parameters

Parameter	Value	Unit
Primary beam ion species	H to ^{238}U	
Beam kinetic energy on target	> 200	MeV/u
Maximum beam power on target	400	kW
Beam duty factor	100	%
Beam current on target (^{238}U)	0.7	emA
Beam spot size on target (90%)	1	mm
Driver linac beam-path length	517	m
Average uncontrolled beam loss	< 1	W/m

The layout of FRIB is as shown in Fig. 1, which consists of two ECR (Electron Cyclotron Resonance) ion sources, a low energy beam transport with a pre-buncher and a chopper, a RFQ (Radio Frequency Quadrupole) linac, LS1 (Linac segment 1) to accelerate the beam up to > 15 MeV/u, LS2 and LS2 to accelerates the beam > 200 MeV/u, two folding segments to confine the footprint and facilitate beam collimation, and a beam delivery system to transport to the target. The beam is stripped to higher charge states after LS1 section.

*Work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661
#shen@frib.msu.edu

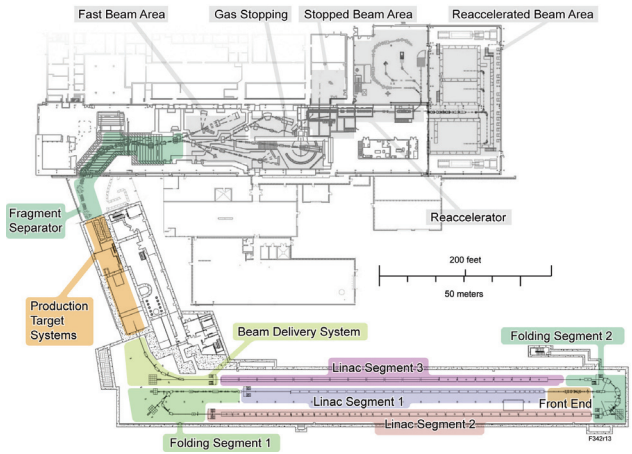


Figure 1: Layout of the FRIB driver accelerator, target and fragment separator as shown in colored area.

ARCHITECTURE

A distributed high level controls environment has been adopted for its beam commissioning and operation as shown in Fig. 2. It is a 3-tier structure consisting of upper level, middle layer, and low level respectively.

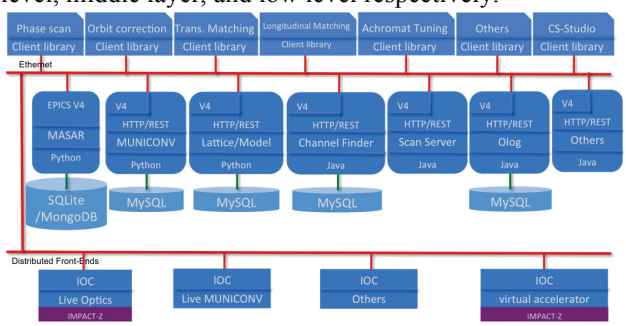


Figure 2: FRIB high level controls system architecture.

Since EPICS (Experimental Physics and Industry Control System) has been adopted as FRIB hardware control framework, the low level is represented as an EPICS IOC (Input Output Controller) for integration purpose. Some services are implemented in the low level layer, the middle layer consists some general services, and high level is mainly for OPI (Operator Interface).

LOW LEVEL SERVICE

At FRIB, the low level service is implemented as an EPICS IOC. One typical service is a virtual accelerator, which provides emulated accelerator behaviour such as setting and reading back a power supply setting, and reading beam orbit at each BPM position.

The architecture of FRIB virtual accelerator is as shown in Fig. 3. It uses IMPACT-Z [3] as underneath simulation engine. During first initialization, it reads

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

accelerator from accelerator layout file and accelerator setting file, generates an IMPACT-Z input file, and initialize all hardware settings. Once the input file is ready, it executes the IMPACT-Z engine, reads simulation results, and updates the result. Every time when there is any setting change, it updates the lattice file, runs IMPACT-Z and updates the results.

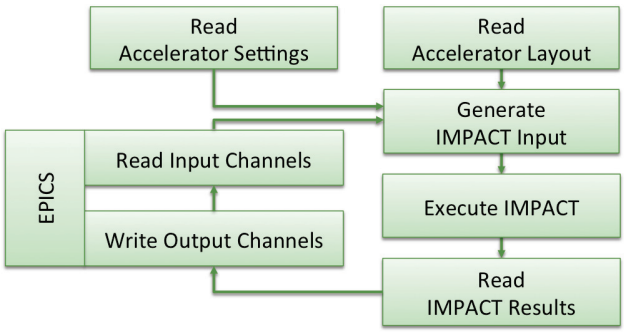


Figure 3: FRIB virtual accelerator.

MIDDLE LEVEL SERVICE

Currently, to support FRIB construction, beam commissioning and operation, a minimum set of middle layer services has been identified as below:

- eTraveler;
- Cable database;
- Configuration database (CDB);
- RBAC;
- Archiver Appliance;
- Channel Finder;
- Olog;
- Scan Server;
- MASAR;
- BEAST;
- META data store;
- Lattice/Model Web Server;
- Online model;
- Physics data server.

There are 2 different technologies adopted at FRIB to implement the middle layer services, which are EPICS V4 and web service respectively.

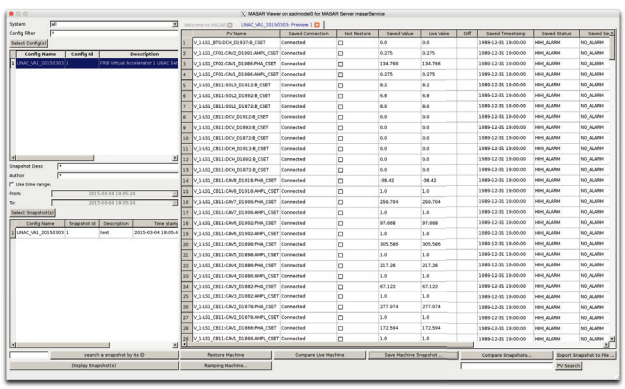


Figure 4: FRIB MASAR.

EPICS V4

The EPICS V4 has been determined as FRIB official framework for the middle layer service. The first server adopted was MASAR [4], which is a save and restore tool originally developed at NSLS II project, as shown in Fig. 4, which is one user client developed in PyQt. The development to integrate MASAR into CS-Studio is on going, and will be available shortly.

Web Service

Together with EPICS V4, web service is used to complement their weaknesses. Most databases related middle layer services listed above are developed as a RESTful web service except MASAR. One typical example is CDB as shown in Fig. 5. The CDB development is a collaboration effort with ESS (European Spallation Source) to manage machine configuration and installation information.

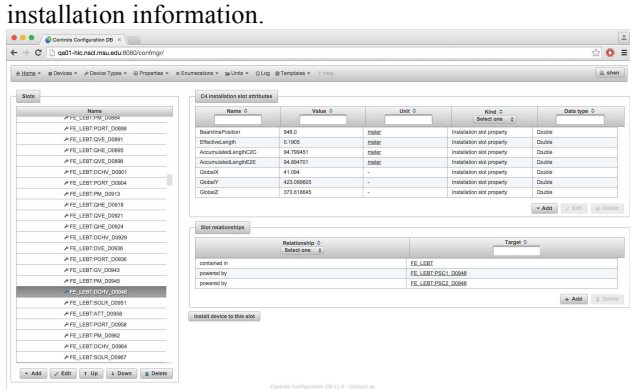


Figure 5: Configuration database.

Beyond above information, CDB also stores parameters for FRIB MPS (Machine Protection System). A software-based RPS (Run Permit System) is under design to provide several key functions for safe operation of the machine. It monitors relevant inputs from hardware subsystems, which includes but not limited to cryomodule, vacuum, power supply/magnet, diagnostics, and target equipment to verify whether they are all in the desired state for the selected machine mode, beam mode, ion species with the charges, beam energy, and the beam destination, and establishes the Beam Permit signal. All parameters for RPS are saved in CDB such as allowed setting range for a device.

HIGH LEVEL CONTROL AND CONTINUOUS INTEGRATION

The OPIs for FRIB high-level controls will be developed using CS-Studio framework. Its HMI (Human Machine Interface) has been well defined and documented, and the validation will be done with the help of CS-Studio OPI validation tool.

At FRIB, an automatic software deployment procedure is adopted as shown in Fig. 6. A developer checks out source code from FRIB git [5] repository, commits and checks in any change for the review, and gets the change merged after it passes the review. Once the merge finished, a Continuous Integration process is triggered to

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

compile and test all changes, and a package could be created after passing all tests. Package manager could manually or automatically upload the new package, and deploy it.

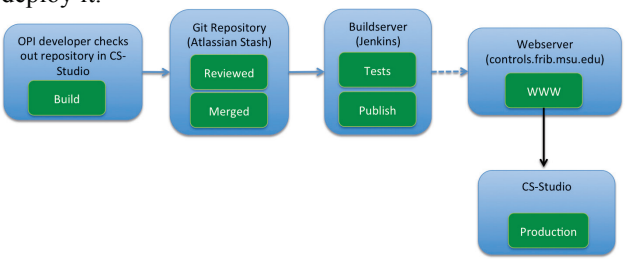


Figure 6: FRIB software development workflow.

PHYSICS APPLICATIONS

The development plan for FRIB physics application was described in [6].

Online Model

As mentioned in [6], an online model is one of the most critical components for our beam commissioning since FRIB has been designed with some specific features as below:

- Solenoid focusing lattice;
- Non-axisymmetric field components at QWRs (Quarter Wave Resonators). The non-axisymmetric nature of QWR induces dipole and quadrupole components, which has significant contribution to FRIB beam dynamics;
- Accelerate multi-charge-state beams simultaneously. FRIB is designed to accelerate up to five charge states to achieve high beam intensity. This gives a stringent beam-on-target requirements, which should be met especially for multi-charge-state beams;
- Achromat arc sections between linac segments;
- Second order achromat with sextupole magnets.

An online model named TLM (Thin Lens Model) is under active development at FRIB to meet its needs. Some details and latest status of TLM could be found in [7].

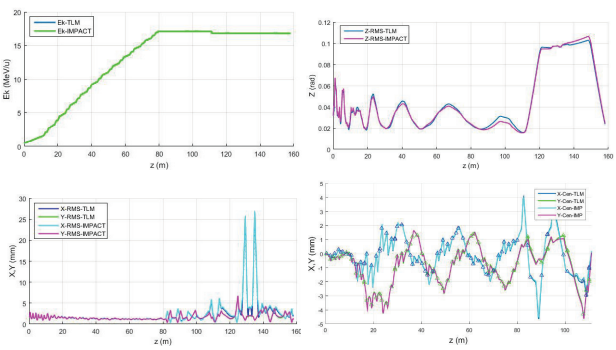


Figure 7: Benchmark of TLM against IMPACT for LS1 and FS1: (top left) Kinetic energy, (top right) RMS phase, (bottom left) transverse RMS beam size, and (bottom right) transverse beam orbit.

Fig. 7 shows a benchmark against IMPACT-Z, which compares results from the TLM and from the IMPACT-Z using the same lattice. The results show that with the lattice of FRIB LS1 and FS1 section, kinetic energy mean error is within 5e-5, longitudinal rms phase mean error is within 5%, transverse rms size mean error is within 2%, and transverse beam orbit mean error achieved within 2%.

As described above, the TLM now achieved an acceptable performance comparing with multiple particles tracking code. System integration is planned to integrate the TLM into our Python scripting environment.

Online model has been planned as a middle layer services, which is under active development. To enable physics application development from its early stage, an interface library has been developed to call it directly from physics application. Its architecture is as shown in Fig 8.

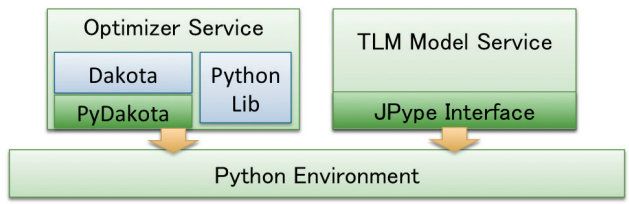


Figure 8: Architecture of TLM Model interface with Python Environment.

A prototype result for orbit correction is as shown in Fig. 9 against FRIB virtual accelerator.

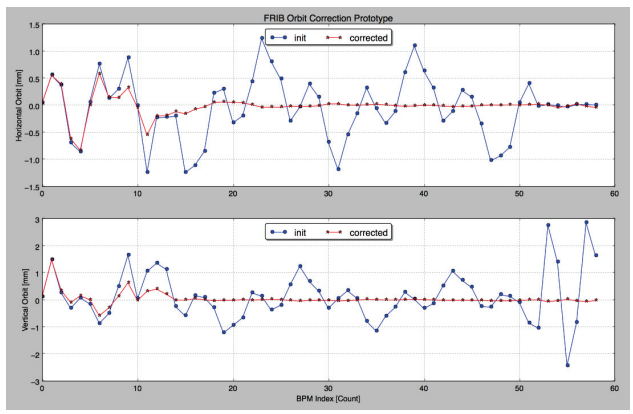


Figure 9: FRIB orbit correction prototype. The upper plot is for X direction, and lower part is for Y direction. The blue line is its original orbit, and red line is after correction.

The orbit distortion is given by artificially exciting the first steering magnet, and orbit correction is performed by using downstream correctors to flatten the downstream orbit. The results is with 5 iterations, and each iteration costs about 16 seconds with a few hundreds model calling.

System Integration

With the middle layer service, physics application development becomes much easier than ever before.

With Channel Finder service [8], there is no need to hardcode PV name any more, and the code is much clean and simple as shown in Fig. 10.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

```
In [4]: # get the cavity to scan
cavities = ap.getElements("CAV")
# get pv name of cavity phase settings
device = str(cavities[0].pv(field="PHA", handle="setpoint")[0])
readback = str(cavities[0].pv(field="PHA", handle="readback")[0])
# get 2 downstream BPMs near that cavity
bpmdowns = ap.getNeighbors(cavities[0], "BPM", n=2, elemself=False)[2:]
# get specific BPMs
bpms = ap.getElements(['LS1_WA01:BPM_D1155', 'LS1_WA02:BPM_D1189'])
# get all cavities in between to be disabled
cav2disable = ap.getElements('LS1_CA*:CAV*', start=cavities[0].se, end=bpms[1].sb)
# get pv information for amplitude and phase
cav_pv = [str(cav.pv(field="AMP", handle='setpoint')[0]) for cav in cav2disable]
cav_ph = [str(cav.pv(field="PHA", handle='setpoint')[0]) for cav in cav2disable]
tolerance = 1.0
start = 0
stop = 360
step = 2
meas_dev = [readback,
            str(bpms[0].pv(field="PHA")[0]), str(bpms[1].pv(field="PHA")[0]),
            status_pv,
            str(bpms[0].pv(field="ENG")[0]), str(bpms[1].pv(field="ENG")[0]),
            str(bpmdowns[0].pv(field="PHA")[0]), str(bpmdowns[1].pv(field="PHA")[0])]
samples = 1
timeout = 5.0
delay = 1.5
samples=1

In [5]: orig = ca.caget(device)

        #original setting for downstream cavities
        orig_phase_amp = ca.caget(cav_pv+cav_ph)

        # turn off down stream cavities
        res = ca.caput(cav_pv+cav_ph, [0.0]*len(cav2disable)*2, wait=True)
```

Figure 10: Using Channel Finder service to search correct channel name.

Once get all needed channel names, user could ask other service to perform a desired work, for example a scan for RF cavity scan. Fig. 11 shows an example to use Scan Server to perform a 2-π scan.

```
In [6]: # perform a 1-D phase scan
scanId_id = scanclient.scanId(device, start, stop, step, meas_dev,
                              title="1D scan example",orig = orig,
                              readback=readback, tolerance=tolerance, wait=False,
                              timeout=timeout, delay=delay, samples=samples)

In [7]: # check scan status
scanstatus = scanclient.getscanstatus(scanId_id)
while not scanstatus.isDone():
    print '.', scanstatus.percentage(), "%",
    time.sleep(2.0)
    scanstatus = scanclient.getscanstatus(scanId_id)

. 1 % . 2 % . 5 % . 7 % . 10 % . 12 % . 15 % . 18 % . 20 % . 23 % . 25 % . 28 % . 31 % . 33 % . 36 % . 38 % . 41 % . 44 % . 46
% . 48 % . 49 % . 51 % . 54 % . 57 % . 59 % . 62 % . 64 % . 67 % . 70 % . 72 % . 75 % . 77 % . 80 % . 83 % . 85 % . 88 % . 90 %
. 93 % . 96 % . 98 %

In [8]: # restore device
res = ca.caput(cav_pv+cav_ph, orig_phase_amp, wait=True)
# get scan data
results_ID = scanclient.getscandata(scanId_id, [device] + meas_dev)
print results_ID[4]

['OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK',
'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'OK']
```

Figure 11: Example of 2-π scan using Scan server.

As shown in Fig. 12, the phase scan becomes one code, and the rest lines are for checking the scan progress, restoring setting to that before scanning, and manipulating the data like plotting. The scan result is as shown in Fig. 9.

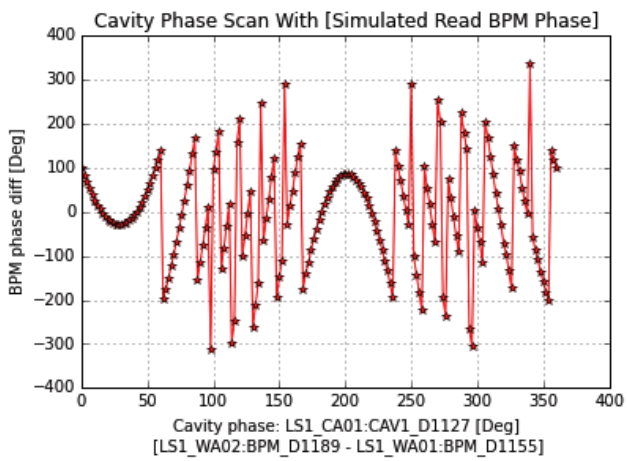


Figure 12: Phase scan result, which is for the first cavity of FRIB LS1 section. The names of related PVs are from Channel Finder Service, and the scan is done with Scan server.

CONCLUSION

Distributed service oriented architecture has been adopted for FRIB beam commissioning and operation. The infrastructure to support physics application development is under active development, and minimum set of service is ready to enable the development at its early stage. Two (2) technologies are used for our middle layer service development, which are EPICS V4 and web service respectively. We have started to develop physics applications under Python environment, and within an integrated environment.

ACKNOWLEDGMENT

The authors would thank our FRIB colleagues, especially S. Lund, E. Pozdeyev, J. Wei, Y. Yamazaki, Q. Zhao, Y. Zhang, for their helpful discussions and suggestions.

REFERENCES

[1] http://www.frib.msu.edu
[2] J. Wei et al, “FRIB Accelerator: Design and Construction Status”, MOM1I02, proceedings of HIAT15, Yokohama, Japan (2015).
[3] J. Qiang et al. Impact. In Proceedings of the 1999 ACM/IEEE conference on Supercomputing, page 55. ACM, 1999.
[4] G. Shen et al., “NSLS II Middlelayer Services”, MOPPC155, proceedings of ICALEPCS2013, San Francisco, US (2013).
[5] https://git-scm.com
[6] M. Ikegami, G. Shen, “Development Plan for Physics Application Software for FRIB Driver Linac”, MOPWI023, proceedings of IPAC15, VA, US (2015).
[7] G. Shen et al., “Development Status of a Thin Lens Model for FRIB Online Model Service”, TUDBC2, proceedings of ICAP2015, Shanghai, China (2015).
[8] https://github.com/ChannelFinder/ChannelFinder Service

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

LabVIEW EPICS PROGRAM FOR MEASURING BINP HLS OF PAL-XFEL*

Hyojin Choi†, Kwang Won Seo, Youngjin Suh, Kyehwan Gil, Seung Hwan Kim, Heung-Sik Kang,
Department of Accelerator, PAL-XFEL, Pohang, Korea

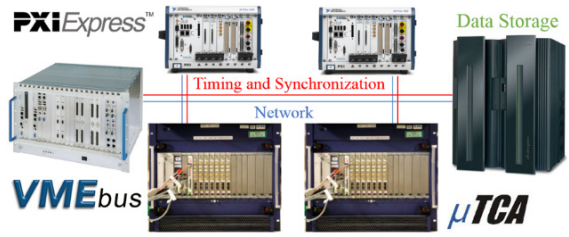
Abstract

In PAL-XFEL, a 4th generation light source, the HLS (Ultrasonic-type Hydrostatic Levelling System) developed at BINP (Budker Institute of Nuclear Physics) in Russia was installed and operated in all parts of PAL-XFEL in order to maintain observations of the vertical change building floor by the ground sinking and uplifting. For this, a HLS measuring program was written using NI LabVIEW and an EPICS IOC Server was built using the CA Lab which has been developed at BESSY (Berlin Electron Storage Ring Society for Synchrotron Radiation) in Germany. The CA Lab was improved and verified in order to confirm that it could support EPICS BASE libraries V3.14.12, and EPICS CA Client and that the EPICS IOC Server could be easily constructed by CA Lab in a 64-bit LabVIEW. This made Multi-core CPU (Multi-core Processor / Multi-thread Program) resource of 64bit Computer System (64bit Hardware PC, 64bit Windows OS, 64bit LabVIEW Multi-thread Programming) to be 100 percent utilized. This study proposes a configuration process for the HLS measuring program algorithm and a building process for the EPICS IOC Server by using CA Lab.

INTRODUCTION

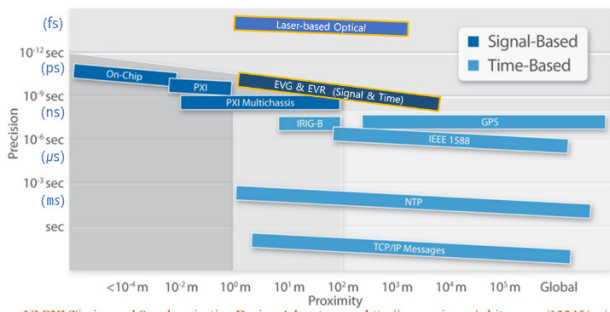
As devices created due to big science and big technology have been placed far away from control systems in terms of space and geography, advanced technology, including that for a distributed control system has become necessary (7.7 Control and Timing [1]). Various dispersed control platforms have been installed for PAL-XFEL like Figure 1. Owing to recent rapid developments in information technology (IT), such as advanced robot technology, Internet of things (IoT), big data and artificial intelligence, it is easy to establish such a new system through buying a distributed control system device developed by diverse companies. Engineers in many accelerator research institutes around the world have been engaged in efforts to advance effective technological developments by sharing technology through the open platforms of control systems and combining their experience and knowledge [2].

As shown in Figure 2, in order to synchronize an ordinary distributed control system, the time-based (time-stamp) synchronization of the entire big scientific device should be effected, first of all. After this, the signal-based synchronization of a module inside each device or the device should be achieved. Regarding RF reference of FEL and pump-probe experimental devices of beamline, a jitter and drift need a synchronization system of the sub-10 rms femtosecond level [3], and a laser-based optical synchronization system for the highest precision of synchronization has been developed to be applied to them (7.8 Femtosecond precision optical synchronization [1])[4].



① Establish a distributed control system.
② Design a high-performance and high-bandwidth network system suitable for a distributed control system and establish it.
③ Install a storage and backup system that can perform high-speed processing of large data in order to store large measurement control data without data loss.
④ Install a timing and synchronization system to get synchronized measurement data.

Figure 1: Construction of a distributed control system.



NI PXI Timing and Synchronization Design Advantages http://www.ni.com/white-paper/13345/en/

Figure 2: Signal-based and time-based synchronization architecture performance.

Timing and synchronization technologies correlate events in time, which is necessary to perform coordinated activities. For software to orchestrate these coordinated activities, the program needs to be synchronized and include a concept of time. As shown in Figure 3, hardware and application programs can be accurately synchronized only when the timing structure is within all systems.

In order to operate big scientific devices stably for a long time, measurement control devices that operate individually also should be highly reliable and available to be operated for a long time without problems occurring. For this purpose, stable and reliable hardware platforms and operating systems (OS) should be selected.

*Work supported by Ministry of the Science, ICT and Future Planning
†choihyo@postech.ac.kr

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

With regard to the persons taking charge of these devices, it is important for them to perform the application programming in an effective and rational manner. Only when they understand the principles and concepts of the physical, chemical, instrumental and electric movements of controlled systems, can they think of the best algorithms and write effective (optimum) application programs.

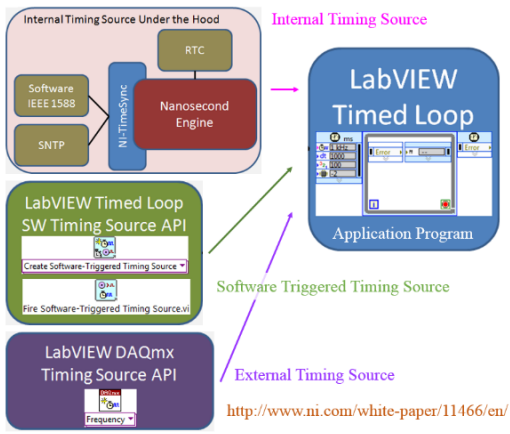


Figure 3: Timing and synchronization in NI LabVIEW.

NI PXI PLATFORM

PCI eXtensions for Instrumentation (PXI) is based on industry-standard computer buses and permits flexibility in building equipment. Since it was developed in 1997, PXI has been managed by PXI Systems Alliance (PXISA). Regarding PXI Architecture, refer to the PXI Hardware Specification file issued by PXISA [5]. With regard to PXI, various kinds of high-performance and input-output modules have been produced, so people can choose the right modules for their purposes. If fast feedback control is necessary, a real-time system should be established with NI FlexRIO custom instruments using FPGA. The real-time system using FPGA should be remotely connected to a network and replaced with an improved algorithm. Figure 4 show the NI PXI platform.

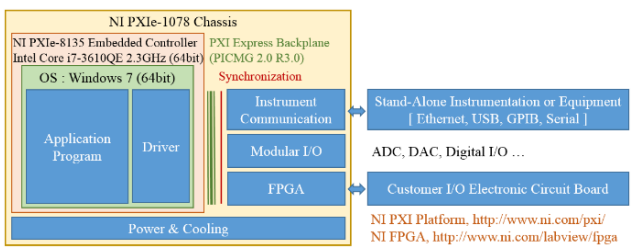


Figure 4: Construction of a PXI platform.

NI PXI Multi-core Controller

Figure 5 shows the NI PXIe-8135 block diagram and architecture of Intel Core i7. A programmer should review the specifications of hardware programs and controllers in use and then write the best and most effective program algorithm for the purposes at hand. The Intel Core i7-3610QE processor, a multi-core processor, applied to the PXIe-8135 controller has four cores and each core can make two threads work [6]. As shown in Figure 6, operational states of a multi-core processor can be checked through the Windows task manager.

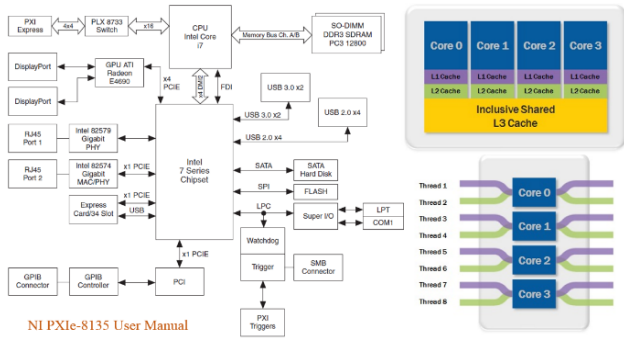


Figure 5: NI PXIe-8135 block diagram and Intel Core i7 architecture.

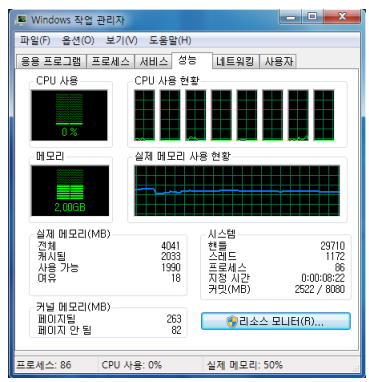


Figure 6: Windows task manager.

Multi-core Programming with LabVIEW

Many programmers are enthusiastic about the performance of multi-core but they sometimes can't utilize it properly. As shown in Figure 7, applications written with LabVIEW can choose a processor of multi-core in a simple way. There are various program languages that we can choose and each program language has its strong points and weak points as well as distinguishing features. The advantages of the LabVIEW program are that it is convenient and immediate. Anyone can learn about it quickly and easily control various kinds of diagnostic control hardware systems [7].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

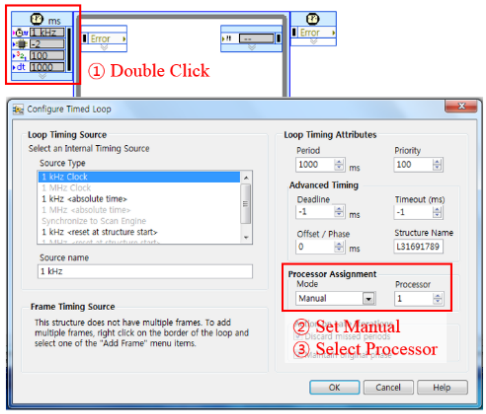


Figure 7: How to choose multi-core processors.

BESSY CA LAB

There are diverse ways to materialize EPICS communication in LabVIEW, as shown in Figure 8. CA Lab that satisfies the conditions of the EPICS BASE library (V3.14.12) and can be used in LabVIEW was developed and distributed by BEESY in Germany. Programmers can download suitable Ca Lab for their LabVIEW version (32bit or 64bit) and OS type (Windows or Linux) and use it. Ca Lab is a 64bit EPICS library that can use functions of 64bit CPU, 64bit OS and 64bit LabVIEW perfectly and BESSY Ca Lab provides ways to use it and a file of examples. Figure 9 shows the concept of movements of Ca Lab [8].

- LabVIEW® DIM Interface ⇔ EPICS - DIM Interface
- LabVIEW® Shared Memory Interface to EPICS IOC by SNS
- LabVIEW® ActiveX CA by Kay Uwe Kasimir, ORNL
- LabVIEW® Data Logging and Supervisory Control Module, NI
- LabVIEW® native EPICS implementation by SNS
- LabVIEW® CA Lab Channel Access implementation by BESSY

https://wiki.gsi.de/Epics/ConnectingLabVIEWandEPICS

Figure 8: Connecting LabVIEW and EPICS.

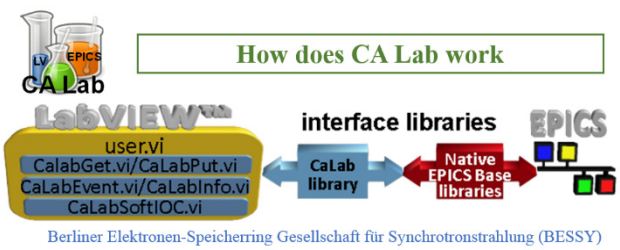


Figure 9: Concept of Ca Lab movements.

HLS IOC SERVER

Figure 10 shows the constitution of IOC server programs using PXI. A program bringing measurement data of HLS connected to a private network periodically reads the measurement of HLS with a time interval. Measurements of HLS are recorded in the shared memory which sends a wakeup signal to the analysis & display program after reading all HLS measurements. The analysis & display program that receives the wakeup signal then analyzes HLS measurements recorded in the shared memory, and displays the analysis result to the monitor and enters a wait condition, a sleep mode. When the Ca Lab IOC server program connected to the IOC network gets a request from data storage to store HLS data, the program sends HLS measurements data in the shared memory to data storage.

As shown in Figure 4, the NI PXIe-8135 controller has two Ethernet ports. IOC EPICS communication was connected to Port 2 (Intel82574) exclusively for servers and private communication for BINP HLS measurement was connected to Port 1 (Intel82579).

BINP HLS TEST ON PAL-XFEL

Figure 11 shows the method of using BINP HLS and the result of testing ULSE 2 sets which was borrowed from BINP to learn about the operation [9].

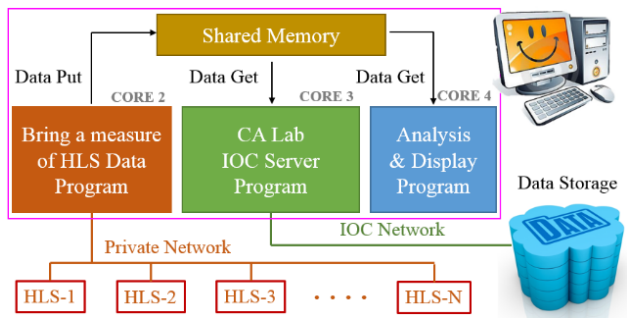


Figure 10: Multi-processor program.

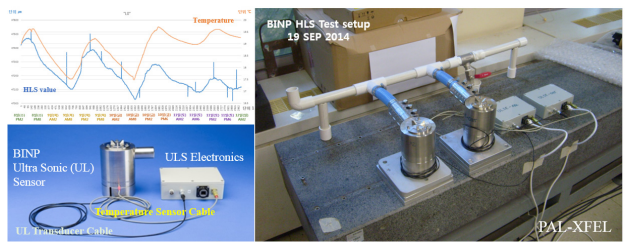


Figure 11: BINP HLS test on PAL-XFEL.

REFERENCES

[1] A.W. Chao, et al., "Handbook of Accelerator Physics and Engineering", 2nd Edition, Wold Scientific Pub Co Inc.

[2] D. Gurd, et al., "Accelerator Control and Global Networks – State of the Art", proc. of LINAC2004, FR201 (2004).

[3] S. Schulz, et al., "Femtosecond all-optical synchronization of an X-ray free-electron laser", proc. of Nature Communications (2015).

[4] S. Hunziker, et al., "Reference Distribution and Synchronization System for SwissFEL: Concept and First Results", proc. of IBIC2014, MOCZB2 (2014).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

[5] PXI System Alliance (PXISA), http://www.pxisa.org/
[6] Top Eight Features of the Intel Core i7 Processors for Test, Measurement, and Control, http://www.ni.com/white-paper/11266/en/
[7] Advantage of Using LabVIEW in Academic Research, http://www,ni.com/white-paper/8534/en/
[8] BESSY CA Lab, http://www-csr.bessy.de/control/SoftDist/CA_Lab/
[9] H. Choi, et al., "The BINP HLS to Measure Vertical Changes on PAL-XFEL Buildings and Ground", proc. of FEL2015, MOP047 (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PvaPy: PYTHON API FOR EPICS PV ACCESS*

S. Veseli, Argonne National Laboratory, Argonne, IL 60439, USA

Abstract

As the number of sites deploying and adopting EPICS Version 4 grows, so does the need to support PV Access from multiple languages. Especially important are the widely used scripting languages that tend to reduce both software development time and the learning curve for new users. In this paper we describe PvaPy, a Python API for the EPICS PV Access protocol and its accompanying structured data API. Rather than implementing the protocol itself in Python, PvaPy wraps the existing EPICS Version 4 C++ libraries using the Boost.Python framework. This approach allows us to benefit from the existing code base and functionality, and to significantly reduce the Python API development effort. PvaPy objects are based on Python dictionaries and provide users with the ability to access even the most complex of PV Data structures in a relatively straightforward way. Its interfaces are easy to use, and include support for advanced EPICS Version 4 features such as implementation of client and server Remote Procedure Calls (RPC).

INTRODUCTION

EPICS Version 4 (EPICS4) [1] extends Version 3 [2] with features like support for complex data structures and service oriented architecture, optimized data transfers, as well as support for high level data and image processing. It also comes with a comprehensive set of C++ and Java APIs. However, what has been missing until recently is support for scripting languages. PvaPy aims to fill that gap by providing a Python API for the EPICS PV Access (PVA) protocol.

Rather than providing a direct Python implementation of the PVA protocol, PvaPy wraps EPICS4 C++ code using the Boost.Python [3] framework, a C++ library that enables seamless interoperability between C++ and Python. The main advantage of this approach is that it allows us to build on the existing EPICS4 code base and functionality, which significantly reduces PvaPy development effort.

BUILD PROCESS

Prerequisites for building PvaPy from sources [4] include the following:

- EPICS Base (v3.14.12.x, or v3.15.x) [5]
- EPICS4 C++ release (v4.4.0 or v4.5.0) [6]
- Python development header files/libraries (v2.6.x or v2.7.x) [7]
- Boost (v1.41.0 or later); installation must include the Boost.Python library [3]
- Standard set of GNU development tools (gcc, make, autoconf, etc.) [8]
- Sphinx (Python Documentation Generator) [9]; this is an optional package, generating documentation at build time is not essential.

Except for EPICS Base and the EPICS4 C++ release, all software dependencies listed above are typically included in most Linux operating system (OS) distributions. PvaPy has not been built or tested on Microsoft Windows.

PvaPy utilizes the standard EPICS build infrastructure [10]. However, unlike most EPICS modules, it also offers the possibility of configuring the software build automatically, using the GNU Autoconf [11] and a set of M4 [12] macros. Automated configuration determines compiler flags appropriate for the given operating system, and for the specific versions of Boost and Python that are installed on the build machine. Configuration scripts also determine the PvaPy API version that is suitable for the particular version of EPICS4 release, as well as prepare user environment setup scripts. User setup scripts modify PYTHONPATH environment variable so that PvaPy's "pvaccess" module can be imported within Python scripts or for interactive usage.

SOFTWARE FEATURES

PvaPy provides C++ code which calls the EPICS4 C++ libraries and defines a set of high-level classes for data objects, exceptions, and client/server interfaces. Those classes and their interfaces are exposed to users as the Python "pvaccess" module using the Boost.Python framework. PvaPy also defines a number of low-level utility and helper classes that are either required by EPICS4 APIs, or handle things like conversion between various Python and EPICS4 data structures. Note that the new high-level PVA Client C++ module [13] (available as part of the EPICS4 v4.5.0 release) greatly simplifies EPICS4 client interfaces and significantly reduces the number of internal classes that are implemented in PvaPy.

PvaPy Objects

EPICS4 C++ data types and modelling APIs are part of the PVData C++ package [14]. In PvaPy, the base class for all PV data types is *PvObject*, which represents a generic PV Structure. PvObject is initialized with a Python dictionary of PV introspection data, a set of key/value pairs describing the underlying PV structure in terms of field names and their types. The dictionary key is a string (the PV field name), and the value can be one of:

- PVTYPE: a scalar type, any of BOOLEAN, BYTE, UBYTE, SHORT, USHORT, INT, UINT, LONG, ULONG, FLOAT, DOUBLE, or STRING
- [PVTYPE]: a single element list, representing a scalar array

* Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under contract DE-AC02-06CH11357.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- {key:value,…}: a dictionary, representing a structure
- [{key:value,…}]: a single element list containing a dictionary, representing a structure array
- (): an empty tuple, representing variant union
- [()]: a single element list containing an empty tuple, representing variant union array
- ({key:value,…},): a single element tuple holding a dictionary, representing a restricted union
- [({key:value,…},)]: a single element list containing a single element tuple of a dictionary, representing a restricted union array

In this way, we can easily describe even the most complex PV structures using standard Python data types and structures like dictionaries, lists and tuples. For example, a PV structure containing a structure array and a restricted union would be initialized as in Figure 1 below:

```
pv = PvObject({
  'sArray':[{'i':INT,'d':DOUBLE}],
  'u':({'f':FLOAT,'s':STRING},)
})
```

Figure 1: Initializing PvObject consisting of a structure array and a restricted union.

Field values for PvObject instances can be set using a dictionary keyed on the field names:

```
pv.set({
  'sArray':[
    {'i':1,'d':1.1},
    {'i':2,'d':2.2}
  ]
})
```

Figure 2: Setting PvObject's value via a Python dictionary.

The corresponding "get()" method returns a Python dictionary of all the PvObject's field values. Another way of manipulating and accessing a PvObject's fields is to use setters and getters that correspond to different field types. For example, setting a structure array can be done through the "setStructureArray()" method:

```
pv.setStructureArray(
  'sArray',
  [
    {'i':1,'d':1.1},
    {'i':2,'d':2.2}
  ]
)
```

Figure 3: Setting a specified structure array field.

Even though the PvObject class can represent any PV Data structure, PvaPy also comes with a number of specialized classes that wrap some of the standard PV Data types. Those include classes for various scalar types (*PvByte*, *PvInt*, etc.) and scalar arrays (*PvScalarArray*), unions (*PvUnion*), timestamps (*PvTimeStamp*), and alarms (*PvAlarm*). Note that wrapper classes for the EPICS4 Normative Types (NT) [15] have not yet been fully implemented.

Channel Class

PvaPy's *Channel* class provides the Python interface for communicating with PV Access channels, as well as for their monitoring. It is worth noting that this class also supports Channel Access (the EPICS Version 3 protocol) as well as PV Access. As of the EPICS4 release v4.5.0, the Channel class implementation is based on the PVA Client C++ package [13].

Users have the ability to retrieve and set process variable values through a Channel's "get()" and "put()" methods. The "get()" method returns a PvObject representing the current value for the given process variable. The "put()" method accepts either a PvObject or a standard Python data type as input for setting the process variable. For example, when "doubleArray" is the name of a PV channel for a structure containing an array of doubles, the following Python statements will initialize the Channel object and set its PV value:

```
c = Channel('doubleArray')
c.put([1.0,2.0,3.0])
```

Figure 4: Initializing the "doubleArray" Channel object and setting its PV value via a Python list.

The Channel class' monitoring functionality allows users to subscribe to PV value changes and process them with a Python function that takes a PvObject as an argument and has no return value. The code in Figure 5 monitors the above "doubleArray" channel, and prints the sum of the array's values after every change:

```
def sum(pv):
  s = 0
  for d in pv.get()['value']:
            s += d
  print s
c.subscribe('sum',sum)
c.startMonitor()
```

Figure 5: Monitoring PV channels.

Note that one can subscribe to PV value changes with an arbitrary number of monitor processing functions.

RPC Server and Client

The *RpcServer* class is used for hosting one or more PVA Remote Procedure Call (RPC) services. Users define an RPC processing function (which may be a Python class member), and register it with an RpcServer instance. The RPC processing function takes a client request PvObject as input, and returns a PvObject containing the processed result. Figure 6 below illustrates code for defining and registering a simple RPC service that returns sum of two numbers provided in an RPC request:

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

```
def sum(pvRequest):
  a = pvRequest.getInt('a')
  b = pvRequest.getInt('b')
  return PvInt(a+b)
srv = RpcServer()
srv.registerService('sum',sum)
srv.listen()
```

Figure 6: A simple RPC service returning the sum of two numbers from the client's request.

A single RpcServer class instance can host multiple RPC services, each accessible on their own PVA channel whose name is given in the "registerService()" call. The RpcServer can be started in its own thread by invoking the "startListener()" method instead of the blocking "listen()" function call shown above. This is typically used for multi-threaded programs, or for testing and debugging in Python's interactive mode.

RpcClient is a client class for PVA RPC services. Users initialize an RpcClient object giving the service's channel name, prepare a PV request object, and then invoke the service as in the following example:

```
c = RpcClient('sum')
request = PvObject({'a':INT,'b':INT})
request.set({'a':1,'b':2})
sum = c.invoke(request)
```

Figure 7: An RPC client for the "sum" service.

The result returned by the above call will be a PvObject containing the sum of the two numbers in the request.

Exceptions

PvaPy's "pvaccess" module exposes a number of exception classes that may be raised by the API under different error conditions. Examples of these are *FieldNotFound*, *InvalidDataType*, *InvalidRequest*, etc. Note that all PvaPy's exceptions derive from the base PvaException class, and that the exception hierarchy is preserved from C++ to Python using custom exception translator code and Boost.Python's translator registration mechanism (see [3] for examples).

Documentation

All exposed PvaPy classes and methods have been documented in the code, relying on Boost.Python's support for user-defined docstrings [3]. API reference documentation can be generated from the docstrings in various formats at build time, using the Sphinx documentation generator. Alternatively, users can access the official documentation [4] generated by the EPICS4 automated builds [16].

FUTURE PLANS

The most recent PvaPy version is bundled with EPICS4 release v4.5.0 [6]. Although it is fairly functional, there are quite a few desired features, development process improvements, and performance enhancements planned for the future:

- Implementation of wrapper classes for all Normative Types [15]; the current software only supports a few NT wrapper types
- Full support for PVA channels; at the moment operations like "putGet()" and "getPut()" are not supported
- Support for Python 3; at the moment PvaPy only supports recent Python 2 versions (2.6.x or 2.7.x)
- Support for NumPy arrays [17]
- Channel monitor performance and usability enhancements; at this time, processing monitor data on multiple CPU cores requires a significant amount of user-written code
- Test framework integration and test suite development; at the moment all testing is done manually
- PVA Server implementation

Note that the above list is not exhaustive and only includes some of the most important planned features.

CONCLUSION

PvaPy is the EPICS4 Python API for PV Access. It relies on the underlying EPICS4 C++ libraries and Boost.Python framework for interfacing Python and C++.

In addition to providing Python tools for EPICS4 application developers, one of PvaPy's goals is to help promote EPICS4 usage by making it more accessible to new users. As the examples presented in this paper illustrate, PvaPy interfaces have been designed with the end user in mind: to be as simple, flexible and intuitive as possible, while still retaining all capabilities and features provided by the PVA protocol.

ACKNOWLEDGMENT

I would like to thank A.N. Johnson for his work on ensuring that PvaPy's build conforms to EPICS standards, M. Kraimer and M. Davidsaver for their work on prototyping support for PV unions, M. Kraimer for the development of pvaClientCPP package, K. Vodopivec for his early feedback and suggestions, as well as to R. Lange and D. Hickin for their work on automated builds and preparing software release. I would also like to thank N.D. Arnold and the entire EPICS 4 working group for their support and encouragements during PvaPy development.

REFERENCES

[1] G. White et al., "Recent Advancements and Deployments of EPICS Version 4", this conference proceedings; EPICS4 website: http://epics-pvdata.sourceforge.net;
[2] EPICS website: http://www.aps.anl.gov/epics
[3] Boost website: http://www.boost.org; Documentation for the Boost.Python module can be found at http://www.boost.org/doc/libs/release/libs/python

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

[4] Similar to other EPICS Version 4 modules, PvaPy project is hosted in GitHub: https://github.com/epics-base/pvaPy; Documentation corresponding to the most recent code can be found at http://epics-pvdata.sourceforge.net/docbuild/pvaPy/tip/pvaccess.html
[5] EPICS Base releases can be downloaded from http://www.aps.anl.gov/epics/download/base/index.php
[6] EPICS4 production releases can be found at http://sourceforge.net/projects/epics-pvdata/files
[7] Python website: http://www.python.org
[8] GNU website: http://www.gnu.org
[9] Sphinx website: http://sphinx-doc.org
[10] M.R. Kraimer et al., "EPICS Application Developer's Manual", Chapter 4. For EPICS Base 3.15.2, the manual can be downloaded from http://www.aps.anl.gov/epics/base/R3-15/2-docs/AppDevGuide
[11] GNU Autoconf project website: http://www.gnu.org/software/autoconf/autoconf.html
[12] GNU M4 project website: http://www.gnu.org/software/m4/m4.html
[13] PVA Client C++ project website (GitHub): https://github.com/epics-base/pvaClientCPP
[14] EPICS4 PV Data C++ v4.5.0 reference document: http://epics-pvdata.sourceforge.net/docbuild/pvDataCPP/4.5.0/documentation/pvDataCPP.html
[15] EPICS4 Normative Types specification document: http://epics-pvdata.sourceforge.net/alpha/normativeTypes/normativeTypes.html
[16] EPICS4 CloudBees Jenkins Build Server website: https://openepics.ci.cloudbees.com; APS Jenkins website: https://jenkins.aps.anl.gov;
[17] NumPy is Python package for scientific computing; http://www.numpy.org

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

HIGH LEVEL APPLICATIONS FOR HLS-II

K. Xuan#, G. Liu, C. Li, J. Wang, L. Wang, W. Li, J. Y. Li
National Synchrotron Radiation Laboratory, University of Science and Technology of China, Hefei, Anhui 230029, China

Abstract

The Hefei light source was overhauled beginning from 2010 and completed in the end of 2013. The new light source is renamed as HLS-II. A set of high level application tools, including physical quantity based control IOC, lattice calibration tools, orbit feedback, etc., were developed for the light source commissioning and operation. These tools have been playing important roles in the commissioning and operation of the light source. This paper reports some critical applications.

INTRODUCTION

The Hefei light source was overhauled between 2010 and 2013 using a completely new structure. The new light source is renamed as HLS-II. The HLS-II is comprised of an 800 MeV Linac and an 800 MeV storage ring. The installation of the light source components were finished in the end of 2013. The machine commissioning was started in the beginning of 2014. In January 2015, the HLS-II was brought into operation. In order to make the commissioning more effective, and achieve high performance operation, a set of high level application tools were developed based upon the Experimental Physics and Industrial Control System (EPICS) [1]. These tools have been playing important roles in both the commissioning and the operation of the HLS-II. This paper reports some of these tools and their applications.

PHYSICAL QUANTITY BASED CONTROL SYSTEM

The physical quantity based control system was first developed at the Duke FEL laboratory [2]. With necessary modifications, the HLS-II control system is developed using familiar software structure as the Duke FEL control system. The functional diagram of the control system is illustrated in Fig. 1. The HLS-II control system is comprised of three parts, IOCs for hardware controls, soft IOCs for conversions between physical and engineering quantities, and OPIs for displaying essential parameters, and running high-level tools and a number of simulation programs.

The conversion between physical and engineering quantities is realized by a soft IOC. In many cases in the light source commissioning and operation, a set of related quantities (records) need to be simultaneously processed to keep the electron beam undisturbed. This requirement is fulfilled using the EVENT mechanism of EPICS. For example, when the energy (ENG) of the storage ring is changed, a set of event with designated values are posted, and all the records related to the focusing strength of quadrupole magnets are processed as they receiving these events. This scheme has been well serving for the energy ramping and lattice adjustment of the storage ring.

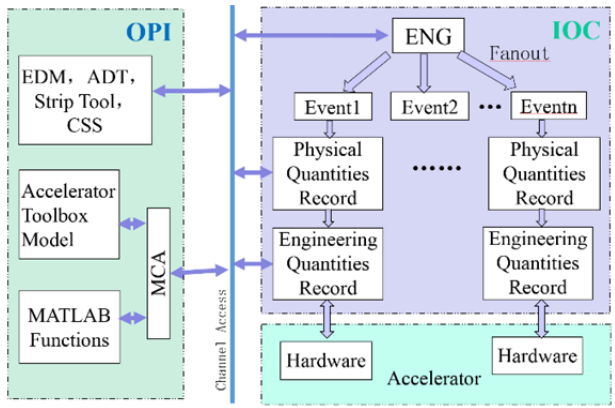


Figure 1: Functional diagram of the HLS-II control system.

The physical quantity of each component of the light source is usually divided into several parts for different purposes. For example, the total strength of a quadrupole magnet is given by

$$K = K_{design} + K_{adj} + K_{tune} + K_{comp}, \quad (1)$$

where K_{design} is the design value of that magnet, K_{adj} is used for adjustment, K_{tune} is used to change the transverse tunes, K_{comp} is used for lattice compensations. Another example is the kicking angle of an orbit corrector, its total kicking angle is composed of

$$\theta = \theta_{adj} + \theta_{feedback} + \theta_{comp}, \quad (2)$$

where θ_{adj} is used for adjustment, $\theta_{feedback}$ is used by the orbit feedback correction, and θ_{comp} is used to compensate residual field of some magnetic elements, such as undulators.

The physical quantity based control system of the HLS-II directly controls the physical quantities of the accelerator elements, including the magnetic field of various magnets, beam energy, and so on. This feature of the control system enables it to directly control the parameters of the accelerators and the electron beam. The physical quantities and related engineer quantities are automatically converted inside of the control system, and can be shared by different high level applications. This significantly improves the feasibility and effectiveness of the system. This system have been well serving the

xuanke@ustc.edu.cn

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

commissioning, machine study and operation of the HLS-II light source.

LATTICE CALIBRATION AND CORRECTION

The optical parameters of the HLS-II storage ring were found very different from their design values. This is mainly caused by the alignment and field errors of various magnets. The parameter variation seriously impacts the performance of the light source. In order to eliminate this effect, we perform lattice calibration and correction using a least square method. The matlab based program, Linear optics from closed orbits (LOCO) [3], originally developed in Stanford Linear Accelerator Center (SLAC) is used for lattice fitting in the calibration. The measured response matrix using correctors and BPMs is used as the input of LOCO. Based upon the LOCO fitting, beta functions of the storage ring are corrected using K_{comp} which is discussed in previous section.

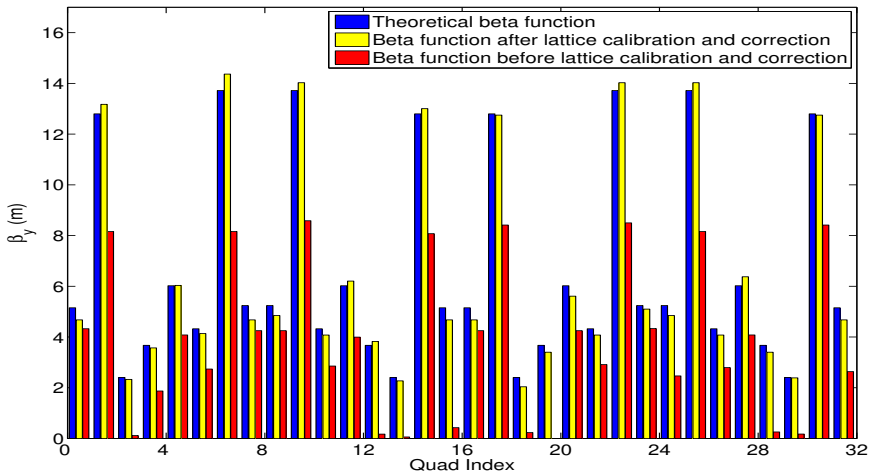


Figure 2: Design and measured vertical functions of the HLS-II storage ring.

Table 1: Beam Parameters Before and After Lattice Correction. These Parameters are Measured Using a 30 mA Electron Beam.

	Before Correction	After correction
Tune shifts	1.5%，7.3%	0.1%，0.1%
Beta beating	97.6%	6.8%
Lifetime	1.46 hrs	7.50 hrs

The beta functions of the storage ring were measured before and after the lattice correction for comparison. The measurements are based upon

$$\beta = \frac{4\pi\Delta\nu}{\Delta K l_{eff}}, \quad (3)$$

i.e. the beta function at the location a quadrupole magnet is calculated using the variation of the quadrupole strength and the measured tune shift. The measured vertical beta functions before and after the lattice correction are plotted in Fig. 2, respectively. The design vertical beta function is also plotted in the figure for comparison. This figure indicated that the beta function before the lattice correction is very different from the design value. The maximum beta beating is 97.6% comparing to the design one. After performing lattice correction, the measured beta functions are very close to the design ones. The maximum beta beating is close to our measurement accuracy (%5).

Some critical beam parameters were also measured and are listed in Table 1. The results show that the tune shift due to various errors is significantly reduced and the lifetime of the beam is increased from 1.46 hours to 7.50 hours.

This technique is also used in the lattice compensation for various insertion devices (IDs) installed in the storage ring, and achieved expected results.

The lattice calibration and correction significantly has improved the performance of the light source.

ORBIT FEEDBACK OF THE STORAGE RING

Orbit feedback of the storage ring is essential for high performance operation of the HLS-II. The orbit feedback system is comprised of four parts, the BPM system, corrector system, EPICS based control system, and a feedback program developed using matlab. The functional diagram of the system is illustrated in Fig. 3.

There 32 BPMs are installed in the storage ring. Button type electrodes are employed for beam position detection. Electronics modules manufactured by Librea are used for signal processing. 64 digital power supplies (PSs) are used to energize 32 horizontal and 32 vertical orbit

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

correctors, respectively. EPICS IOCs communicate with these PSs via fibre using RS232 protocol. There is a soft IOC used for controlling feedback parameters. These parameters are controlled and monitored during operation via an EDM based OPI control panel. The feedback program is written in matlab and running in the background on a Linux computer.

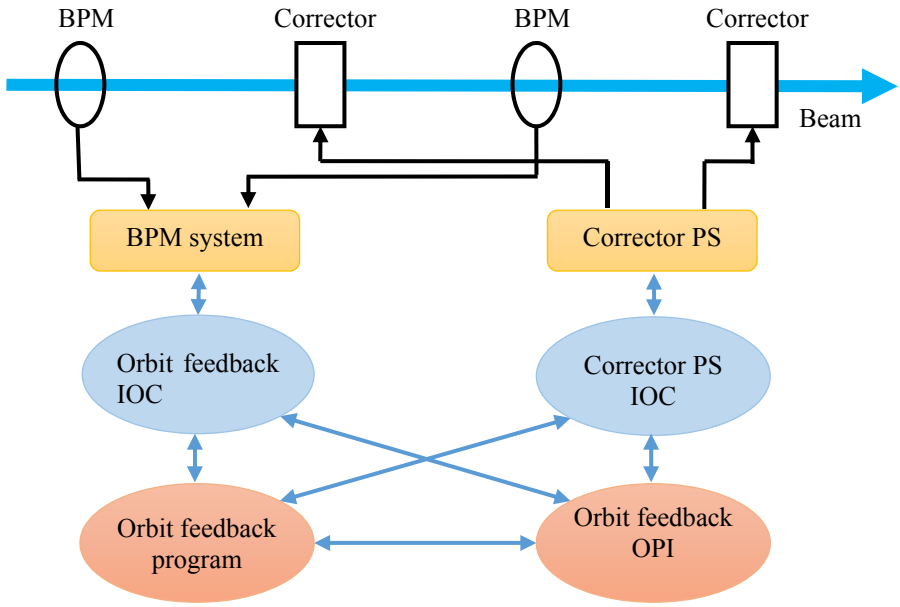


Figure 3: The functional diagram of the HLS-II orbit feedback.

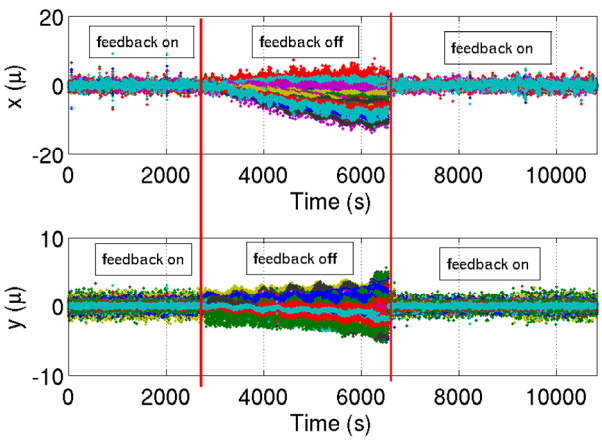


Figure 4: The HLS-II storage ring beam orbit stability with and without feedback, respectively. The values of each BPM are plotted using a designated color.

The effectiveness of the orbit feedback is test by monitoring the orbit variation with the feedback turned on and off, respectively. The results are shown in Fig. 4. During the test, the feedback is first turned on and the orbit is monitored for about 45 minutes. Then the feedback is turned off for about one hour. Finally the feedback is turned on again and the orbit is monitored for about 70 minutes. The results show that when the feedback is turned on, the orbit variation is within ± 5 μm, and the orbit have significant shifts when the feedback is turned off.

SUMMARY

The physical quantity based control system of the HLS-II can be used to directly control the physical parameters of accelerators. The high level applications developed based upon this system play important roles in the light source commissioning, and are providing strong supports for stable and high performance operation of the light source.

ACKNOWLEDGMENT

The authors would like to thank all our colleagues at NSRL who provide us technical supports. We also would like thank Prof. Y. K. Wu, Dr. H. Hao, Dr. S. Mikhailov and Dr. V. Popov at Duke FEL laboratory for their useful discussion and encouragement.

REFERENCES

[1] website: http://www.aps.anl.gov/epics.
[2] Y. K. Wu, S. Hartman, S. F. Mikhailov, "A physics based control system for the DUKE storage ring", PAC'03, Portland, Oregon, U.S.A, May 2003, p. 2482 (2003); http://www.JACoW.org
[3] J. Safranek, G. Portmann, A. Terebilo, "MATLAB-Based LOCO", EPAC'02, Paris, France, June 2002, p.1184; http://www.JACoW.org

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

USE OF TORNADO IN KAT-7 AND MeeRKAT FRAMEWORK

C. De Villiers#, B. Xaia‡, SKA South Africa, National Research Foundation of South Africa, South Africa

Abstract

The KAT-7 and MeerKAT radio telescope control systems (www.ska.ac.za) are built on a rich Python architecture. At its core, we use KATCP (Karoo Array Telescope Communications Protocol), a text-based protocol that has served the projects very well. KATCP is supported by every device and connected software component in the system. However, its original implementation relied on threads to support asynchronous operations, and this has sometimes complicated the evolution of the software. Since MeerKAT (with 64 dishes) will be much larger and more complex than KAT-7, the Control and Monitoring (CAM) team investigated some alternatives to classical threading. We have adopted Tornado (www.tornadoweb.org) as the asynchronous engine for KATCP. Tornado, popular for Web applications, is built on a robust and very efficient coroutine paradigm that in turn is based on Python's generators. Co-routines avoid the complexity of thread re-entrancy and lifetime management, resulting in cleaner and more maintainable user code.

This poster will describe our migration to a Tornado co-routine architecture, highlighting the benefits and some of the pitfalls and implementation challenges we have met.

KATCP IN THE KAT-7 AND MEERKAT SYSTEMS

KATCP [1,2] is a simple ASCII communication protocol layered on top of TCP/IP.

It has been developed as a part of the Karoo Array Telescope (KAT) and MeerKAT projects and used at SKA South Africa for the monitoring and control of hardware devices. In this role it has been very successful and the specification is currently at Revision 5.

The original KATCP implementation provided a blocking client and a non-blocking CallbackClient.

Base message types are Request, Reply and Inform - the latter are sent asynchronously by a server to provide out-of-band data or (in some cases) to provide a way of segmenting the results of a previous request.

KATCP additionally defines a software Sensor type. Sensors are created with names and data types such as float, string etc. Dynamically a sensor may acquire a value and a status which it communicates to its registered listeners via callbacks. A listening client may set a strategy on the sensor which causes it to push its value and status to the listener periodically or on certain events, such as a value change. An ad-hoc query mechanism is also available.

KATCP messages, sensors, servers and clients are the building-blocks of the KAT-7 and MeerKAT Control and Monitoring systems. These robust abstractions support the next layer of the architecture, which comprises software proxies to abstract access to real hardware, and components that partition the work of system startup and shutdown, scheduling, observation control and monitoring.

LIMITATIONS OF THREADS

In earlier implementations of KATCP, the inherent concurrency of real-time processes was modelled using software thread. Threads provide the illusion of parallel execution within a single processor core by performing pre-emptive task switches at the system level. This has benefits for processing efficiency since the system need never be idle - while some thread of control is awaiting an event, another thread can be executing. This can be especially useful in networked systems where many cycles would otherwise be wasted waiting for I/O events.

However threads also have some well-known drawbacks. Each thread has its own execution context and this means that threading is resource-intensive, so that the number of active threads must be limited. Perhaps even more important is the complexity they introduce into software design. Since any thread of control may be interrupted or resumed at essentially arbitrary moments, software becomes ‘non-linear’ and the designer must carefully guard against inadvertent corruption of shared resources. Numerous best-practices and software constructs exist to alleviate these problems, but all contribute to the complexity and cost of software development and maintenance.

Finally the Python language, which has proved immensely valuable in the development of our systems, implements a Global Interpreter Lock (GIL) which essentially disables threading for compute-bound tasks on a single processor.

MIGRATION TO TORNADO

Like many teams facing these challenges, we have been interested in the developments in coroutine-based concurrency frameworks. Coroutines are a generalisation of subroutines based on co-operative multitasking, in contrast to the pre-emptive model used by threads Coroutines differ from subroutines in allowing multiple entry-points (and multiple entries per entry-point) within the body of a routine, with the preservation of the full execution context at that point. Because task-switching is

#charles@ska.ac.za
‡ bxaia@ska.ac.za

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

cooperative, the developer can tell by inspecting the code where a context switch may occur. This makes it much easier to ensure and verify the determinism of the code. Task switches are 'lightweight' because only the normal stack mechanism is required to save and restore context. All coroutines in a process typically run within a single execution thread.

Coroutines are an important tool to support lightweight concurrency, but for a large system one also needs a scheduling mechanism so that independent subtasks can execute without mutual awareness. This is achieved through a coroutine framework. After some research we chose the open-source Tornado framework.

Tornado[3] is a Python web framework and asynchronous networking library, and offers non-blocking I/O and concurrency support via coroutines. It is capable of scaling to tens of thousands of open connections and concurrent handlers. In addition it provides a scalable, non-blocking Web server and application framework. Although Web development is not our primary focus, the MeerKAT GUI is web-based, and HTTP servers are also proving useful in other areas. Tornado allows for multiple long-lived client connections with minimal overhead.

The MeerKAT GUI displays have been completely re-engineered using Tornado and other modern technologies such as AngularJS. The MeerKAT GUI displays real-time data from the back-end components. Using the Tornado web server and websockets, tests have shown that it can comfortably handle multiple, concurrent, long-lived connections from components as well as human users. Additional libraries and adapters have eased integration, such as toredis (a Redis client on top of Tornado), sockjs_tornado (WebSocket emulation), etc.

KATCP Implementation using Tornado

An objective of our adoption of Tornado was to replace the use of threading throughout the codebase. Because the original KATCP client and server base classes were thread-based, this was the natural starting-point for the implementation.

The Tornado scheduler is called the 'ioloop'. Every component and activity requiring scheduling must have a reference to the ioloop, This reference may be obtained from the global execution context, or passed in as a parameter. The latter method allows for a 'local' ioloop to be used in specific cases.

A particular challenge of the CAM implementation was our large legacy codebase, which has many instances of operations expecting synchronous (blocking) responses. The Tornado ioloop mechanism, however, is non-blocking and returns Futures - placeholders for the result of possibly incomplete operations. Obtaining the result of the operation that returns a Future requires the use of the yield keyword within a coroutine. On encountering this construct, the ioloop engine suspends the current operation and continues with the execution of other coroutines until the result of the Future is available. Then the original stack context is restored and the function continues with the result it has obtained.

KATCP proxies and other top-level components typically run within their own threads or processes to minimize I/O contention. To facilitate the transition from thread-based to coroutine-based concurrency, a compatibility layer was added to KATCP. Some of the classes in this compatibility layer are briefly described below.

The IOLoopManager helper class provides a facade for an ioloop instance that may be shared across components, or running in a private thread. This class exists to abstract this difference and to guard against inadvertent thread contention.

Even when a section of the codebase has been converted to coroutines, we often need a way to return a synchronous result for clients that expect this, and that may be running within their own arbitrary threads. This has been achieved by implementing Python code decorators within our custom compatibility layer: the *_make_threadsafe()* wrapper ensures that arbitrary code is executed within the ioloop's own thread, while the *_make_threadsafe_blocking()* decorator additionally guarantees that the Future's result will be resolved before it is returned to the blocking caller. A DeviceClient instance may call *enable_thread_safety()* before it is started, in order to apply the relevant decorators to all its methods. The whole instance thus becomes implicitly threadsafe and/or blocking, and hence suitable for integration with legacy code.

A feature of KATCP from its origins was introspection; a client can query any device on the network and obtain a specification of its interface (requests and sensors). On connection. the client builds a local representation of the server's interface, and proxies those capabilities using Python's dynamic binding. This feature is implemented in the InspectingClient class. The InspectingClient monitors its own connection and synchronisation state and attempts to re-initialise itself if the connection is lost or the server indicates an interface change. The Inspecting Client in turn may be included in a higher-level container, where it can be commanded and interrogated by an interactive user or a script.

Figure 1 shows a conceptual view of the CAM software layers involved in the Tornado integration.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

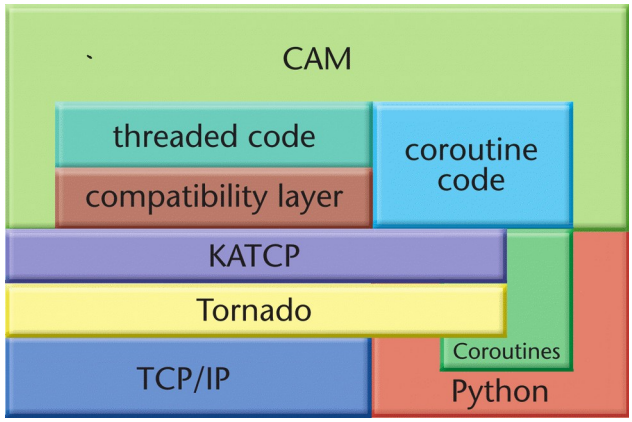


Figure 1: CAM Software Layers.

CONCLUSION

Our adoption of Tornado has brought a number of benefits as well as some challenges.

Benefits

Tornado in CAM has begun to deliver on its promise of efficient multitasking. This is especially evident in areas of our codebase that have been fully converted to the Tornado idiom, such as the sensor-archiving component KatStore and the web-based GUI. Tornado has its own testing framework (built on the standard Python unittest) and this is helping us to eliminate the thread-management challenges that used to hamper our testing. Code at the application level is greatly simplified by the elimination of the complex locking and concurrency control that were necessary for safe threading. This in turn can only benefit reliability and maintainability in the long run.

Challenges

Coroutines and the event loop were an unfamiliar paradigm to most of the team. It takes time to fully understand coroutines and futures, and to recognise the ways in which existing code must change to accommodate them. Despite the compatibility tools added to KATCP, a significant effort has been required to integrate the changes with the rest of the system, and many unit tests initially failed because of explicit or implicit threading dependencies. Some components and some tests have still to be converted.

Debugging can be difficult because of the many layers of 'scaffolding' code introduced by the framework; of course any concurrent model has similar problems. Better tools may help here.

ACKNOWLEDGEMENTS

KATCP was developed as part of the Karoo Array Telescope (KAT) project.

Tornado was developed at FriendFeed.

The Tornado implementation of KATCP was mainly implemented by Neilen Marais as part of the MeerKAT project.

REFERENCES

[1] KATCP documentation,
website: https://pythonhosted.org/katcp/
[2] KATCP GitHub Repository,
website: https://github.com/ska-sa/katcp-python
[3] Tornado documentation,
website: https://tornado.readthedocs.org/en/stable/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

BUNCH TO BUCKET TRANSFER SYSTEM FOR FAIR

J.Bai[1,2], T.Ferrand[1,3], D.Beck[1], R.Bär[1], O.Kester[1,2], D.Ondreka[1], C.Prados[1],W.Terpstra[1]
1. GSI Helmholtzzentrum für Schwerionenforschung, Darmstadt, Germany
2. IAP, Goethe Universität Frankfurt, Frankfurt am Main, Germany
3. Technische Universität Darmstadt, Darmstadt, Germany

Abstract

For the FAIR accelerator complex, synchronization of the bunch to bucket (B2B) transfer will be realized by the General Machine Timing system and the Low-Level RF system. Based on these two systems, both synchronization methods, the phase shift and the frequency beating method, are available for the B2B transfer system for FAIR. This system is capable to realize the B2B transfer within 10ms and the precision better than 1° for ions over the whole range of stable isotopes. At first, this system will be used for the transfer from the SIS18 to the SIS100. It will then be extended to all transfers at the FAIR accelerator facility. This paper introduces the synchronization methods and concentrates on the standard procedures and the functional blocks of the B2B transfer system.

INTRODUCTION

FAIR is aiming at providing high-energy beams with high intensities. Based on the existing GSI accelerators UNILAC and the SIS18 serving as an injector, high intensity ion beams over the whole range of stable isotopes will be accelerated in the new heavy ion machine SIS100 to higher energies. The FAIR new accelerator complex with storage rings will in the full version consist of the SIS100, the Collector Ring CR, the accumulator/decelerator ring RESR and the New Experimental Storage Ring NESR. An additional High Energy Storage Ring HESR serves experiments with high energy antiprotons. The B2B transfer means that one bunch of particles, circulating inside the source synchrotron, is transferred into the center of a bucket of the target synchrotron. For the FAIR project, there are many transfers involving the bunch to bucket (B2B) transfer. E.g. the B2B transfer from the SIS18 to the SIS100, from the SIS18 to the ESR, from the SIS100 to the CR, from the CR to the HESR and later to RESR.

For the FAIR accelerator complex, synchronization of the B2B transfer system will be realized by General Machine Timing (GMT) system [1] and Low-Level RF (LLRF) system) [2].

The main tasks of the GMT system are time synchronization of more than 2000 nodes with nanosecond accuracy, distribution of timing messages and subsequent generation of real-time actions by the nodes of the timing system. The GMT consists of a Data Master (DM), a Clock Master (CM), the White Rabbit (WR) timing network and integrates nodes [3]. The DM schedules actions by broadcasting messages, which will be received and executed by the corresponding node at the designated time.

For the synchronization of LLRF system, the GMT system is complemented and linked to Bunchphase Timing System (BuTiS) [4]. BuTiS is a campus wide clock synchronization and distribution system, locally generating two delay compensated high precision clock signals with a jitter of 10 femtosecond.

Dependent on the BuTiS, the B2B transfer system at FAIR is distinguished from others. For CERN and other accelerator facilities, the B2B transfer system is based on the cavity measured signals [5]. Instead for FAIR, it is based on the driven signals directly derived from the BuTiS.

METHODS OF SYNCHRONIZATION TWO SYNCHROTRONS

For the proper transfer, the phase advance between the bunch of the source synchrotron and the bucket of the target synchrotron must be precisely controlled before the bunch is ejected. The process of achieving the detailed phase adjustment is usually termed "synchronization". There are usually two methods available for the synchronization process, the phase shift method and the frequency beating method.

A short description of the phase shift method is as follows. At a scheduled time well before ejection, the phase of the beam in the source synchrotron and the phase of the bucket in the target synchrotron are measured with respect to the phase of a common central reference signal, which is synchronously distributed to the source and target synchrotrons. Based on the measured phase, the radio frequency (RF) frequency of the source or target or both synchrotrons is modulated away from the nominal value for a period of time and then modulated back so that the phase shift created by the frequency modulation could compensate for the expected phase difference. After the phase shift, the bunches can be injected into buckets. If the RF frequency of the source and target synchrotrons are same, it brings an infinite time frame in ideal situation beginning from the end of the phase shift process, within which the bunches can be injected into buckets at any time. This is the so called synchronization window for the phase shift method (See Figure 1). The phase shift process must be performed adiabatically for the longitudinal emittance to be preserved [6]. But when the target synchrotron has no particles, the phase shift can be done for the target synchrotron without adiabatical consideration (e.g. Phase jump).

The frequency beating method uses the effect of two RF signals of slightly different frequencies, perceived as periodic variations in phase difference whose rate is the difference between the two frequencies. The RF frequency of the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

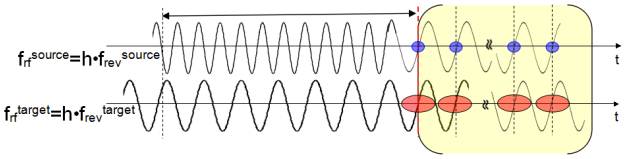


Figure 1: The phase shift method.

source or the target or both synchrotrons is detuned long before the ejection, then the phase difference between the phase of the bunch/bucket and the reference signal is measured. Based on the measured phase, the synchronization is realized when the phase difference of two RF frequencies corresponds to the expected phase difference. Because of the slightly different RF frequencies, there exists the mismatch between the bunch and bucket centers. The requirement for this mismatch is better than 1°, which brings a symmetric time frame with respect to the time of the expected phase difference, this is the maximum synchronization window for the frequency beating method (see Figure 2). During the detune process, the energy of the beam should not be affected.

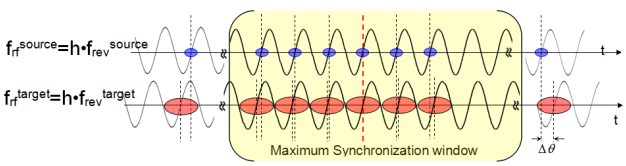


Figure 2: The frequency beating method.

There is a preference to start with the frequency beating method at FAIR, since it is easier to be realized. Many source and target synchrotrons begin the beating automatically, because the ratio of the circumferences is not a perfect integer.

STANDARD PROCEDURES FOR THE B2B TRANSFER

The standard procedures for the B2B transfer system is the same, no matter which synchronization method is chosen. Figure 3 illustrates the standard procedure from the viewpoint of the accelerator cycle. The top part shows the chronological steps of the frequency beating method, according to which the RF frequency is detuned during the acceleration ramp. The bottom part shows the steps of the phase shift method, in this case the phase shift is done after the RF frequency reaches the flattop. The synchronization window must be advanced by the kicker delay in cables, electronics, kicker preparation time and so on. The emergency kickers can be triggered at any time during the acceleration cycle by the machine protection system (MPS), which indicates the status of the synchrotron. The B2B transfer process basically needs to follow six steps (see Figure 3):

1. Announce the B2B transfer and freeze the stabilization system.

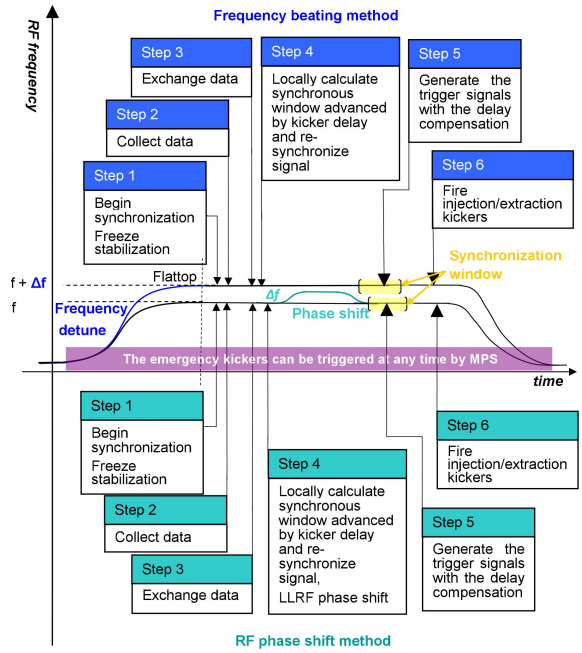


Figure 3: The procedure for the B2B transfer within one acceleration cycle.

2. Collect data from the source and target synchrotrons.
3. Exchange data between source and target synchrotrons.
4. Calculate the synchronization window locally in both synchrotrons with an advance of the kicker delay and re-synchronize RF signal. For the phase shift method, the phase shift process is done.
5. Generator trigger signals for the kickers.
6. Fire the kickers by the kicker electronics.

FUNCTIONAL BLOCKS OF THE B2B TRANSFER SYSTEM

The most important B2B transfer components are specified from a functional point of view. Figure 4 shows the functional block diagram. Two blue boxes show the supply room of each synchrotron.

RF Phase Measurement Module

The RF phase measurement module in each supply room is used to measure the phase difference $\Delta\varphi$ between the reference signal and the dedicated RF signal (group Direct Digital Synthesis (DDS) [2]) of each synchrotron. A reference RF signal is generated locally at each supply room. It is a sine wave, whose frequency is a multiple of the BuTiS T0 and whose reset is triggered by the T0 in order to ensure the synchronization of the reference signals in different supply rooms. The group DDS serves as reference signals for the RF cavities.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

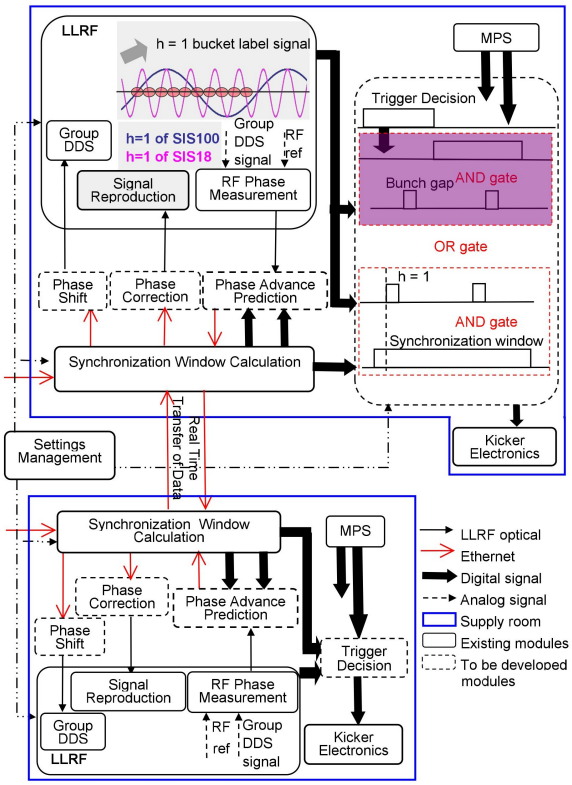


Figure 4: The functional block diagram of the B2B transfer system.

Phase Advance Prediction Module

It makes use of a series of $\Delta\varphi$ measurements from the RF phase measurement module to predict the phase difference after any delay. In addition, it is synchronized to the BuTiS T_0 and C2 [4].

Synchronization Window Calculation Module

This module is responsible for the data exchange between two synchrotrons and the synchronization window calculation. It is also in charge of the start of the synchronization process.

Phase Correction Module

The phase correction module makes use of the phase difference information exchanged between two supply rooms of the synchrotrons to eventually determine a dedicated phase offset and transmits it to the signal reproduction module.

Signal Reproduction Module

The signal reproduction module duplicates the RF signals of the desired harmonic of the other synchrotron and precisely adjusts the phase according to the shared phase information from the phase correction module. It provides the bucket label signal. (e.g. RF h = 1 signal of the SIS100 at the SIS18)

Phase Shift Module

The phase shift module makes use of the the restrictions from the front end controller (FEC) [3] and of the expected phase shift to define a corresponding phase modulation profile. And than it furnishes the group DDS module with phase steps to achieve the expected phase shift.

Group DDS Module

Each cavity is driven by a local DDS unit, cavity DDS [2]. The group DDS receives a revolution frequency ramp from the central control system (CCS) and derives all the needed RF signals of the desired harmonics, one of which is selected as a reference RF signal for the cavity DDS. This cavity DDS receives its phase correction by comparison of the cavity gap signal and the reference RF signal. So the phase of the cavity gap signal is locked to that of the selected reference RF signal. This is realized by the so called the beam phase control [2]. In the case of the phase shift method, the phase shift module may be implemented in an additionally beam phase control for the group DDS module to achieve the expected phase shift.

Real Time Data Transfer

The WR network [1] is responsible for the real time date exchange between two synchrotrons.

Trigger Decision Module

This module is used to produce the trigger signal for the kicker electronics for regular extraction/injection and the emergency extraction.

Kicker Electronics Module

After receiving the trigger signal from the trigger decision module, the kicker electronics produces kicker pulse with the specified width for the regular extraction/injection or the emergency extraction.

CONCLUSION

At first, the B2B transfer system will be used for the transfer from the SIS18 to the SIS100. It will then be extended to all transfers at the FAIR accelerator facility. The B2B transfer system is also available for the standard case of extracting the beam from a synchrotron into a fixed target, like the beam and target collision of the Compressed Baryonic Matter (CBM) experiment.

ACKNOWLEDGMENT

The author would like to thank all staff in the department of control systems of GSI for their patient guidance, enthusiastic encouragement and useful critiques of the research work. The author would also like to thank staff from the Department Ring RF Systems to provide useful information about the LLRF system.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REFERENCES

[1] D. Beck, et al., *The new white rabbit based timing system for the FAIR facility*, PCaPAC'12, India, p. 242, (2012).

[2] H. Klingbeil, et al., *New digital low-level rf system for heavy-ion synchrotrons*, Phys. Rev. ST Accel. Beams, 102802 (2011).

[3] D. Beck, et al., *The General Machine Timing System for FAIR and GSI* GSI.

[4] P. Moritz, *BuTiS - Development of a Bunchphase Timing System*, GSI Scientific Report 2006, p. 64, (2007).

[5] P. Baudrenghien, et al., *SPS Beams for LHC: RF Beam Control to Minimize Rephasing in the SPS*, 6th European Particle Accelerator Conference, Stockholm, Sweden, p. 1702, (1998).

[6] P. Baudrenghien, et al., *Beam-Dynamics View of RF Phase Adjustment for Synchronizing J-PARC RCS with MR or MLF*, KEK Internal, p. 24, (2008).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TIMING SYSTEM AT MAX IV - STATUS AND DEVELOPMENT

J. Jamroz, V. Hardion, V. Martos, J. Forsberg, D. Spruce, MAX-IV, Lund, Sweden

Abstract

A MAX IV construction of two storage rings (SR1.5GeV and SR3GeV) and a short pulse facility (SPF) has been proceeding over last years and will be finished in the middle of 2016. In 2014, few timing procurements were successfully finalized according to the MAX IV requirements (see [1] for details) and the installation works are ongoing along with the TANGO control system integration. The design covers the timing synchronization (TIM) and acquisition, fast orbit feedback (FOFB), fast machine protection system (FMPS) and integration of MAX IV operation modes. The LINAC commissioning started in 2014 and was successful together with the first beam line (FEMTOMAX). The SR3GeV commissioning started in August 2015 and is ongoing. The SR1.5GeV is being installed and will be commissioned in 2016.

MAX IV OPERATION MODES

MAX IV will work in 4 different modes:

- LINAC 3GeV (LIN)
- LINAC 3GeV + Short Pulse Facility (SPF)
- LINAC 1.5GeV + Storage Ring 1.5GeV (SR1)
- LINAC 3GeV + Storage Ring 3GeV (SR3)

LIN and SPF are rated for 100Hz injection whereas SR1 and SR3 for 10Hz.

Each mode is synchronized to its own radio frequency (RF). LIN and SPF are driven by 39 harmonics of the LINAC master oscillator (MO) 3GHz (2998500000/39=76884615.38Hz in reality, around 77MHz). The 39 harmonic is required for a synchronization of two laser systems, one at the beginning of the LINAC, other 400m away in the short pulse facility. This implementation also keeps the same constant phase for other frequencies: 100Hz LINAC injection trigger, 1kHz laser pumping frequency etc.

SR1 is locked to 100MHz (RF1) and SR3 is locked to 100MHz (RF3) but those frequencies are a bit different as the real value comes from the construction and temperature parameters of storage rings. Both rings work in the same way, each SR TIM generates a LINAC injection trigger which is locked to its RF and in constant phase to its machine clock (MC). The MC phase shift (10ns step) defines the SR bucket number. SR3 stores 176 electron buckets and its MC is equal to 568kHz, whereas SR1 stores 32 electron buckets and its MC is equal to 3.12MHz.

On top of that, the synchronization to mains electricity (ME) 50Hz will be implemented, so the injection will always be applied under constant parameters of high power devices.

In summary, 3 different LINAC triggers are delivered:

- LIN and SPF – TRIG0 (3GHz, 77MHz, 50Hz),
- SR1 – TRIG1 (100MHz, 3.12MHz, 50Hz),
- SR3 – TRIG3 (100MHz, 568kHz, 50Hz).

The brackets include a set of frequencies which are synchronized one other. The future timing improvement considers adding 3GHz master oscillator (MO) to TRIG1 and TRIG3 (see “MO 3GHz SYNCHRONIZATION” section for more information).

SWITCHING MODES

Switching over each operation mode has not been defined yet.

Main assumptions:

- The LINAC trigger runs 24/7 regardless other modes, so LINAC devices keep constant parameters (temperature etc.).
- Max LINAC death time (no trigger) around 5 second.
- To avoid an overcurrent injection to the nominal stored beam (500mA) in SR1 and SR3 without dumping the beam (BEAMDUMP).
- To avoid an overcurrent injection to a single storage ring bucket, so not to excess the nominal 5nC/bucket.
- Full automatic operation, one instruction/button of the operator should be enough to run the machine.
- Once SR1 and SR3 are “full”, SPF/LIN mode should be activated (nominal conditions).
- Except TIM, other devices (which are much slower) will be a part of the switching mechanism (e.g. SR1 and SR3 LINAC bending magnets etc.).
- Each timing system (SPF, SR1, SR3) has to operate autonomously 24/7 regardless the actually selected mode, delivering acquisition triggers, beam pinging, MC, FMPS functionality etc.

CORE HARDWARE

Beam Position Monitors (BPM-s)

The BPM system is based on Libera Brilliance+ which is a uTCA 2U box containing:

- 1x timing module (EVRX) - connected to the timing system via an optical link,
- 3x or 4x BPM processor unit - connected to RF pick ups in the storage rings,
- 1x gigabit data exchange module (GDX) which exchanges the beam orbit data/calculations,
- 1x RS485 card controlling power supplies which are connected to ring corrector magnets,
- 1x CPU module running Linux and connected to Ethernet.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

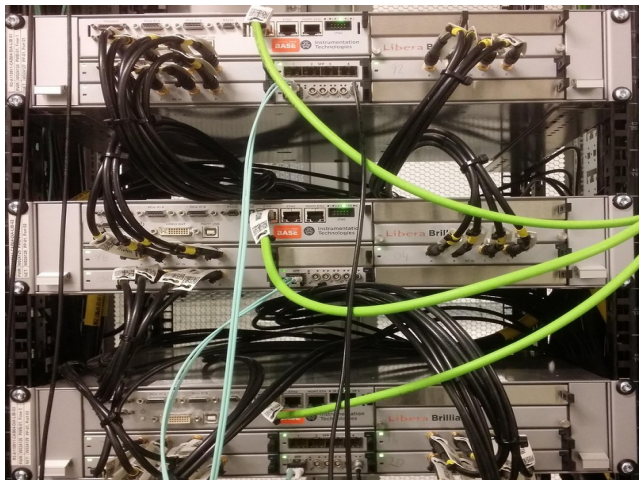


Figure 1: One generic BPM rack in SR3

The real example of one BPM set is presented in Figure: 1.

EVRX

The card processes acquisition triggers, machine clock, postmortem signals and generates interlocks (optical and electrical). Initially, the optical link compatibility was not supported and a dedicated firmware development has been requested to Instrumentation Technologies (I-Tech). It was finished successfully in June 2015 and it is during a validation process which will be finished at the end of 2016.

This improvement added new features to:

- FMPS: BPM interlock response time in microsecond range, propagation of the interlock signal over the whole machine in less than 1ms.
- TIM fast acquisition: possibility to implement higher amount of trigger signals (event codes) than 3 standard electrical inputs.
- TIM gets deterministic access to the BPM fast acquisition which is a base of a fast orbit feedback (FOFB).
- TIM gets access to control absolute time of BPM-s.
- TIM and FMPS can drive external devices in the deterministic way via EVRX electrical inputs/outputs.

BPM Processor

Standard features cover MAX IV needs. In total, 200 BPM-s were installed in SR3, driven by 117.53MHz sampling clock and performing 10.139ksps fast acquisition. In case of SR1, 36 BPM-s were installed, driven by 118.67MHz sampling clock and performing 10.074ksps fast acquisition.

The generated data is a base for the future FOFB implementation.

GDX

The GDX module is a core of FOFB system. It interconnects in series all the Libera boxes via an optical link, making a full data circle around SR1, SR3 and performs beam orbit correction calculations. Requirements were defined in early 2014 and the optical installation was finished in March 2015. The I-Tech company delivered a MAX IV specific implementation which will be a base of a future FOFB integration. The development will be done at the end of 2016 (or in 2017). Except that, the module will deliver around 3.5 MB/s of global magnet data and 31 MB/s of global orbit data to the control system.

RS485 Module

The module has been developed upon request. The RS485 protocol is compatible to “Innovative Test Systems” (ITEST) corrector power supplies which will be driving corrector magnets with the 10kHz update rate of the FOFB set point.

CPU Module

A generic computer which contains: 2x CPU N270 @ 1.60GHz, 1GB RAM with Ubuntu 10.04.4 LTS operating system and provides the distributed control system – TANGO (see [2]).

TIM – Micro-Research Finland (MRF)

The timing procurement has been awarded to MRF company at the end of 2014. The hardware components were delivered in March 2015. Except that, two additional procurements were issued:

- “Fibre backbone for Timing system” - material and installation, covering TIM and FOFB requirements, awarded to COROMATIC and delivered in March 2015.
- “Compact PCI standard for MAX IV” awarded to “RECAB Embedded Computers” and delivered in April 2015.

In summary, the TIM implementation consists of MRF cPCI cards based on 8b/10b 2.5Gb/s optical link, OM4 fiber optic cables and generic cPCI components. A real installation example is presented in Figure: 2.

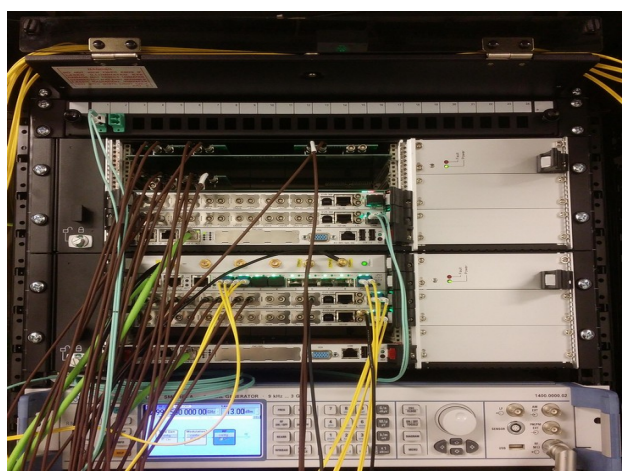


Figure 2: Main LINAC timing rack.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PROTECTION/SAFETY SYSTEMS

General Implementation

A machine protection system (MPS) and personal safety system (PSS) are based on programmable logic controllers (PLC-s) whereas the fast machine protection system (FMPS) uses timing cards based on field-programmable gate arrays (FPGA-s).

MPS monitors conditions of: cooling water, chamber and magnet temperatures, power supplies, vacuum and system interlocks, protecting locally devices/systems and globally the machine. PSS manages conditions related to a human protection and blocks user actions if safety requirements are not resolved.

MAX IV had two main protection challenges: LINAC and storage rings.

LINAC

The LINAC MPS is simpler than storage rings as except the general conditions, the LINAC beam is not stored. The LINAC MPS response time constrain is in range of 100ms and the standard PLC-s cover all the requirements. In addition, the main LINAC trigger can be only one and has to be selected for a particular operation mode, the trigger integration is presented in Figure 3.

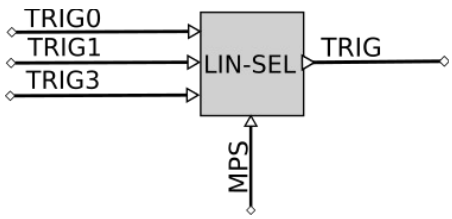


Figure 3: LINAC trigger selector with the MPS functionality

The selector is controlled by MPS which monitors the current conditions of each storage ring, blocking the TRIG, or changing the operation mode, if the stored current for the particular mode reached the set limit (nominal value), or in case if the operation mode was not applied properly (e.g. some devices could not be enabled because of a failure).

Storage Rings

FMPS is mainly important for the storage rings as 3GeV 500mA beam outside its orbit can damage the chamber in few milliseconds, moreover if a vacuum pollution appears in any beam line, a protecting fast closing valve can be melted in 3ms by the photon beam produced by an insertion device. The main FMPS requirement is to dump the beam in less than 3ms. The storage ring MPS implementation is presented in Figure: 4. The green line presents a monitoring bus by TANGO. The orange line shows the LINAC trigger disable capability. The red line is the most critical because triggers the BEAMDUMP. The MPS BEAMDUMP acts directly on RF transmitters via slow but reliable PLC contacts whereas FMPS acts on low level RF system (LLRF) which is instant but without switching off the plant which is powered off 100ms later by MPS-PLC as the same alarm signal is exchanged between these two systems. The red signal is also a source for the postmortem (PM) acquisition (blue line) which can be triggered by FMPS, MPS-PLC and TANGO.

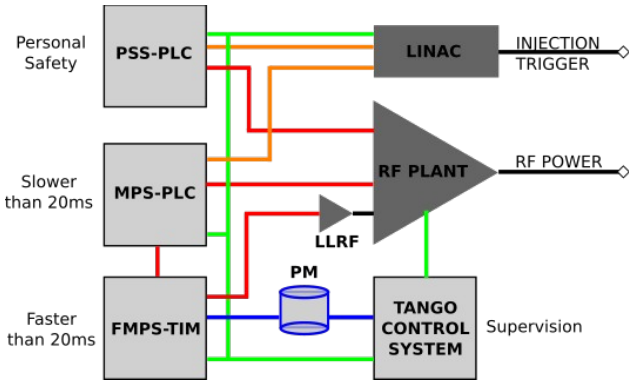


Figure 4: MPS flow of storage rings - general block diagram.

TANGO INTEGRATION

One global timing graphical user interface (GUI) is under definition and will be implemented in 2016. Nevertheless, all the timing components/nodes have been included in TANGO and the commissioning is proceeding by utilizing standard TANGO tools (see [2] for more information).

General

The timing components are presented in Figure: 5.

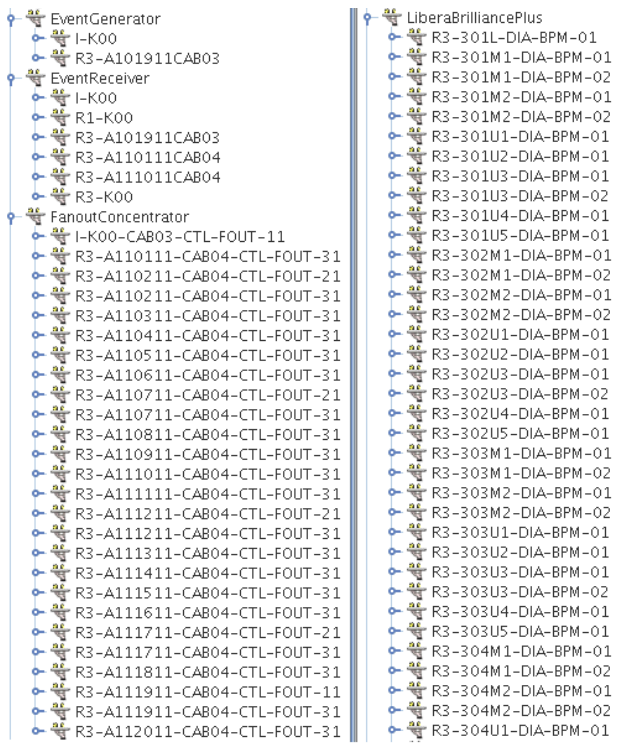


Figure 5: A JIVE (a TANGO database browser) representation of the timing system: MRF TIM (SR3, 20 sectors) on the left, Libera BPM (SR3, 4 sectors) on the right.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

State Grid

All the timing components have been included to be monitored in a state grid for a future system reliability estimation. The implementation is presented in Figure 6. The state grid is a GUI application intended to give the operator an overview of the current condition of the control system. It presents the TANGO states of a set of devices, e.g. "ON", "OFF", "ALARM", "FAULT", etc. The state grid relies on a specialized TANGO device that runs in the control system, listening to changes in state for a configured set of devices. The grid application receives updates from this device whenever a state changes.

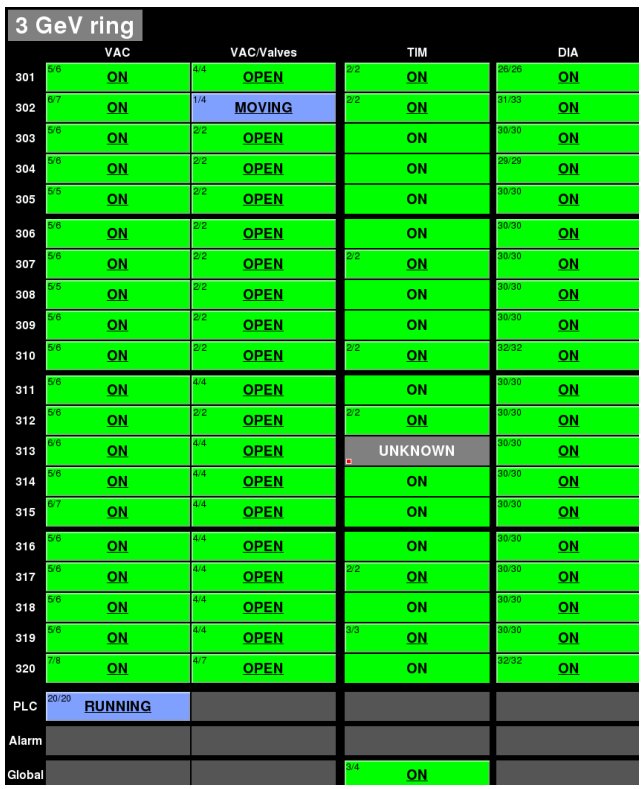


Figure 6: A segment of the SR3 state grid.

MO 3GHZ SYNCHRONIZATION OF THE LINAC TRIGGER FOR STORAGE RING INJECTIONS

The future improvement of the MAX IV timing is related to the LINAC trigger (TRIG1 and TRIG3) which is not synchronized to the LINAC 3GHz MO which is used to produce electron bunches in the thermionic gun. Around three (from 2 up to 3.5) electron bunches (330ps period) are chopped by the LINAC chopper synchronized to RF1 or RF3 and construct a charge for one SR bucket (see [3] for details). This solution has a drawback as each LINAC shoot has a different charge per bucket.

The potential solution is presented by the formula:

- MO+TRIG(RF,MC,ME)=TRIG(MO,RF,MC,ME).

The idea is to design a device with “two inputs” (for MO and TRIG) which synchronizes the TRIG phase to MO. The MO signal is a 3GHz sinus and the TRIG signal is a TTL pulse (width adjustable).

If someone has an idea how to solve this issue, please email the paper author.

REFERENCES

[1] J. Jamroz et al., “TIMING SYSTEM AT MAX IV”, THPPC103, Proceedings of ICALEPCS2013, San Francisco, CA, USA, (2013)

[2] TANGO control system, http://www.tango-controls.org/

[3] D. Olsson et al., “A chopper system for the MAX IV thermionic pre-injector” http://dx.doi.org/10.1016/j.nima.2014.05.052

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

OPERATION STATUS OF J-PARC TIMING SYSTEM AND FUTURE PLAN

N. Kamikubota#, N. Kikuzawa, F.Tamura, and N. Yamamoto
J-PARC-Center, KEK & JAEA, Tokami-mura, Ibaraki, Japan

Abstract

The beam commissioning of J-PARC started in 2006. Since then, the timing system of J-PARC has contributed stable beam operation of accelerators. The present timing system is reviewed from the aspects of history, fiber-optic cable network, VME modules and their configurations, followed by upgrade studies for the future.

INTRODUCTION

J-PARC (Japan Proton Accelerator Research Complex) is a high-intensity proton accelerator complex, located in Ibaraki, Japan. It consists of three accelerators: a) 400-MeV linac (LI), b) 3-GeV Rapid Cycling Synchrotron (RCS), and 30-GeV Main Ring (MR), and three experimental facilities: d) Material and Life Science Experimental Facility (MLF), e) Neutrino Experimental Facility (NU), and f) Hadron Experimental Facility (HD) [1-3].

In J-PARC, there are two time cycles. The rapid cycle, 25 Hz, is used at LI, RCS and MLF. Through LI was designed to be operated at the 50-Hz repetition rate, current operation is 25 Hz. The slow cycle is used at MR, NU and HD. In 2015, when MR delivers proton beams to NU (HD), 2.48s (6.00s†) is used, respectively. Since the slow cycle determines the overall time behaviour of accelerators, it is often called "machine cycle". Two different cycles are co-exist in J-PARC.

The J-PARC Timing System started operation since 2006. The purpose is to provide a trigger to each of the accelerator components with a specified delay. In general, only one trigger is necessary within the 40ms (2.48s/6.00s) time slot for the rapid-cycle (slow-cycle) machines, respectively.

PRESENT TIMING SYSTEM

History and Overview

The early studies for the J-PARC controls were carried out in 2003 [4], in which the design for the timing system was one of the issues. In 2003, the VME-bus modules for timing control, both the send module and the receiver module were developed in collaboration with domestic companies [5]. Mass productions of VME modules and related NIM modules were in 2005-2007. The first beam to the LI (RCS, MR) was in 2006 (2007, 2008), respectively. Summarized history is shown in Figure 1.

#norihiko.kamikubota@kek.jp
† 5.52s after RUN64, October 2015

The control system for the J-PARC, including the timing system, is based on EPICS toolkit [6]. Both low-level support (device driver and EPICS databases) and high-level applications for the timing system were developed by ourselves. In order to manage table-formatted registers of VME modules, the waveform record-type of EPICS is used. Java and Python have been preferred for high-level GUI applications. .

In 2015, we have 118 (43, 45) receiver modules, which correspond to ~540 (~220, ~300) end-point signals for LI (RCS, MR), respectively. In addition, a few receiver modules exist for each of three experimental facilities. Only one send module exists for the whole accelerator complex.

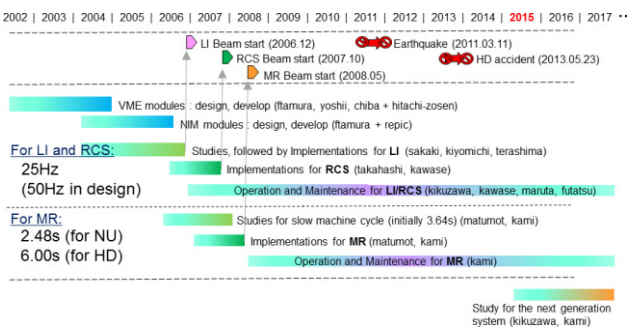


Figure 1: History of the J-PARC timing system.

Base-signals and Their Distribution

We have three "base-signals": a) 12MHz master clock (CLK), b) 25Hz trigger (Trig), and c) type-code (Type). The CLK is provided by a commercial high-stability function generator. The Trig is generated from the master clock, and used as the start signal of each rapid-cycle. The Type, a 32-bit number, is sent at the Trig rate after serialized. A receiver module uses it to generate delayed trigger signals during the next rapid-cycle.

We have fiber-optic cable networks throughout our facility buildings. Three base-signals are generated in the Central Control Building (CCB), then distributed to all the facility buildings using the fiber network (Figure 2).

A schematic view of base-signal distribution is shown in Figure 3. Several E/O, O/E and fan-out modules are used. All of the receiver modules in facility buildings receive the same base-signals.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

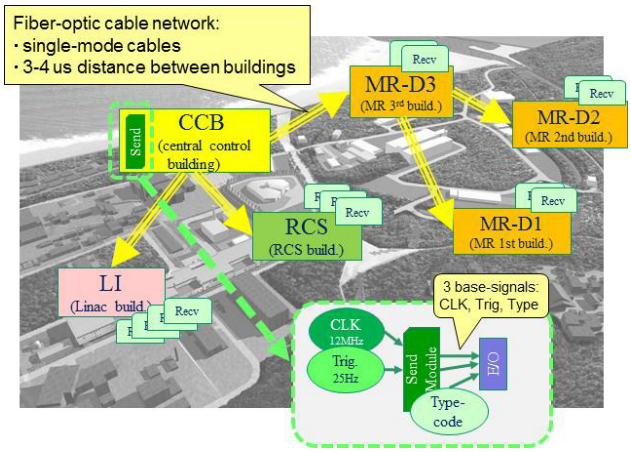


Figure 2: Fiber-optic cable network and the base-signals.

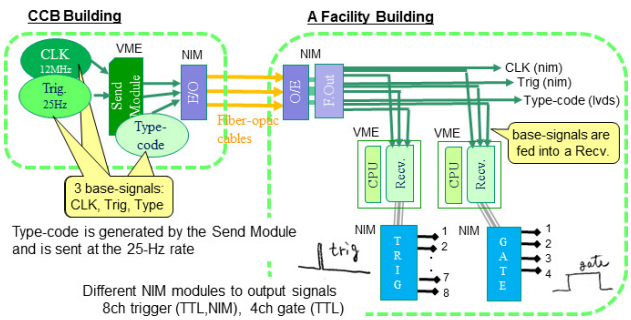


Figure 3: Schematic view of the base-signal distribution.

Send Module

The core of the J-PARC timing system is the send module. It has a type-memory, which consists of 64 type-sequences. At a run-time, one type-sequence is selected and used. The send module sends a type-code in the selected type-sequence one-by-one at the 25-Hz rate. The length of the type-sequence corresponds to the machine cycle; for example 62 (150) type-codes turn to the 2.48s (6.00s). The maximum size of a type-sequence is 1024.

A type-code is a 32-bit number, divided into four subsections: M1 (7bit for spare), M2 (8bit for LI), M3 (8bit for RCS) and M4 (8bit for MR). Each receiver module is set to use one of the subsections, thus only 8-bit is used at a run-time. The MSB (the 31st bit) is used to notify the end of the type-sequence.

Figure 4 is a GUI screen for the send module control in edit mode. Type-sequences and type-codes used in our operations are shown. In Figure 4. The red area is the MR injection period. Values of M2 (for LI) and M3 (for RCS) are different. LI and RCS behave differently during the MR injection according to the received type-codes.

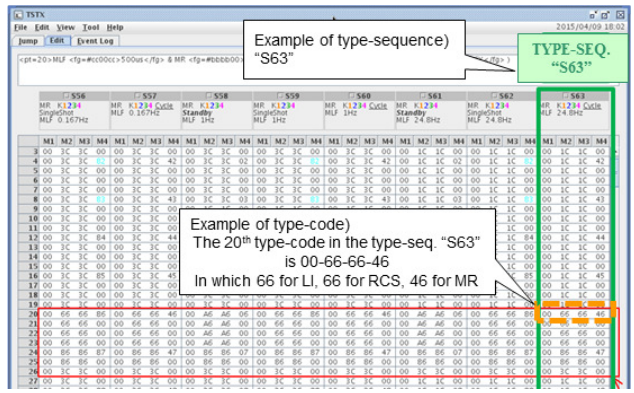


Figure 4: Java-based GUI for a send module control.

Receiver Module

A receiver module is capable to control eight independent delays, with the time resolution of 10.41ns. NIM modules are used to generate pulsed or gated signals (Figure 3). A receiver module has a LUT (Look-up-Table), which contains 256 x 8 delay-words. One delay-word determines the behaviour of one of eight channels, according to the received type-code.

A delay-word contains 24-bit delay count and control bits. After receiving a 25-Hz trigger, an internal counter is reset and starts up counting at the 96 MHz rate. When the counter reached at the delay count in the delay-word, an output trigger is generated. The MSB (the 31st bit) is used to disable output. The internal 96-MHz clock is produced from the 12-MHz master clock by a PLL.

A special control bit, the 30th, is used to continue counting even when the next 25-Hz trigger arrived. This feature is used to generate a large delayed signal, which exceeds the rapid-cycle (40ms). The maximum possible delay is about 170ms.

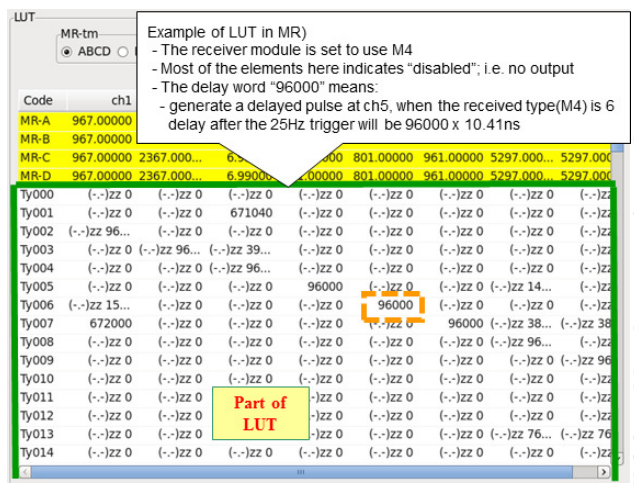


Figure 5: Python-based GUI for a receiver module control.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Figure 5 shows a part of LUT of a receiver module in MR. Most of elements in the LUT indicate "disabled", using the MSB. The highlighted number "96000" means: 1) the ch5 will generate a delayed output when the received type-code (M4 subsection) is 6, 2) amount of delay will be 96000 x 10.41ns after the 25-Hz trigger.

OPERATION AND EXPERIENCES

Beam-slot Sequence in Single Machine Cycle

Each of the 25-Hz time slots in single machine cycle has a predefined beam destination, which we call "beam slot". In 2015, typical operation modes for beam delivery to experimental facilities are: (a) MLF and NU with the machine cycle 2.48s, and (b) MLF and HD with the machine cycle 6.00s. Assignments of beam slots are shown in Figure 6.

In both operation modes, 4 beam slots, from 20th to 23rd, are assigned for MR injection, followed by 2 empty slots. Empty slots are needed to avoid miss-steering caused by the residual magnetic field of the switching magnet. New small magnet to compensate the residual field is in test [7].

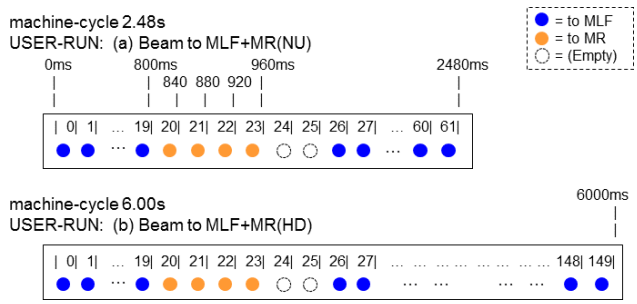


Figure 6: Typical beam-slot sequences.

Transitions of Beam-slot Sequences

In real machine operation, many beam-slot sequences are needed, and transitions between sequences must be considered. In the J-PARC timing system, switching a type-sequence in the send module to another results in a sequence transition. In Figure 7, upper sequences are for beam deliveries to experimental facilities, and lower for single-shot machine studies. Variety of transitions are possible using 64 type-sequences in the send module. Independent beam start and stop for MLF and MR, which is inevitable for our operation, is realized by this feature of the send module.

In daily operation, an accelerator operator changes beam-slot sequences manually. However, when a MPS event (machine fault) occurs, the sequence is changed automatically to the stand-by mode.

Change of the MR operation mode between NU and HD is once per a few months. This is carried out by a change of the machine cycle, left to/from right as shown in Figure 7. The change procedure is done manually also by an operator.

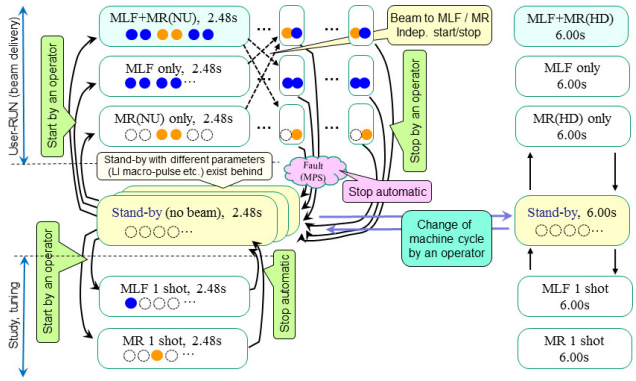


Figure 7: Transitions between beam-slot sequences.

Daily Modulation between Buildings

The length of the fiber-optic cables between buildings is about 1 km (Figure 1). One-way signal transfer takes 3-4 us. Daily time modulation, roughly 1ns, was observed between the CCB and the MR-D3 buildings. This modulation is caused by environmental temperature changes. The amount of this modulation is considered small enough and permissible for our accelerators.

Synchronization Timing

There are some exceptional trigger signals which are called "synchronization timing" [5]. For example, the extraction kicker of the RCS must be synchronized with the circulating beams, however, the standard trigger resolution (~10ns) is insufficient. The trigger for the kicker is generated by the RCS RF system. Addition to it, the MR injection kicker is triggered by the same signal with an appropriate delay. This delay is generated by a dedicated VME board, with the resolution of 2ns.

FUTURE PLANS

Evaluation of the Present System and Future

Since 2006, more than 200 VME modules have worked very well without faults, except few pieces. However, we have some problems:

- During distributing base-signals by metal cables, they suffer external noise influence from pulsed power-supplies. In an extreme case, a few meter cable suffered, especially the type-code in LVDS signal format. Ferrite cores is effective to reduce common-mode noises, but more essential solution is preferable.
- Optic devices used in E/O and O/E NIM modules, Finisar v23826 made in 2006-2010, are already discontinued. Re-producing the same modules is impossible.
- No good proposal for a small extension. When only one delay is necessary for a new component, set of a VME system and NIM modules are necessary. It requires too much space and cost.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

In order to overcome above problems, we start discussion on the future possible directions. Two ideas are shown in Figure 8.

The first idea is to introduce a FPGA board with a SFP. SFP seems a safe optic device for future long availability. In addition, merging/dividing three base-signals into/from single fiber-optic cable will be possible, using a FPGA technology.

Recently, the MRF timing products have been used in many accelerators [8]. Thus, the second idea is to develop a protocol converter from the J-PARC timing to the MRF. The MRF provides small-factor platform, such as cPCI or uTCA form-factor, which would solve our space problem. In addition, direct optic-link to a front-end module (EVR) would result in an effective measure against external noises.

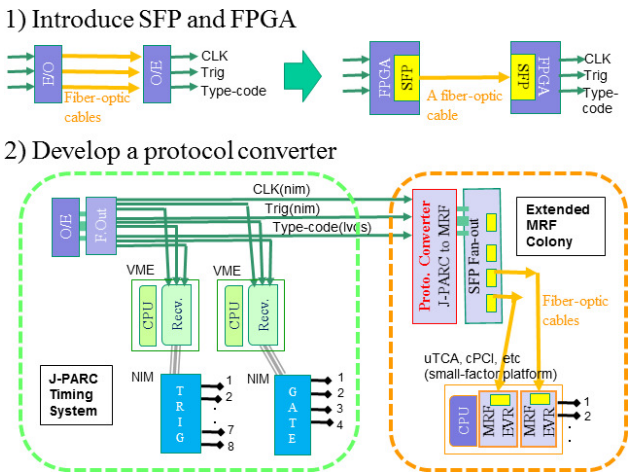


Figure 8: Two ideas for the future.

50Hz Operation of the Linac for the ADS

The ADS (Accelerator-Driven Transmutation) is an experimental facility in J-PARC. It is not constructed yet. Recently, a construction plan was approved. Additional 25 Hz beams will be needed for the ADS around 2018-2019.

In fact, the 50 Hz operation of the Linac is in the original design. The timing modules were also designed to work at the 50-Hz rate. Thus, beam-slots for the ADS can be assigned as in Figure 9.

Transitions between beam-slot sequences will be more complicated with the ADS. A plan to develop an enhanced send module with 1024 type-sequences (currently 64) is discussed.

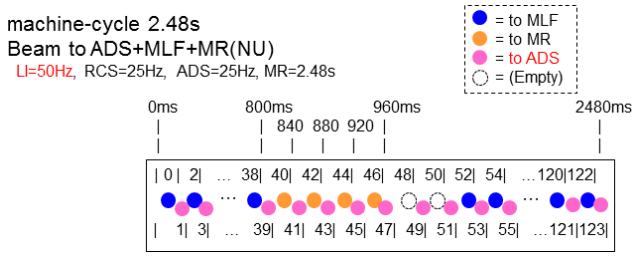


Figure 9: Plan of a beam-slot sequence with the ADS.

CONCLUSION

J-PARC Timing System hardware was developed roughly 10 years before, in collaboration with domestic companies. Software was developed by ourselves. Since 2006, the system has been used successfully in daily accelerator operations, with good enough reliability.

Recently, we have started discussions on possible migration and extension of the system for the next decade.

ACKNOWLEDGMENT

The initial development related to the send module was carried out by Y. Itoh, H. Yoshikawa, H. Sakaki and H. Takahashi. The scheme of the MR timing was designed and implemented by T. Matsumoto. We appreciate them, since these contributions were indispensable in the early development era of the J-PARC timing system. Thanks are also to Y. Sawabe, M. Kawase, K. Futatsukawa and the accelerator operators, for their continuous efforts to maintain and update the timing system.

REFERENCES

[1] J-PARC website: http://j-parc.jp/index-e.html
[2] ,S. Nagamiya, “Introduction to J-PARC”, Prog. Theor. Exp. Phys. (2012)02B001.
[3] T. Koseki and K. Hasegawa, “Present Status of J-PARC - After the Shutdown due to the Radioactive Material Leak Accident -”, IPAC’14, Dresden, Germany, THPME061, pp.3374-3375; www.JACoW.org
[4] J. Chiba et al., “Present Status of the J-PARC Control System”, ICALEPCS 2003, Gyeongju, Korea, p.1-5
[5] F. Tamura et al., “J-PARC Timing System”, ICALEPCS 2003, Gyeongju, Korea, p.247-249
[6] N. Kamikubota, et al., “J-PARC Control toward Future Reliable Operation”, ICALEPCS 2011, Grenoble, France, Oct. 2011, MOPMS026, pp. 378-381; www.JACoW.org
[7] J. Takano et al., “Residual Field Correction of Pulse Bending Magnet", JPS Conference Proceedings Vol. 8, 012023, Tsukuba, Japan (2015).
[8] MRF Website: http://www.mrf.fi

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REAL-TIME PERFORMANCE IMPROVEMENTS AND CONSIDERATION OF PARALLEL PROCESSING FOR BEAM SYNCHRONOUS ACQUISITION (BSA)*

K. H. Kim#, S. Allison, T. Straumann, E. Williams
SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA

Abstract

Beam Synchronous Acquisition (BSA) provides a common infrastructure for aligning data to each individual beam pulse, as required by the Linac Coherent Light Source (LCLS)[1]. BSA allows 20 independent acquisitions simultaneously for the entire LCLS facility and is used extensively for beam physics, machine diagnostics and operation. BSA is designed as a part of LCLS timing system [2,3] and is currently an EPICS record based implementation, allowing timing receiver EPICS applications to easily add BSA functionality to their own record processing. However the lack of real-time performance of EPICS [4] record processing and the increasing number of BSA devices has brought real-time performance issues. The major reason for the performance problem is due to the lack of separation between time-critical BSA upstream processing and non-critical downstream processing. We are improving BSA with thread level programming, breaking the global lock in each BSA device, adding a queue between upstream and downstream processing, and moving out the non-critical downstream to a lower priority worker thread. We are also investigating the use of multiple worker threads for parallel processing in Symmetric Multi-Processor (SMP) system.

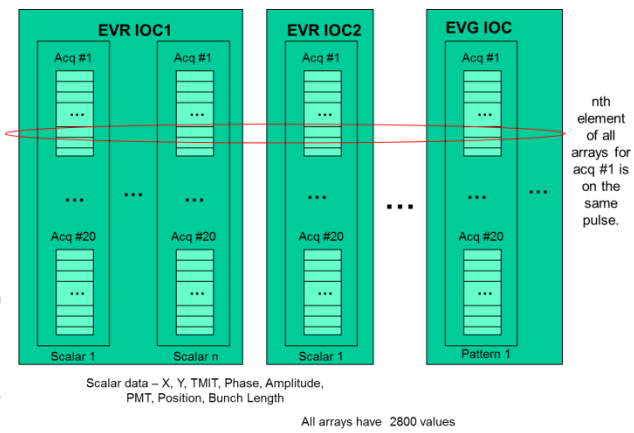


Figure 1: Data Acquisition across IOCs

BEAM SYNCHRONOUS ACQUISITION

BSA has been designed as a part of event system in LCLS-I to acquire all of beam dependent scalars across multiple IOCs in the entire accelerator facility on the same beam pulse, allowing correlation analysis using the pulse by pulsed aligned acquisition data (Figure 1). BSA acquires up to 2,800 values per scalar in one acquisition request; each value of the 2,800 can be an average of up to 1,000 values. It also provides RMS values and other statistics. The BSA can process 20 different acquisitions simultaneously [5].

BSA is implemented in three parts. A user request for an acquisition is done by EPICS CA client. Data gathering is processed on the Event Generator (EVG) and Event Receiver (EVR) IOCs. When gathering is finished, access of prepared data waiting on IOCs is done by CA clients, with checks for a good acquisition.

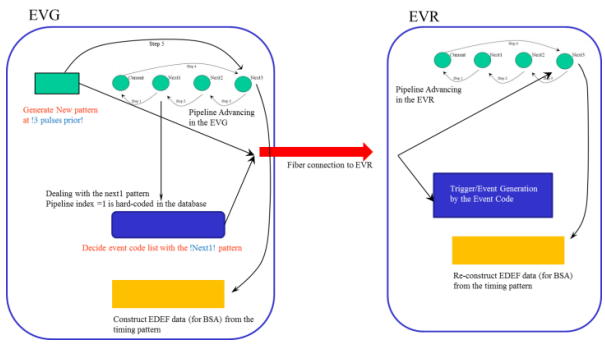


Figure 2: 360Hz tasks in EVG and EVR

360Hz Task in EVG

Acquisition setup and start requests done on the EVG IOC. The EVG IOC performs 360Hz checking and user notification when a BSA is finished. A 360Hz event task wakes up on an interrupt from a clock synched AC powerline zero-crossing which also provides timeslot for the event system. The event task is the heart of EVG IOC, generating a timing pattern for 3 pulses ahead and schedules timing events for the next pulse. The event task also checks for a match between the new pulse's timing pattern, beam code, and each active acquisition for BSA. It keeps a count of the number of measurements and the number of values in the current average per acquisition. One part of the timing pattern represents which acquisitions are matched. The timing pattern is broadcasted to EVRs and contains pulse id, timestamp, and additional BSA information. The pulse information is pipelined, thus the timing pattern is for 3 pulses ahead (Figure 2).

*Work supported by the the U.S. Department of Energy, Office of Science under Contract DE-AC02-76SF00515 for LCLS I and LCLS II.
#khkim@slac.stanford.edu

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

360Hz Task in EVR

The Event Definition (EDEF) bits included in 360Hz data are sent by EVG to EVR. EVR IOC caches the data on the EVR data interrupt. The EVR IOC 360Hz event task is activated on the next fiducial interrupt (actually, event code 1), copying the data to the end of the timing pipeline, and then shifting the pipeline. Finally, the 360Hz task updates EDEF table which is used by BSA processing done later in the same pulse (Figure 3).

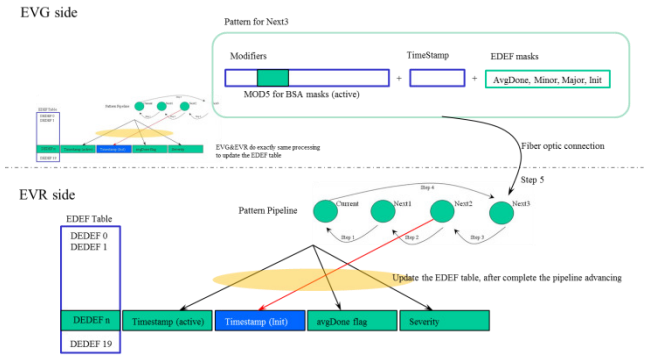


Figure 3: 360Hz tasks and EDEF table update

BSA Processing

BSA processing is driven by data source processing variable (PV). The data source PV provides new data with timestamp and triggers BSA processing. The data receptor PV receives new data and timestamp, and then checks the timestamp against the EDEF tables. If there is a timestamp match, the data is included in the BSA buffer. The data receptor PV calculates RMS and average, if required, and then requests BSA record processing to update the BSA buffer PV. Each BSA record then delivers data to a compress record.

IMPROVEMENT REQUIRED

LCLS-I has around 1,150 BSA PVs and more than 980 BSA PVs work successfully. However, around 160 PVs sometimes fail. The reason of BSA failure varies, and is sometimes due to misconfigured data source PVs, and rates. Some data sources PVs deliver the data too late without enough time to process BSA. The misconfiguration and lazy data source PVs are fixed, as they are found, to resolve the BSA problems.

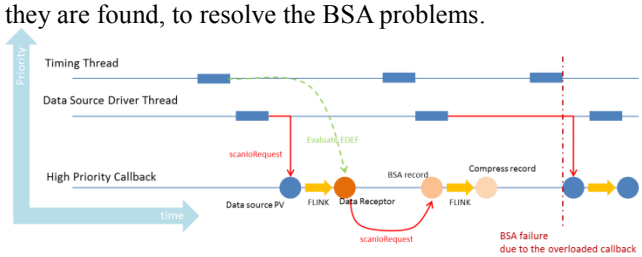


Figure 4: BSA fail scenario

We also found, under test conditions, that some BSA PVs fail even when there is an enough time budget. Sometimes the data source PVs are not completed within the expected time, and get a wrong timestamp. Thus, we discovered an unexpected delay on the data source PV and BSA processing, due to EPICS record processing. The data source and BSA processing are processed by high priority callback thread in the EPICS IOC. The callback thread also processes other records and results in unexpected delay (Figure 4).

BSA IMPROVEMENT

EVR side BSA processing has been implemented in two parts – data reception and BSA record processing – in the current implementation. The data receptor PV is triggered by the data source PV through a forward link when new data is ready. The data receptor matches timestamp to decide if the data goes into the BSA buffer, and performs RMS and average calculations, and then requests BSA record processing to update BSA data buffer implemented in a compress record. The timestamp matching in the data receptor PV is the only time-critical part of BSA processing. It cannot be delayed though the rest of the processing can be delayed if we do not lose information.

If we separate out the time-critical part, and process it immediately after data is ready, the rest is queued up for a lower priority thread. The lower priority thread takes care of the non-time-critical parts and finally updates BSA data buffer whenever it can get CPU time (Figure 5).

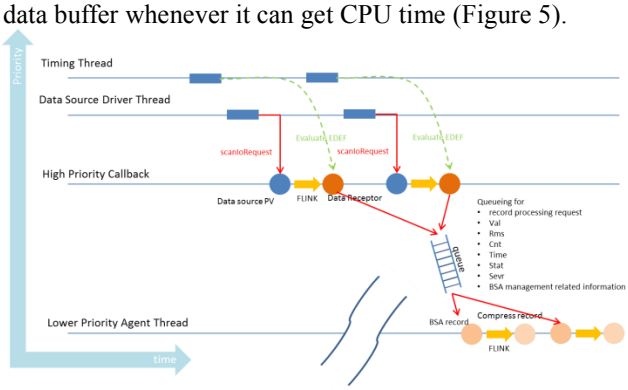


Figure 5: Processing timeline for BSA improvement

We provide a new API for this improvement. The new API is called by the device driver of the data source. In this case, the time-critical part is run in the driver context. We keep the EPICS record interface – data receptor PV – for backward compatibility. In this case, the time-critical part is run by the high priority callback thread. But, the non-time-critical part is run by the lower priority thread.

The new EPICS base R3.15 provides parallel callbacks. We can parallelize the high priority callback with the new feature. The improvement also allows multiple lower priority agent threads for the non-time-critical parts. The multiple threads share a single queue to receive BSA information from the upstream time-critical part (Figure 6). It improves performance in Symmetric Multi-Processor (SMP) system. Recently, we moved to the linuxRT [6] operating system from RTEMS [7] and the new platform provides the multi-core system. The parallel

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

callbacks and multiple lower priority agent threads bring a performance improvement for the multi-core system.

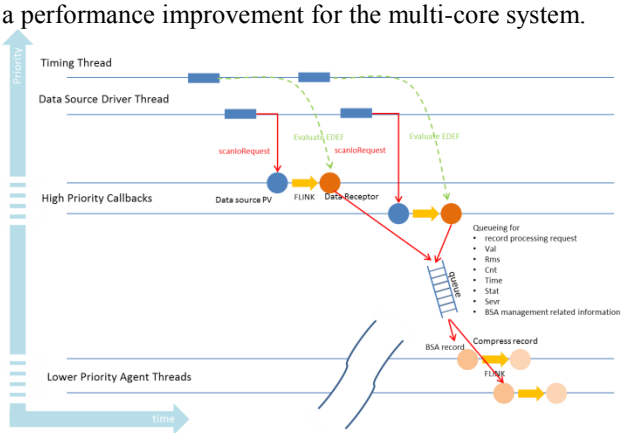


Figure 6: Consideration for parallel callbacks for multi-core system

PROTOTYPING AND TEST RESTULTS

We made a quick prototype for the BSA improvement and tested its performance.

Test Results for a Single Core System

We tested the prototype with a single core system (MVME6100/RTEMS/epics-3.15.1). For a realistic test, we forced 100% CPU load with an infinite loop in the lowest priority thread which has no effect on BSA processing and any other mission critical tasks. The system could handle up to 64 BSA PVs at 120Hz rates for 8 active EDEFs. The BSA capacity almost doubled. The system only allows 32 BSA PVs with the same condition. We also found that the time-critical part spends around 5 micro-seconds, and non-time-critical part spends slightly longer than 10 micro-seconds. Thus, the queueing mechanism and lower priority agent thread improves the BSA capacity for a single core system.

Test Results for a Multi-Core System

We also tested the prototype with a multi-core system (x86, 32 cores/linuxRT/epics-3.15.1). We also forced 100% CPU load in the lowest priority infinite loop thread. We tried parallel callbacks for upstream processing and multiple lower priority agent threads for downstream processing.

We assumed a dramatic performance improvements but it only doubled compares to the original code. The number of callback threads and the number of agent threads did not affect the result. We varied these numbers from16 to 32.

Finally, we discovered a global locking and lockset issues in EPICS record processing, and it affects the performance of record processing for parallel callbacks. We asked the EPICS core development team to fix this issue and they provided a snapshot version for testing. We contributed to the debugging of the snapshot version and continue our testing with it.

CONCLUSION

We have developed a quick prototype for BSA improvement. We separated out time-critical and non-time critical parts and queued the non-time-critical processing into a lower priority thread. We verified the prototype shows improvement for a single core system. We also tested the prototype for a multi-core system with parallel callbacks and multiple agent threads. We expected a dramatic performance improvement for the multi-core system but it only shows a similar improvement as the single core system due to a global locking issue in the EPICS record processing. We need more multi-core testing with an improved version of EPICS.

We will implement an API for production which can be called by the data source driver to avoid EPICS record processing overhead. It will improve performance.

REFERENCES

[1] J. Arthur, *et.al.*, "Linac Coherent Light Source (LCLS) conceptual design report," SLAC-R593, SLAC (2002)

[2] P. Krejcik, *et.al.*, "Timing and Synchronization at the LCLS," DIPAC 2007, Venice, Italy, May 2007, SLAC-PUB-12593

[3] J. Dusatko, et.al., "The LCLS Timing Event System," BIW 2010, Santa Fe, New Mexico, USA, May 2010

[4] "Experimental Physics and Industrial Control System," http://www.aps.anl.gov/epics

[5] M. Zelazny, *et.al.,* "Orbit Display's use of the Physics Application Framework for LCLS," SLAC-PUB-13788 SLAC (2009)

[6] "Real-Time Preemption Patch-Set," http://elinux.org/images/4/4e/Real-Time-Preemption-Patchset.pdf

[7] "RTEMS Real Time Operating System (RTOS)," https://www.rtems.org

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

APPLICATION USING TIMING SYSTEM OF RAON ACCELERATOR

S. Lee*, CW. Son†, HJ. Jang‡, IBS, Daejeon, South Korea

Abstract

RAON is a particle accelerator to research the interaction between the nucleus forming a rare isotope as Korean heavy-ion accelerator. RAON accelerator consists of a number of facilities and equipments as a large-scaled experimental device operating under the distributed environment. For synchronization control between these experimental devices, timing system of the RAON uses the VME-based EVG/EVR system. This paper is intended to test high-speed device control with timing event signal. To test the high-speed performance of the control logic with the minimized event signal delay, we are planing to establish the step motor controller testbed applying the FPGA chip. The testbed controller will be configured with Zynq 7000 series of Xilinx FPGA chip. Zynq as SoC (System on Chip) is divided into PS (Processing System) and PL (Programmable Logic). PS with the dual-core ARM cpu is performing the high-level control logic at run-time on linux operating system. PL with the low-level FPGA I/O signal interfaces with the step motor controller directly with the event signal received from timing system.

This paper describes the content and performance evaluation obtaining from the step motor control through the various synchronized event signal received from the timing system.

INTRODUCTION

The RAON [1] is a new heavy ion accelerator under construction in South Korea, which is to produce a variety of stable ion and rare isotope beams to support various researches for the basic science and applied research applications. To produce the isotopes to fulfill the requirements we have planed the several modes of operation scheme which require fine-tuned synchronous controls, asynchronous controls, or both among the accelerator complexes. RAON, which is a large experimental machine, consists of many experimental devices and additional facilities. For synchronized control under the distributed environment, timing system of the RAON uses the VME-based Event Generator (EVG)/Event Receiver (EVR) [2] system.

TIMING SYSTEM

The timing system uses the EVG/EVR system of the Micro-Research Finland Oy company in VME form factor. The timing system consists of an EVG which converts timing events to optical signals, Fan-Out unit that repeats the signals, an array of EVRs and VME-controller. The EVRs decode the optical signal and produce hardware signal or software interrupt for the timing event. It uses the MVME6100 or 3100 of Emerson company as VME-controller. The controller also uses real-time operating system, VxWorks 6.9 for a fast response. Finally, EPICS IOC (mrfioc2) on VxWorks controls the FPGA-based EVG/EVR boards.

The characteristics of MRF timing system are :

- Event driven system, 256 event codes
- Event generation using external RF reference clock
- 50 ~ 125MHz event clock rate
- Events generated
  - From external HW inputs
  - Two sequencers (up to 2048 events/sequencer)
  - Multi counters
- Cascaded Event Generators
- Different Clock Synchronization

Hardware Configuration

Hardware configuration of the timing system consist of:

- XLI GPS Time System
- Rubidium Frequency Standard Clock Source (FS725)
- Event Trigger System (EVG/EVR,Fanout Repeater/Concentrator)
- MVME 6100/MVME 3100 controller board
- VME Crate (Wiener)
- SMA 100a RF Signal Generator

Figure 1 shows hardware configuration to test timing system.

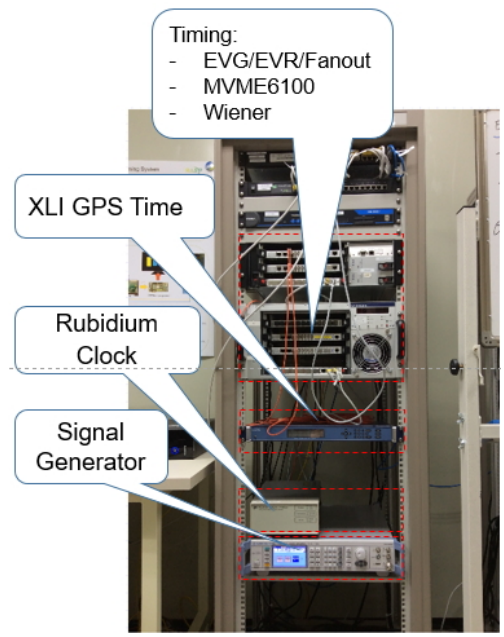


Figure 1: Timing hardware configuration.

* silee7103@ibs.re.kr
† scwook@ibs.re.kr
‡ lkcom@ibs.re.kr

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Referring to the configuration of Fig. 1, GPS receiver synchronizes with rubidium clock and EVG to 1PPS. Signal generator generates RAON reference clock (81.25 MHz) to EVG. The two input signals of EVG are synchronized by locking the phase. The generated signals from EVG according to event codes registered by mrfioc2 are distributed to EVRs through Fan-out repeater.

Software Configuration

Software modules to operate the timing system consist of:

- Workbench 3.3, vxWorks IDE
- VxWorks 6.9 Realtime Operating System or RTEMS for MVME3100
- EPICS Framework (R3.14.12.5)
- MRFIOC2/SRSIOC
- Network Time Protocol (NTP)

FAST STEPPER MOTOR TESTBED USING TIMING SIGNAL

Configuration of Fig. 2 shows the overall of a fast stepper motor testbed operating by the input of the timing signal. As described already before, the EVG of the timing system receives two input signals, the external event source which is locked to GPS and RF reference clock. The synchronized events of EVG are distributed through optical fiber links to EVRs. The EVRs received some events from EVG decode those events and trigger TTL level signals to ZC706 evaluation board. The PL of zynq which is received from external TTL signal gives the signal to motor controller according to the motor's parameters. The setting of the parameter values for the motor control is performed on the EPICS interface of the zynq PS. PS is equipped with the EPICS software module on the cross-compiled linux OS of the ARM processor. The interface between PS and PL uses the AXI interface of the ARM Advanced Microcontroller Bus Architecture (AMBA). The AMBA [3] is an open-standard, on-chip interconnect specification for the connection and management of functional blocks in system-on-a-chip (SoC) designs. Software module for AXI interface between both area should be also developed in the area of the linux device driver. The types of motor which can be supported by a low voltage drive board of the Analog Device [4] are brushless DC, PMSM, brushed DC or stepper motor.

System on Chip: FPGA - Zynq

Zynq [5, 6] as SoC is divided into Processing System (PS,dual-core ARM cpu) and Programmable Logic (PL, FPGA). Communication between PS and PL is through the AXI interface of ABMA.

ZC706 Evaluation Board of Xilinx

The ZC706 evaluation board for All Programmable SoC (AP SoC) provides a hardware environment for developing and evaluating designs targeting the Zynq®-7000 AP SoC. [7] The ZC706 board has the SMA port which is the programmable user clock for the TTL out signal of the EVR timing board. The reason which selected the board is to have the user clock port to receive input from the external trigger.

Controller for Motor Drive

Figure 3 shows the controller and low voltage drive board of the Analog Devices to drive the motor. Controller board communicates with ZC706 via FPGA Mezzanine Card (FMC) connector. FMC connector of the controller board is connected to XADC interface, digital I/O and sensors, two gigabit ethernet modules, power line and so on. XADC and digital I/O signals among those connections are delivered to the low voltage drive board via the drive board connector. The low voltage drive board received those signals generates Pulse Width Modulation (PWM) signal through the MOSFET gate driver and drives the stepper motor. The low voltage drive board is operated in 12~24V DC power from the external power source.

Linux on Zynq PS (ARM Processor)

To operate linux OS on the ARM processor of the zynq PS there consists of:

- ARM Cross Compile Tool Chain (arm-linux-gnueabihf)
- Linux Kernel Source (Linaro)
- Bootloader (BOOT.BIN)
- Board Support Package (Linux Device Tree)
- Root File System (Busybox)

Linaro [8] is a not-for-profit engineering organization that works on free and open-source software such as the Linux kernel, the GNU Compiler Collection (GCC), power management, graphics and multimedia interfaces for the ARM family of instruction sets and implementations thereof as well as for the Heterogeneous System Architecture. The linaro kernel source is the ubuntu-based kernel source including a lot of device driver such as zynq device tree. The bootloader of PS uses U-Boot [9] supporting for the zynq device of Xilinx. To boot the zynq device, it should be made up the boot.bin file created by Vivado [10] tool. The boot.bin [11] file consists of fsbl.elf (First Stage BootLoader), u-boot.elf (Second BootLoader), uImage (kernel zImage for uboot), zynq.bif (Boot Image Format file) and user.bit (bitstream file of FPGA). The linux kernel is using the root file system generated by the Busybox [12] tool.

Linux Device Driver

Linux device driver for zynq PS uses Industrial IO (IIO) module of the Analog Devices. Libiio [13] is a library that has been developed by Analog Devices for easy interface with IIO devices. Localbackend module of Fig. 2 goes into kernel mode from user mode through the system call and accesses the iio device driver via "struct file_operations". Basically the iio device driver is a character device type.

Software Interface

EPICS IOC is being implemented using the libiio library that abstracted the low-level details of the hardware, as shown in Fig. 2. EPICS is a set of open source software tools,

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

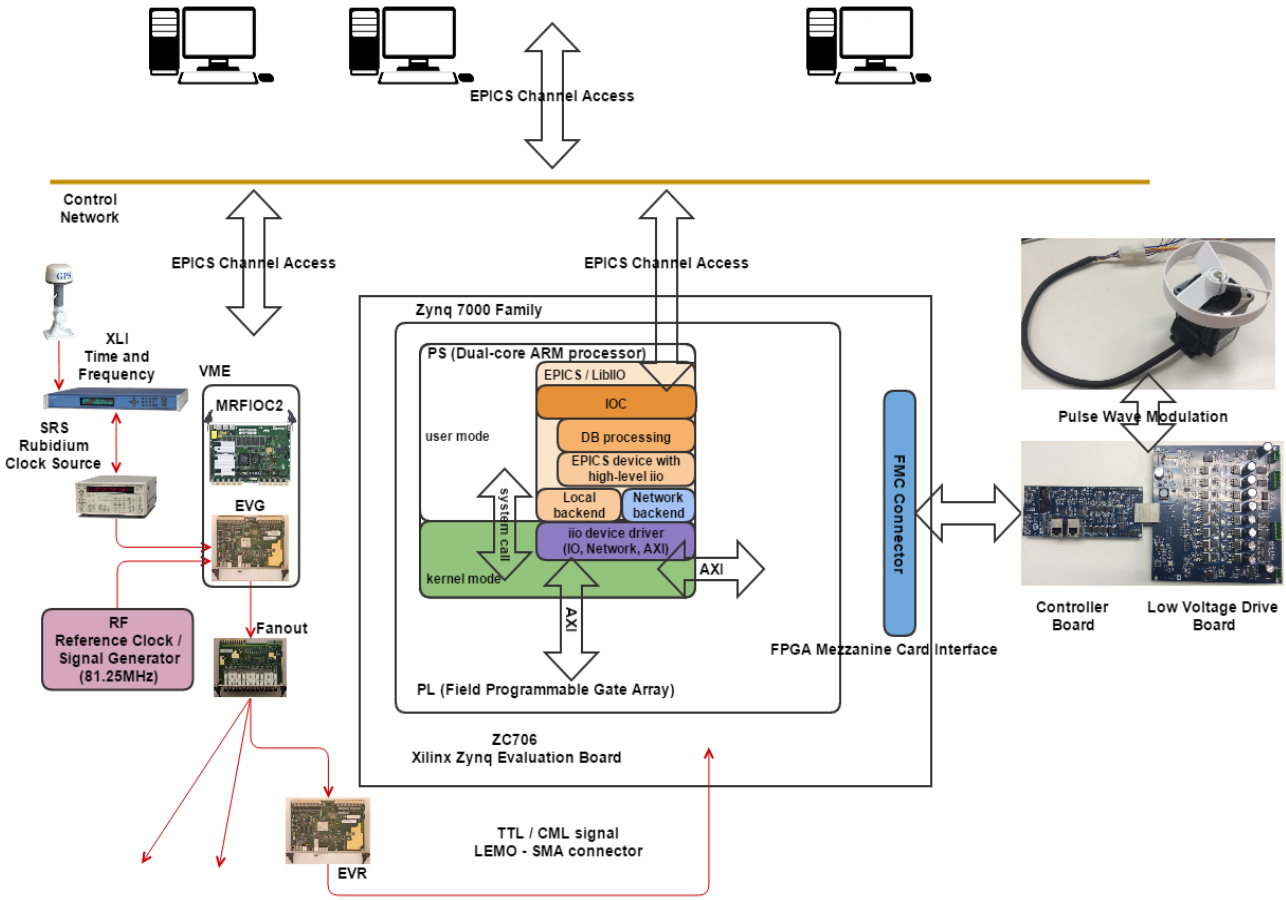


Figure 2: Fast motor testbed adopted zynq device using timing event signal.

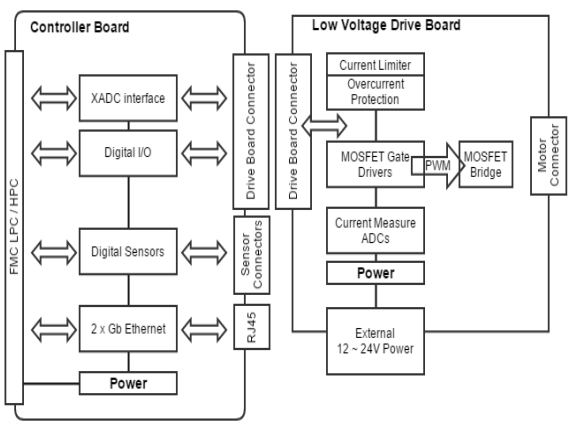


Figure 3: Controller and low voltage drive board.

libraries and applications developed collaboratively and used worldwide to create distributed soft real-time control systems for scientific instruments such as a particle accelerators, telescopes and other large scientific experimental facilities [14]. The device support routine of EPICS is implemented using the libiio library to read or write the parameter values for the motor control. The motor control parameters become the Process Variables (PVs) of EPICS and are carried out the EPICS database processing according to EPICS scan rate. Also, it is going to develop additional EPICS waveform PV to monitor the PWM signal of the low voltage drive board.

FPGA Interface reuses the Hardware Description Lanaguage (HDL) code which is provided by the Analog Devices. FPGA HDL code is develeped by VerilogHDL using Vivado of Xilinx. When the motor controller was purchased from Analog Devices, it operated on ZedBoard. It should be changed to operate the ZC706 board as well as ZedBoard. It is going to transplant the code of ZedBoard into ZC706 board. In addition, It is being implemented FPGA code to drive the motor receiving the input signals (EVR TTL signals) via user clock port of ZC706 board.

SUMMARY

Table 1, 2 summarize the configuration of hardware and software as shown in Fig. 2.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Table 1: Hardware Configuration for Stepper Motor Testbed

Hardware	Contents	Company
GPS	GPS antena	Symmetricom
	XLi time and frequency	
Clock Sync.	Rubidium frequency	Standford
	standard model FS725	Research
Ref. Clock	Signal Generator or RF	
	Reference (81.25 MHz)	
EVG	Event Generator	Micro
EVR	Event Receiver	Research
FanOut	Event Repeater	Finland Oy
MVME6100	VME Controller	
ZC706	Zynq 7000 family Eval. kit	Xilinx
	Processing System	
	(dual-core ARM processor)	
	Progrmmable Logic	
Controller	Motor controller board	Analog
Low Volt.	Low voltage drive board	Devices
Motor	Stepper or BLDC	

Table 2: Software Configuration for Stepper Motor Testbed

Software	Contents	Module
EPICS	Base R3.14.15.2	PS of Zynq
Libiio	Industrial I/O library	
	supplied Analog Devices	
Kernel	Linaro kernel source	
	including iio device driver	
Bootloader	U-Boot for zynq	
IOC	In-house using Libiio	
FPGA	Analog Devices and	PL of Zynq
	In-house code	
VxWorks	Real-time OS for VME	Timing
	including BSP	
MRFIOC2	EPICS IOC for timing	
SRSIOC	IOC for rubidium clock	
Workbench	Workbench3.3 for VxWorks	Development
Vivado	Vivado14.4 including SDK	Tool
	with node lock license	
Busybox	for rootfs system (free)	
Toolchain	for ARM cross-compile (GNU)	

CONCLUSION

Timing system can distribute the fine synchronized event signal at a high speed. The objective of the stepper motor control testbed is to know how to operate the timing system and how to apply it to the high speed controller. The overall implementation is still underway, however if the distributed control system using EPICS makes a connection with the high speed parallel processing of FPGA, it is possible to improve the performance and efficiency of the control system. Zynq SoC can be considered as an ideal device to implement this high-speed control system.

ACKNOWLEDGMENT

This work is supported by the Rare Isotope Science Project funded by Ministry of Science, ICT and Future Planning(**MSIP**) and National Research Foundation(**NRF**) of Korea(Project No. 2011-0032011).

REFERENCES

[1] Y. K. Kwon, *et. al*,"Status of Rare Isotope Science Project in Korea", Few-Body Syst 54, 961-966, (2013).

[2] Micro-Research Finland Oy website: http://www.mrf.fi

[3] Advanced Microcontroller Bus Architecture: http://en.wikipedia.org/wiki/Advanced_Microcontroller_Bus_Architecture

[4] Analog Devices website: http://www.analog.com

[5] Xilinx-FPGA website: http://www.xilinx.com

[6] Zynq-Book Document website: http://www.zynqbook.com

[7] ZC706 Evaluation Board Document: http://www.xilinx.com/support/documentation/boards_and_kits/zc706/ug954-zc706-eval-board-xc7z045-ap-soc.pdf

[8] Linaro Document website : https://en.wikipedia.org/wiki/Linaro

[9] U-Boot Document website : http://www.denx.de/wiki/U-Boot

[10] Xilinx Vivado Document website : http://www.xilinx.com/products/design-tools/vivado.html

[11] Boot Image Document website : http://www.wiki.xilinx.com/Prepare+boot+image

[12] Busybox Document website : http://www.busybox.net/

[13] Industrial I/O Document website : https://wiki.analog.com/resources/tools-software/linux-software/libiio

[14] EPICS website : http://www.aps.anl.gov/epics/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PROTOTYPE OF WHITE RABBIT NETWORK IN LHAASO

HongMing Li*, GuangHua Gong*, Tsinghua University, Beijing, China
Qiand Du, LBNL, USA

Abstract

Synchronization is a crucial concern in distributed measurement and control systems. White Rabbit provides sub-nanosecond accuracy and picoseconds precision for large distributed systems. In the Large High Altitude Air Shower Observatory project, to guarantee the angular resolution of reconstructed air shower event, a 500 ps overall synchronization precision must be achieved among thousands of detectors. A small prototype built at Yangbajin, Tibet, China has been working well for a whole year. A portable calibration node directly synced with the grandmaster switch and a simple detectors stack named Telescope are used to verify the overall synchronization precision of the whole prototype. The preliminary experiment results show that the long term synchronization of the White-Rabbit network is promising and 500 ps overall synchronization precision is achievable with node by node calibration and temperature correction.

INTRODUCTION

The Large High Altitude Air Shower Observatory (LHAASO) project is a dedicated instrument for searching the origin of galactic cosmic rays, which consists of 4 sub-detector arrays. The 1km^2 complex array (LHAASO-KM2A) includes 5635 scintillation electron detectors and 1221 muon detectors. 500 ps (rms) overall synchronization precision must be achieved among the spread thousands of detectors to guarantee the angular resolution of reconstructed air shower event. [1][2] A small prototype based on the White Rabbit (WR) network [3][4] is built at Yangbajin, Tibet, China on August, 2014. This paper talks about the deployment and synchronization monitor of the prototype.

DEPLOYMENT

Components and Topology

This prototype contains 4 WR switches and 50 WR customized nodes (48 electron detectors and 2 muon detectors). Rubidium clock constrained GPS receiver takes the clock and frequency source of the whole network. The four WR switches is used to build a four layer hierarchy WR network. The grandmaster switch (GMS) uses the frequency from GPS receiver and synchronizes its time (seconds counter) through NTP protocol from GPS receiver. Each WR node synchronizes its local clock against the GMS through fibres of one hundred meters and uses the synchronized clock to timestamp the events. Detail information of each component is listed below:

- 8040C, Rubidium Frequency Standard
- XL-GPS, produced by symmetricom
- WR switch, version V4.0.1
- Compact Universal Timing Endpoint (CUTE) [5], customized WR node, WRPC version V2.1. [6]

Locations

The prototype is deployed in the ARGO experiment hall whose altitude is around 4300m. The locations of WR switches and nodes are showed in Figure 1.

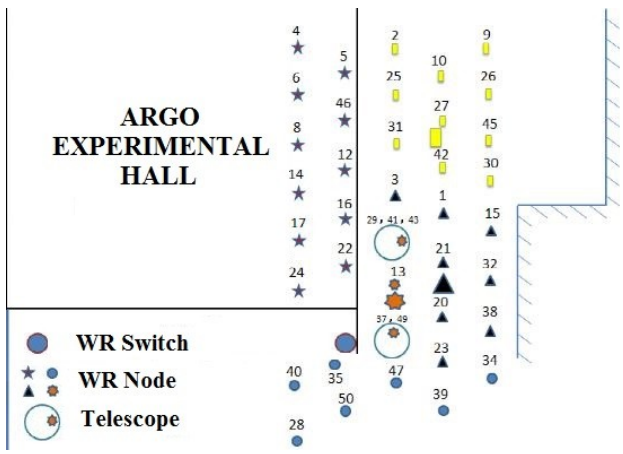


Figure 1: Locations of WR Switches and Nodes

Rubidium Frequency Standard, GPS, GMS and host computers are put in the ARGO Experiment Hall with small temperature variations, while the other three switches are put together in a waterproof steel box out of the Experiment Hall. A normal switch is used to connect the GPS network interface (acts as NTP server), host computer and all four WR switches for remote control.

The telescope is a pyramid of several WR nodes for monitoring the synchronization between the nodes. One node is on the top of another node to make sure they detect the same signal simultaneously.

Calibration Procedure

By following the calibration procedure given in the document [7], a node by node calibration is applied on all WR devices to conquer the differences caused by the components diversity.

As the bad control of the power noise in these boards, the synchronization precision (standard deviation value of the Pulse Per Second (PPS) skew) of each link is decreased to 100ps. Then the synchronization accuracy (absolute mean value of the PPS skew) less than 200 ps after calibration is accepted.

* The authors are with Key Laboratory of Particle and Radiation Imaging, Tsinghua unversity, Beijing, China.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Temperature Correction

Previous experiments have shown that the temperature variations have significant impacts on the synchronization performance of CUTE WR nodes. It can be reduced using the method discussed in [8] and a unified temperature coefficient is used for all nodes.

SYNCHRONIZATION MONITOR

The prototype has been working for about a year. Several methods are applied to measure and monitor the synchronization performance of the WR based network.

Portable Calibration Node

A portable calibration node (PCN) is a normal CUTE WR node put in a water sealed, vibration protected box. It is directly connected with and synced to the GMS through a shield optical power composite cable. So it can be used as a reference clock for other distributed WR switches and nodes to monitor the synchronization performance. Figure 2 shows the measured PPS skew between PCN and GMS while Figure 3 is that between PCN and other switches after the prototype is deployed. The latter measurement lasted for more than 20 hours when the temperature varied from 10 to 26 centigrades.

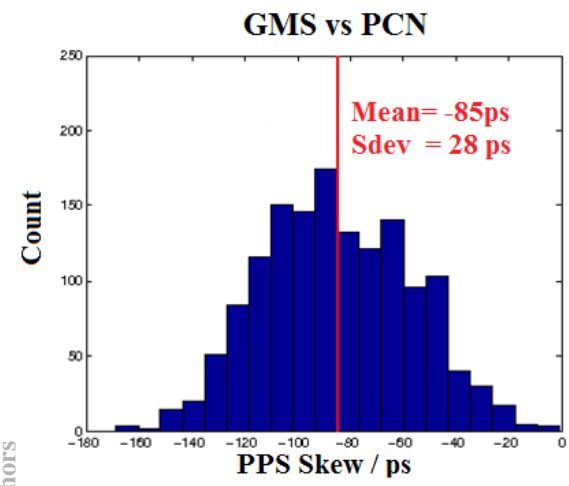


Figure 2: PPS skew between PCN and GMS

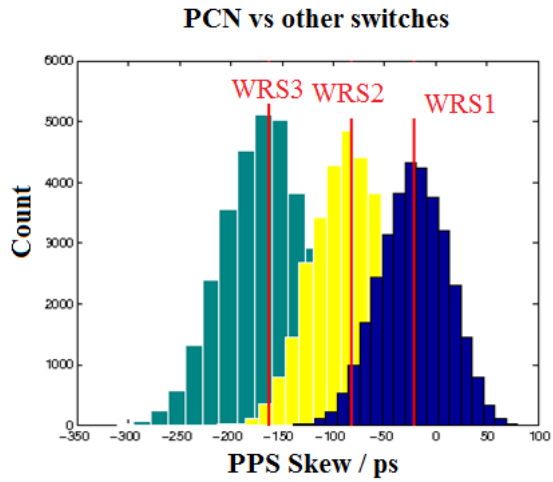


Figure 3: PPS skew between PCN and other switches

Table 1: PPS Skews Between PCN and CUTE WR Nodes

Node Index	Accuracy (mean) / ps	Precision (sdev) / ps
2	-138	100
5	-56	28
8	28	28
12	-27	150
16	5	28
17	-15	162
24	97	156
32	-40	34
39	-28	41
42	-157	135
44	-102	26
50	-45	34

Table 1 lists the measured PPS skews between the PCN and some CUTE WR nodes. Each measurement contains more than 300 samples. As the WR protocol is integrated with the detector electronics, the bad synchronization precision of some nodes is caused by the power noise of the main board which will be reduced in the future design.

Remote Status Monitor

In order to monitor the running status of the WR nodes, a simple network protocol is applied to get the values of timing related registers periodically. The WR link will rebuild after detecting a very large “clock offset” or “skew” value. It is done by remote restarting the WR switches or power off and on the WR nodes manually.

In the future, the etherbone protocol [9] will be integrated into the WRPC core to get a better remote

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

control, as well as supporting remote flash update and multi-boot.

Telescope Results

The telescope described before is used to monitor the synchronization performance between these nodes. The timestamps offset of each synchronizing detection between node 29 and 41 are recorded which can be used to represent the synchronization deviation between the nodes. Figure 4 and 5 show how the mean and standard deviation of the timestamps offset changes over time (Modified Julian Date). Each dot represents the statistics results of the timestamps offset recorded in a hour. Figure 6 shows that the timestamps offset has an almost linear relationship with the temperature difference between the two nodes. It changes about 2 nanoseconds when the temperature difference varies nearly 35 centigrades during the experiment.

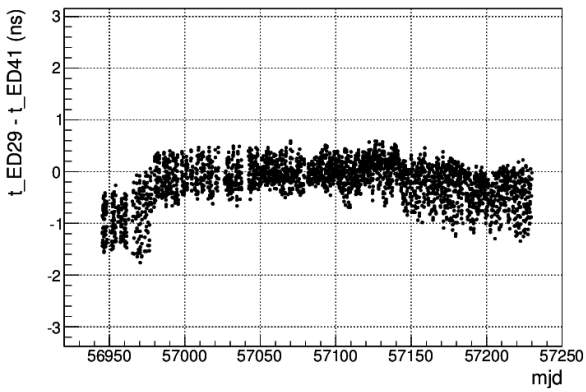


Figure 4: mean of the timestamps offset changes over time

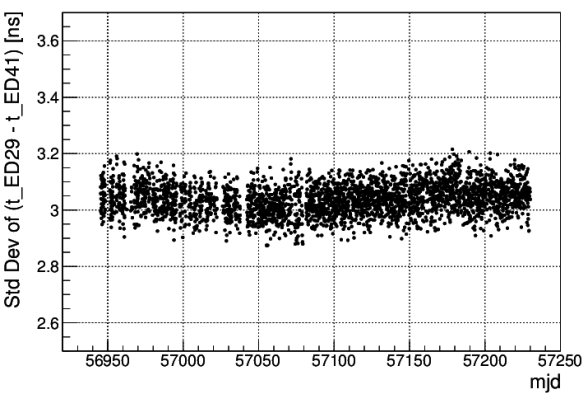


Figure 5: standard deviation of the timestamps offset changes over time

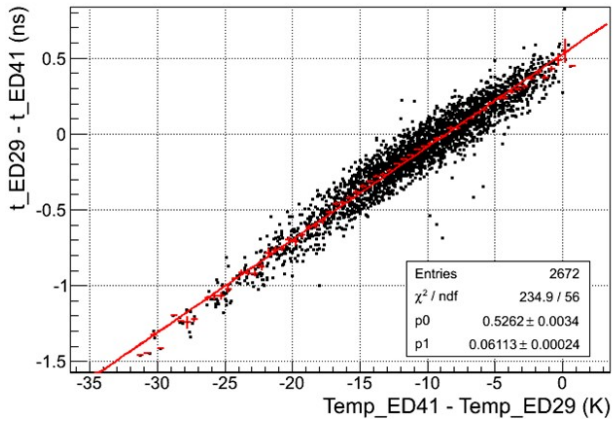


Figure 6: mean of the timestamps offset changes over temperature difference

The synchronization performance contains the contributions of the detectors and the WR network. The large standard deviation of the timestamps offset (~3ns) is mainly caused by the detectors. And as the temperature effect on the WR network has been reduced with dynamic correction, the temperature coefficient is also mainly determined by the detectors.

The results show that the long-term synchronization performance of the WR network is promising and temperature correction is also needed for the whole detector node when the ambient temperature changes a lot.

CONCLUSION

The prototype arrays of LHAASO is deployed at Yangbajin, Tibet, China and has been working well for nearly a year. Several bugs are found and reported to White Rabbit development team during the operation time and some of them have been resolved. It shows that the long term synchronization of the White-Rabbit network is promising and 500 ps overall synchronization precision is achievable with individual calibration and temperature correction. Power noise reduction, etherbone support, remote update and some other improvements will be added into the next update.

ACKNOWLEDGEMENT

This work is supported by the National Science Foundation of China (No. 11275111). The authors would like to thank Huihai He, Jia Liu, Hongkui Lv and the White Rabbit team at CERN for their help.

REFERENCES

[1] G. Gong and et al, "Sub-nanosecond timing system design and development for LHAASO project," in Proceedings of ICALEPCS, Grenoble, France, (2011).

[2] Q. Du and G. Gong, "A packet-based precise timing and synchronous DAQ network for the LHAASO project," Nuclear Instruments and Methods in, Physics Research A, vol. 732, pp. 488–492, (2013).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

[3] J. Serrano, P. Alvarez, M. Cattin, E. G. Cota, P. M. J. H. Lewis, T. Włostowski et al., "The White Rabbit Project," in Proceedings of ICALEPCS, TUC004, Kobe, Japan, (2009).

[4] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, G. Gaderer, "White Rabbit: Sub-Nanosecond Timing Distribution over Ethernet", IEEE Precision Clock Synchronization for Measurement, Control and Communication 2009, pp1-5, Brescia, October 2009.

[5] W. Pan, Q. Du, and G. Gong, "Development of a White Rabbit Interface for Synchronous Data Acquisition and Timing Control," in Real Time Conference (RT), 2012 18th IEEE-NPSS, Berkeley, CA, Jun. 2012.

[6] M. Lipinski, T. Wlostowski, J. Serrano, and P. Alvarez, "White Rabbit: a ptp application for robust sub-nanosecond synchronization", ISPCS, Munich, Germany, (2011).

[7] G. Daniluk. White Rabbit calibration procedure v1.0.Website: http://www.ohwr.org/documents/213

[8] H. Li, G. Gong et al., "Temperature Effect on White Rabbit Timing link", IEEE Trans. Nuclear Science, Vol. 62, Issue 3, pp 1021-1026 2015.

[9] M. Kreider et al., "Etherbone – A Network Layer for the Wishbone SoC Bus", WEBHMULT03, Proceedings of ICALEPCS'11, Grenoble, France, (2011).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A GENERIC TIMING SOFTWARE FOR FAST PULSED MAGNET SYSTEMS AT CERN

C. Chanavat, M. Arruat, E. Carlier, N. Magnin, CERN, Geneva, Switzerland

Abstract

At CERN, fast pulsed magnet (kicker) systems are used to inject, extract, dump and excite beams. Depending on their operational functionalities and as a result of the evolution of controls solutions over time, the timing controls of these systems are based on different hardware architectures that result in a large disparity of software solutions. A Kicker Timing Software (KiTS), based on a modular hardware and software architecture, has been developed with the objective to increase the homogeneity of fast and slow timing control for fast pulsed magnet systems. The KiTS uses a hardware abstraction layer and a configurable software model implemented within the Front-End Software Architecture (FESA) framework. It has been successfully deployed in the control systems of the LHC and SPS injection kickers, the SPS extraction kickers and the SPS tune measurement kickers.

INTRODUCTION

A kicker system must meet two requirements that are contradictory i.e. a high deflection strength and a short rise time of its magnetic field. Both properties are, for a given operational voltage, proportional to the product of the electrical current passing through the magnet and its length.

In order to meet these contradictory requirements, the length of the magnet is reduced by splitting the system into several independently powered magnet modules. Generally, the powering circuit of a magnet module consists of a set of well identified hardware components:

- A resonant charging power supply (RCPS) used to charge a pulse generator in order to decrease the time interval between two successive pulses and reduce the number of faulty shots;
- A line type pulse generator based on a pulse forming network (PFN) supplying quasi rectangular current pulses with variable length and amplitude;
- Up to three high voltage fast switches used to transfer in a controlled way the energy stored in the PFN to the magnet and to adjust the pulse length;
- A coaxial transmission line (TL) connecting the generator to ferrite type magnets;
- A ferrite magnet, built as lumped parameter delay lines and working in ultra-high vacuum;
- A termination resistor (TR) matched to the characteristics impedance of the generator, transmission line and magnet and absorbing the pulse energy supplied by the generator.

On this basis, the simplest kicker system comprises at least one PFN charged by one RCPS and discharged through the TL within one terminated magnet by one high voltage switch as shown in Figure 1.

Figure 1: Simplest kicker system architecture

Although peak current, system impedance, pulse shape, pulse duration and repetition rate are different for every system, the different hardware components have been standardised as far as possible, and, depending of the kicker functions (beam injection, extraction or excitation), the number of RCPS, PFN, high voltage switches and magnet are combined in more or less complex architecture in order to provide the required functionalities. Different types of combination used in the SPS and the LHC are summarised in Table 1.

Table 1: Example of Kicker Architecture

System	RCPS per System	PFN per RCPS	Switch per PFN	Magnet per PFN
SPS Injection	4	2	3	2
SPS East Extraction	1	5	2	1
SPS West Extraction	1	1	1	4
SPS Tune	4	1	3	1
LHC Injection	2	2	2	1

Up to now the timing control of these different hardware combinations was based on dedicated real-time software strongly linked not only to the kicker hardware configurations themselves but also to the beam process where the system was used. With time, this approach has resulted in a high number of software packages to be maintained and to a high dependency of each software package with machine operation conditions.

Additionally, as this approach has been used for more than 40 years, a high diversity of electronic modules is used to generate the delays needed to trigger the different high voltage switches. As the access to these different timing delays is strongly embedded inside the real-time software, maintenance is now becoming more and more difficult due to the obsolescence of the delay modules and the difficulties to replace them without having to do a full re-engineering of the actual software.

In order to solve these two problems, a generic kicker timing software constructed on the basis of the existing set of standardised hardware components has been

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

developed and successfully deployed across a first set of kicker systems.

KICKER TIMING SOFTWARE (KITS)

The control of a kicker is divided into two functional subdomains:

- The *Slow Control* has the role to control the operational state (On, Off, Standby, Faulty) high-voltage pulsed generators and to protect the system against any internal failure modes.
- The *Fast Control* has the role to control the dynamic performance of kicker in accordance with the operational parameters driven by the control room and in perfect synchronization with the timing and the beam of the machine. Through the control modules of the equipment, it generates and distributes the references and the necessary stimulus for the RCPS and PFN to produce the kicks.

The KiTS, a generic software solution, has been developed for homogenisation of the fast control of all the kicker systems across the whole accelerator complex at CERN. Then it is highly bound to the RCPS and PFN hardware components, the control modules and the timing.

Architecture

The operation of the KiTS has to be real time because it should, depending on the settings sent by the control room, provide the required stimulus and references by following timing constraints that must be rigorously respected.

At CERN the FESA Framework [1] has been developed (Front-End Software Architecture) in order to develop object oriented classes for the control of accelerators equipment. FESA provides real time features linked to interrupts coming either from the accelerator central timing system or from the low level equipment hardware (timing or external events synchronization) and standard communication interfaces with external applications. The KiTS has been developed on the basis of this framework and thus benefits from all the features intrinsic to FESA listed above.

As highlighted in the introduction, kicker systems at CERN are based on a reduce set of hardware power components (RCPS, PFN, switch and magnet). Usually all kicker systems do not have the same number of components and their association can differ significantly from one equipment to the others.

In order to rationalise the kicker logical architecture, the association of RCPS with PFN can be called a *Generator*. On this basis, kicker systems can be represented as different configurations of generator (see Fig. 2) each composed of a set of RCPS and PFN.

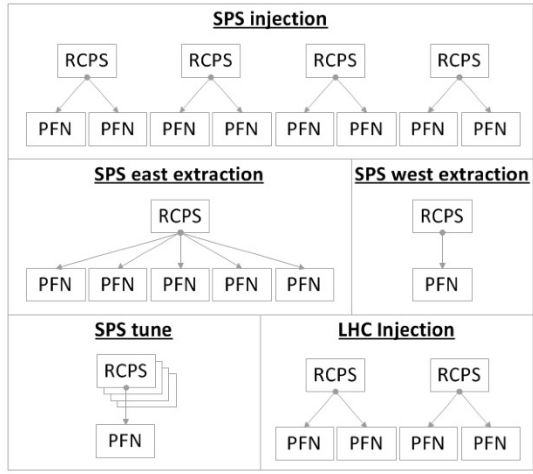


Figure 2: Simplified software model of different type kicker generator architecture

This hardware representation of a kicker system can then be easily modelled within software through an object oriented approach. The KiTS system offers a configurable object-oriented model consisting of three independent FESA classes (see Fig. 3).

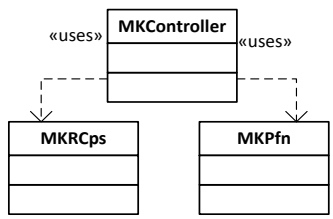


Figure 3: KiTS simplified FESA class model

The *MKRcps* FESA class handles the features of the RCPS hardware components. There are as many instances of the *MKRcps* class as the number of active RCPS elements required per kick in the system. This class manages the kick strength references sent to the power supplies and the timing delays related to the trigger the resonant charging process to load the PFN just before pulsing. It also controls the load balancing of the kick strength distribution between the different RCPS hardware components available for a kick.

The *MKPfn* FESA class handles the features of PFN hardware components (i.e. high voltage switches connected to it). There are as many instances of the *MKPfn* class as the number of active PFN elements required per kick in the system. This class manages the kick synchronisation with beam, the fine internal synchronisation when more than one magnet is used per kick, the kick length and some internal delays when pre-trigger signals are required. It generates the delays that trigger the switches present on the PFN in order to discharge it:

- A first set of delays that re-phase all PFNs relative to a START event synchronised with the beam;
- A second set of delays per PFN for staggering the PFNs with respect to each other.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The *MKController* FESA class is the controller of the KiTS system. It has only one instance (singleton) per system (kicker). It gathers and analyzes information from the control room and from the kicker to provide parameters and operating data to *MKRcps* and *MKPfn* classes. Furthermore, it can generate the required timing when it cannot be obtained by timing or external sources as an example when a kicker system has to be pulsed in local mode.

Each instance of each class has a specific set of configuration parameters in order to define to the operational configuration. Thanks to this FESA model, the KiTS can be adapted to most of kicker generator configuration existing at CERN.

These 3 classes are also based on a second abstraction framework which introduces two standardisation layers (see Fig. 4):

- A business logic layer that homogenises the business common features of control applications in order to make the access to the hardware layer uniform, and to allow to share the business logic commands between the FESA actions.
- A hardware logical layer that homogenises the hardware common features of control applications, and makes uniform the access to the hardware factory.

Thus, we obtain a system of classes where all the classes have a homogeneous architecture where the common features are generalized into Frameworks and where the specific functions of the different logic layers are well separated.

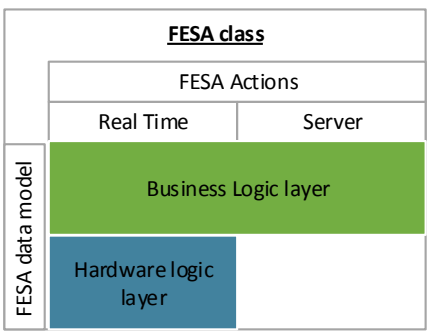


Figure 4: KiTS software layers

Hardware Abstraction

To interact with the active components of the kicker, the real-time tasks uses different types of hardware modules (see Tab. 2).

These hardware modules are not the same on the different kicker systems, then the KiTS must be able to use all these different types of hardware modules to control the different kickers. To do so, it uses a software library based on the factory design pattern which creates and allows to use abstract software objects such as ADC, DAC, DELAY, TDC, SAMPLER, etc. The factory hardware of each abstract type contains the implementation to drive several models of hardware module.

Table 2: List of Modules in the Hardware Factory

Category	Reference	Manufacturer
DAC	VMOD12A2 VMOD12A4	Janztec AG Janztec AG
AG	VMOD12E16	Janztec AG
Pulse delay	CTR V850 / V851 FMC-FD	CERN HIGLAND Technology CERN
Time-to-digital converter	TSM FMC-TDC	CERN CERN
Sampler	VD80 FMC-ADC	INCAA Computer CERN
Digital I/O	CTR VMOD-TTL	CERN Janztec AG

For example, the DELAY hardware factory contains the implementation of the following modules V850, V851, CTR, FMC-FD as shown in Fig. 5.

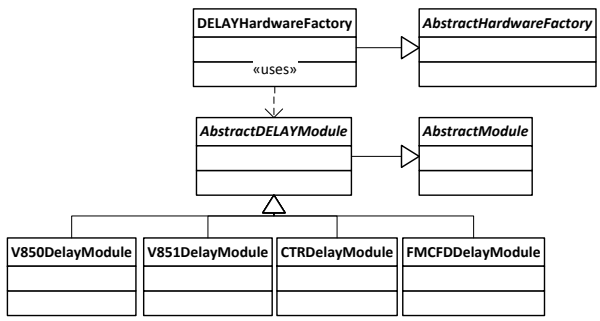


Figure 5: The Delay module factory simplified class model

The list of hardware modules of a factory can be change without any impact on the client applications code as a generic API has be defined for each category of module.

These abstract software objects allow the KiTS to control the hardware modules of any type without knowing the exact implementation of the hardware modules used.

Therefore, this abstraction layer allows the KiTS to be fully independent of the models of the hardware modules used for the control of the kickers.

Timing

Kicker systems work with different timing sources according to the accelerator to which they are used (LHC, SPS, PS…) and with different families of timing events depending of the beam process for which they are used (injection, extraction, tuning). In some cases, some specific events can also be generated by the real-time

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

control system itself (SPS tune kicker of SPS dump kicker) a defined in dedicated configuration files.

To adapt to the different type of kicker timings, the KiTS is based on an event model architecture scheduled by four virtual events. This virtual event model serves as generic timing structure for the KiTS to drive a kicker. These four virtual events are associated by configuration to hardware timing events from the accelerators central timing.

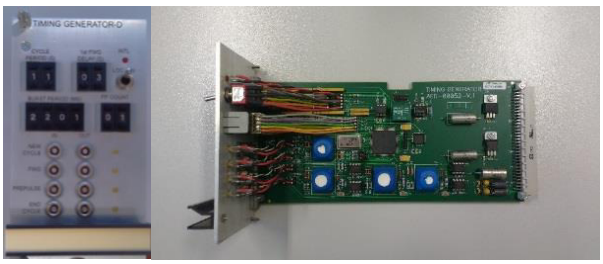


Figure 6: TG-D Module

Before reaching the KiTS, the timing hardware events are controlled by a Timing Generator module (see Fig. 6) providing the following functionalities:

- Control of the correct scheduling of the four events;
- Generation of the four events when the kicker is operated in local;
- Inhibit of all events in case of interlock condition coming from the slow control.

The TG-D module also plays the role of local timing generator which allows pulsing the kicker independently of the central timing.

From the four primary events, it is possible to generate as many secondary events as needed to control a kicker (see Tab. 3). To be noted that a full beam process is always enclosed between a *Start Cycle* and an *End Cycle* event and that between these two events, as many sub-processes as desired can be played.

Table 3: List of Primary and Secondary Events

Primary events	Secondary events
Start Cycle (NC)	
Forewarning (FW)	Forewarning prepared Before RCPS Before prepulse After prepulse
Prepulse	
End Cycle (EC)	

These primary and secondary events are used to trigger the KiTS FESA class real time actions that will call the KiTS FESA class business logic commands (see Tab. 4).

This model of virtual events allows the KiTS to be easily adapted to specific timing requests of different kicker systems in operational mode or test mode.

Table 4: List of KiTS FESA Classes Commands

Class	Commands called by RT actions
MKController	• Reset Equipment • Get User Context • Start / End Timing Cycle • New / End Cycle • Forewarning, • Prepulse, After Prepulse • WatchDog
MKRcps	• Reset Equipment • New / End Cycle • Write / Read / Reset Vref • Start / Stop Sampling • Write Trigger • WatchDog
MKPfn	• Reset Equipment • New / End Cycle • Write Fine Tuning • Write Prepulse • Write / Read Fast Timing • Single PFN Voltage Acq • Start / Stop Sampling • Check / Notify PFN Voltage • WatchDog

CONCLUSION

Thanks to its object oriented architecture, its virtual timing model and its hardware abstraction library, the KiTS has been successfully deployed on different types of kicker system at CERN.

During LS1, the end of support of FESA2.10 has required the migration of the KiTS real time kernel to FESA3. This update has involved an extensive re-engineering of the real-time code in order to remain fully compliant with the new FESA framework. After the successful migration and series of validation tests, the KiTS has been easily deployed to all kicker systems already equipped with it proving the benefits to use a generic approach.

For the kicker systems not yet equipped, the use of the KiTS instead of developing a dedicated software solution has proven to be a more optimised approach reducing significantly development and validation resources, provided that all the required functionalities are already integrated.

The KiTS will be facing more challenges that will continue to profoundly assess its flexibility, modularity and adaptability. Its planned deployment on the renovated PS Booster distribution and consolidated SPS Beam Dumping kicker systems are good examples.

REFERENCES

[1] M. Arruat et al., “Front End Software Architecture”, WOPA04, ICALEPCS’11, Knoxville, TN, USA, 2011.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

STATUS DEVELOPMENT OF SIRIUS TIMING SYSTEM

J.L.N. Brito*, S.R. Marques, D.O. Tavares, L.A. Martins, LNLS, Campinas, Brazil

Abstract

Sirius is a new low-emittance 3 GeV synchrotron light source under construction in Brazil by LNLS, scheduled for commissioning in 2018. Its timing system will be responsible for providing low jitter synchronized signals for the beam injection process as well as reference clocks and triggers for diverse subsystems such as electron BPMs, fast orbit feedback and beamlines distributed around the 518 meters circumference of the storage ring, Booster and Linac. It will be composed of Ethernet-configured standalone event generators and event receivers modules developed by SINAP through a collaboration with LNLS. The modules will be controlled by remote EPICS soft IOCs. This paper presents the system structure and the status of the development, some options for integrating it to the Sirius BPM MicroTCA platform are also discussed.

INTRODUCTION

Sirius light source consists of a 150 MeV Linac, a 150 MeV to 3 GeV booster synchrotron and a 3 GeV, 518 meters circumference, storage ring with 20 straight sections, achieving a very low beam emittance of 0.24 nm·rad [1]. Many devices must be triggered synchronously by a low jitter timing system to guide the electrons from the Linac to the storage ring.

The main purpose of a timing system is to generate and distribute deterministic signals to control the beam injection process. Secondary purposes are to provide timestamps for diverse subsystems to ensure a time-coherent behaviour of data acquisition and to deliver reference clock for the electron BPM and fast orbit feedback systems.

The range of possibilities considered to perform the Sirius timing system can be summarized in four major approaches: (i) a completely commercial timing system [2], (ii) a solution entirely based on scientific test and measurement instruments [3], (iii) an in-house development and (iv) collaborative options, such as White Rabbit Project [4] and SINAP timing system [5]. After surveying these possibilities, the SINAP timing system was chosen based on the available functionality it could offer, proven reliability [6] and collaboration opportunity.

REQUIREMENTS

The Sirius injection system operates with a 2 Hz repetition rate, which must be synchronized to the mains at 60 Hz to turn repetitive the ripple effect of the power supplies on the injected beam in the accelerators. Trigger signals should be derived from the RF frequency to synchronize pulsed elements on Linac, Booster and storage ring to electron bunches. The 90 keV e-gun, that will operate in single-bunch and multi-bunch modes, is the pulsed element with the most demanding trigger signal, requiring jitter less than 50 ps rms, coarse delay resolution of 2 ns and fine delay resolution of 20 ps.

After the commissioning phase, top-up is foreseen to be the Sirius normal operation mode, in which a bunch current measurement system to take the bunch current profile may be necessary to perform the bunch selection. Hence, Sirius timing system should support injection to any bunch and software integration to external systems. Furthermore, other requirements include that critical signals needs to be transmitted through optical fibres to minimize EMI, electrical outputs requires TTL level compatibility and inputs for external triggers are essential.

Table 1 presents the main parameters about Sirius and timing system specifications.

Table 1: Main Sirius Parameters and Timing Specifications

Parameter	**Value**
RF freq. (Storage Ring & Booster)	499.658 MHz
Storage ring circumference	518.4 m
Storage ring harmonic number	864
Storage ring revolution freq.	0.578 MHz
Booster circumference	496.8 m
Booster harmonic number	828
Booster revolution freq.	0.603 MHz
Coincidence number	19872
Coincidence frequency	25.144 kHz
Linac RF frequency	2.997 GHz
Repetition rate	2 Hz
Specification	**Value**
E-gun rms jitter	< 50 ps
E-gun trigger coarse delay step	2 ns
E-gun trigger fine delay step	20 ps
General trigger delay step	8 ns
Electrical outputs	TTL level
Event clock freq. ($\frac{RF\ freq.}{4}$)	124.915 MHz

SYSTEM DESIGN

Sirius timing system is event-based [7], i.e. an event generator (EVG) broadcasts event frames to event receivers (EVR and EVE) through a star topology optical fibre network. An event frame consists of an 8-bit event code and an 8-bit distributed data bus. Each decoded event can generate configurable output triggers and each bit of the distributed data bus maps a channel clock, which can be configured to any integer sub multiple of the event clock. Since the event clock is derived from the RF frequency, this and the channel clocks are synchronized. Therefore, an event

* joao.brito@lnls.br

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

receiver can output configurable triggers and synchronized clock signals to several devices in the accelerators. During an injection process, a sequence of event frames is transmitted every 500 ms (repetition rate).

The event receivers differ according to the output type. While EVEs have LVTTL electrical outputs, EVRs have polymer optical fibre and glass optical fibre outputs. The output signals from EVRs are converted to electrical TTL level by the optical to electrical converter modules STD-SOEs and SOEs, in case of polymer fibre, or STD-MOEs, in case of glass fibre. STD-MOEs and STD-SOEs modules are standalone 19 inches 1U boxes with four optical to electrical converter channels, while SOEs are small boxes with only one channel (Fig. 1).

Figure 1: Optical to electrical converters STD-MOE, STD-SOE and SOE.

EVG, EVR and EVE are also 1U boxes and are configurable through Ethernet UDP interface. All modules were jointly specified by LNLS and SINAP and have isolated electrical inputs for interlock purposes. The development and construction were carried out by SINAP. System modules, fibres and cables are listed in Table 2. A diagram of Sirius timing system network and modules involved in the injection process is shown in Fig. 2.

Table 2: System Modules, Fibres and Cables

Module	**Quantity**
EVG	1
EVR	5
EVE	3
STD-MOE	5
STD-SOE	3
SOE	12
MicroTCA timing board	20
Fibre & Cable	**Quantity**
Level-1 glass fibre (250 m)	8
Level-2 glass fibre (150 m)	13
Level-3 glass fibre (150 m)	20
Polymer fibre (50 m)	20
Electrical cable (20 cm)	35

Hardware

The EVG issues event codes and data clocks at the rate of the event clock, where the events are read at the same rate from a block of RAM, writeable through the Ethernet

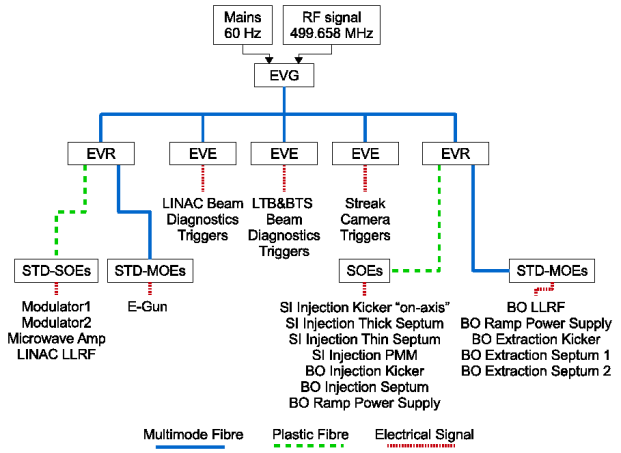


Figure 2: Diagram of Sirius timing system concerning injection process.

UDP interface, used to store a sequence of events. The 2 Hz repetition rate results from the AC line (mains 60 Hz) divided by 30 and it is internally synchronized to the coincidence frequency. On the EVG front panel there are two electrical inputs, one to the RF signal and another to AC line as well as 8 multimode glass optical fibre transceivers to downstream modules.

In the EVR, each output trigger can be configured to be routed to one of the 12 multimode polymer optical fibre outputs on the rear panel or to one of the 8 multimode glass optical fibre outputs on the front panel. The pulse width for each output is adjustable with a resolution of one event clock period (~8 ns). Programmable delays are also available on the front panel outputs, with coarse and fine resolutions of 1/20 event clock period (~0.4 ns) and 5 ps, and on the rear panel outputs with a resolution of one event clock period. Clocks from distributed data bus can be output only on the front panel. Due to the fine delay adjustment of EVR, it is possible to target any bunch by delaying the e-gun trigger with the 1/20 event clock period resolution and other devices with at least one event clock period resolution.

The EVE has 8 electrical outputs mapping decoded clocks or events. Each decoded event generates a configurable output trigger with coarse and fine delay resolutions of 1/20 event clock period and 5 ps, and adjustable pulse width with one event clock period resolution. Another output on the front panel provides a recovered clock derived from the event clock multiplied by 1, 2, 4, 5 or 10, synchronized to the RF.

Distribution Network

The timing distribution network is based on OM3 multimode glass optical fibre operating at 2.5 Gbps rate for distances over 250 meters and multimode polymer optical fibre over distances up to 50 meters. In order to minimize the effect of thermal drifts over the optical links, the length of the fibres is equal on each level.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Software

The timing system software runs an EPICS base-3.14.12.4 IOC on a Linux environment. The IOC is structured in two parts, low and high levels.

The low level implements UDP communication interface, hardware diagnostic and control, i.e. UDP data frames are used to read and write registers that configure and monitor parameters of the modules, such as prescalers of the distributed data bus on the EVG or widths and delays of the EVR outputs.

The high level part controls the injection process by managing the fill pattern, bunch selection and interface with bunch current measurement system to implement the top-up injection mode. Furthermore, it also implements an abstraction layer that converts prescalers into time units and maps the timing system hardware outputs to the connected devices, providing process variables to configure their triggers.

Operator interface panels were developed using CSS (Control System Studio) to monitor and operate the timing system.

Developments

Once the Sirius timing system is able to deliver trigger and clock signals synchronized to the RF, these signals can be used by diverse subsystems, such as electron BPMs, fast orbit feedback and beamlines, to provide reference clocks, perform synchronized actions or timestamps to ensure coherence of acquired data.

Since electron BPM electronics and fast orbit feedback systems adopted the MicroTCA.4 standard [8], a fully compliant timing receiver board is under development by LNLS to distribute triggers and reference clocks for those electronics. A conceptual prototype of a MCH clock distribution board is shown in Fig. 3. Similar approaches with different form factors have been studied to attend beamline demands.

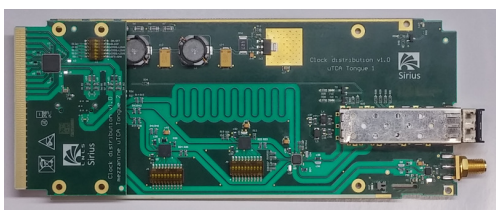


Figure 3: Prototype of the MicroTCA clock distribuition board developed by LNLS.

An important point concerning coherence of acquired data is the post mortem event distribution. For Sirius, this event will be broadcasted over the timing system network to request all relevant devices to store the last buffered measurements. Thanks to the deterministic transmission of the post mortem event, buffers can be precisely time-aligned. This feature will be achieved by a few hardware and FPGA firmware modifications of the EVG, allowing it to receive electrical triggers. A simplified diagram representing part of the timing system network responsible for distributing the signals described in this section is shown in Fig. 4.

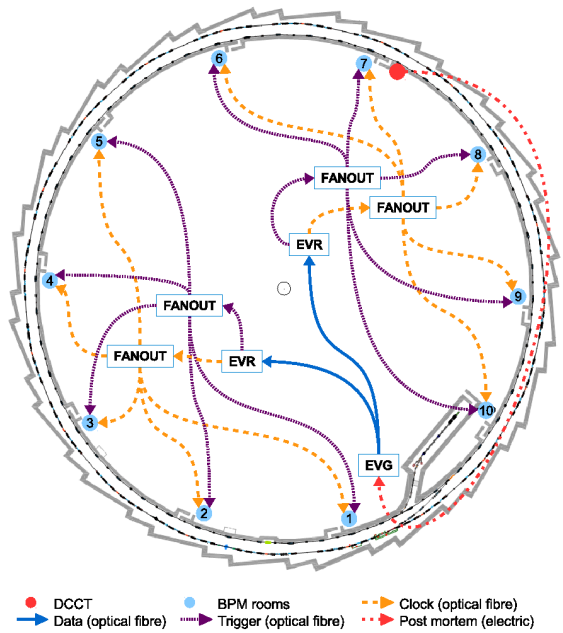


Figure 4: Simplified diagram of Sirius timing system concerning triggers and reference clocks distribution.

TEST RESULTS

The performance achieved by SINAP modules [5] for short-term and long-term (24 hours) jitter between output triggers and the RF master clock were confirmed in tests carried out by LNLS. All modules were verified and typical results indicate short-term jitter values less than 10 ps rms and long-term jitter around 20 ps rms (Fig. 5). These tests were performed using a Rohde & Schwarz SMA100A signal generator and an Agilent DCA-X 86100D 20 GHz bandwidth oscilloscope.

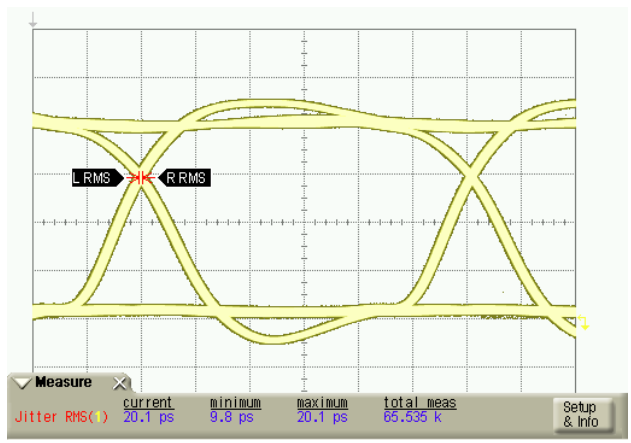


Figure 5: EVE long-term (24 hours) jitter test result.

The short-term jitter of our MicroTCA timing board was characterized using a Rohde & Schwarz FSUP signal source

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

analyzer (Fig. 6). The main purpose was to identify not only the rms jitter, but also the phase noise components of the reference clock signal to source the ADCs of the electron BPM electronics and, for this reason, a particularly relevant bandwidth (1 Hz to 5 MHz) was selected. The measurement is shown in Fig. 7, where the rms jitter is less than 5 ps.

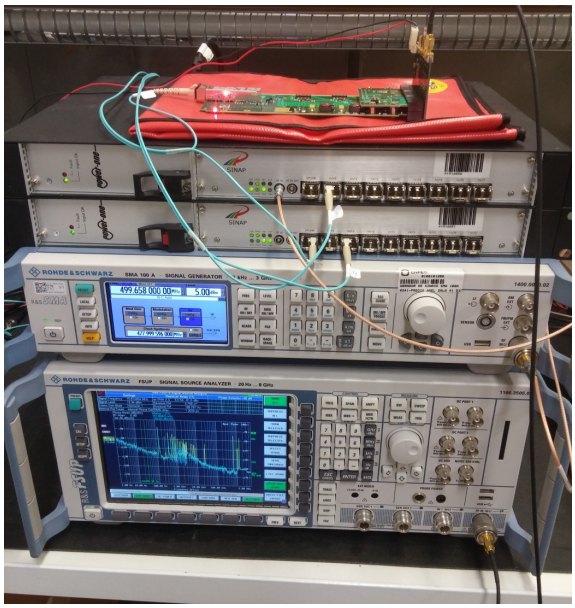


Figure 6: Test setup of MicroTCA timing board. Up to down: MCH board, EVG, EVR, signal generator and signal analyzer.

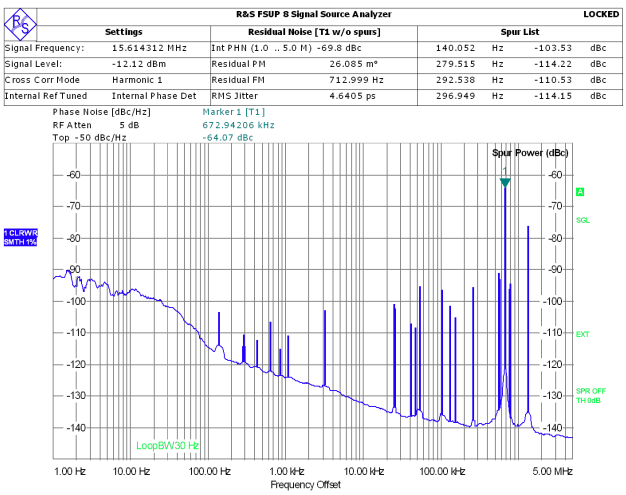


Figure 7: Phase noise analysis of MicroTCA clock distribution board.

CONCLUSIONS

The Sirius timing system structure and functionality necessary to operate during the commissioning phase are completely defined, specifications were achieved and the system has flexibility to accommodate the future developments. Benchmark tests will be performed to characterize propagation delay of optical fibres and to validate the reduced impact of thermal drifts among fibres with the same length. The first units of the MicroTCA timing boards will be produced at the beginning of the next year.

REFERENCES

[1] A.R.D. Rodrigues et al., "Sirius Accelerators Status Report", TUPWA006, Proc. of IPAC'15, Richmond, VA, USA (2015).

[2] Micro-Research Finland Oy, http://www.mrf.fi/

[3] Stanford Research Systems, http://www.thinksrs.com/

[4] White Rabbit Project, http://www.ohwr.org/projects/white-rabbit

[5] M.Liu et al., "Development Status of SINAP Timing System", WEPME032, Proc. of IPAC'13, Shanghai, China (2013).

[6] S.J. Park et al., "Upgrade of Pohang Light Source (PLS) Linac for PLS-II", WE202, Proc. of LINAC'15, Tsukuba, Japan (2010).

[7] F.Lenkszus, R.Laird, "The APS Event System", ICALEPCS'95, Chicago, June 1995.

[8] D.O. Tavares et al., "Development of an Open-Source Hardware Platform for Sirius BPM and Orbit Feedback", WECOCB07, Proc. of ICALEPCS'13, San Francisco, CA, USA (2013).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CERN TIMING ON PXI AND cRIO PLATFORMS

A. Rijllart, O. O. Andreassen, J. Blanco Alonso, CERN, Geneva, Switzerland

Abstract

Given the time critical applications, the use of PXI and cRIO platforms in the accelerator complex at CERN, require the integration into the CERN timing system. In this paper the present state of integration of both PXI and cRIO platforms in the present General Machine Timing system and the White Rabbit Timing system, which is its successor, is described. PXI is used for LHC collimator control and for the new generation of control systems for the kicker magnets on all CERN accelerators. The cRIO platform is being introduced for transient recording on the CERN electricity distribution system and has potential for applications in other domains, because of its real-time OS, FPGA backbone and hot swap modules. The further development intended and what types of applications are most suitable for each platform will be discussed.

INTRODUCTION

At CERN, the accelerator operation requires nanosecond precise timing for many systems that act on the particle beams, such as for injection, acceleration, focusing, measurement and extraction. These systems are synchronised and triggered using the General Machine Timing (GMT) [1] system, which has been custom designed at CERN. GMT electronic boards have been developed for a several bus systems, mainly VME and cPCI. At the time of choice in 2004, PXI and cRIO were not mature enough [2], however, during the last 5 years their success in a large variety of industrial areas [3,4], together with the steady improvement of the graphical system design tools [5] have made the PXI and cRIO platforms an interesting candidate for accelerator systems. But no GMT electronic boards fit in the PXI and cRIO bus systems and the C++ driver software for Linux was not compatible with the operating systems used on these platforms.

The successor of the GMT system [6], called White Rabbit, provides improved timing accuracy and solves most of the GMT's shortcomings, and is actively being developed at CERN. An IEEE standardisation committee has been set up to incorporate it into a new IEEE-1588 standard.

In this paper we describe the adaptation of the GMT to the PXI, the White Rabbit timing on the cRIO and how we envision the White Rabbit on the PXI, to enable CERN engineers and physicists to profit from advantages of industrial platform with CERN timing features.

GMT ON PXI

The CERN accelerator control standards for front-end systems are the VME and cPCI busses. Several equipment groups have chosen PXI, because of their instrumentation needs. However, to be able to use the CERN timing (GMT) a VME or cPCI had to be added to house the timing receiver and run the timing software. The generated triggers were then wired into the PXI either by copper or optical links. When one timing receiver could serve many PXI systems, such as for the collimator control [7], this small overhead is acceptable. For systems with a few PXI crates it would be an advantage if the timing card could be housed in the PXI crate.

For the hardware we use a GMT timing receiver card in PMC format [8] developed at CERN and a PMC carrier for PXI [9] from industry. This carrier is actually a cPCI carrier and therefore compatible with standard PXI crates and PXI Express crates with hybrid slots (PXI/PXIe) (see slot 5 in Fig. 1).

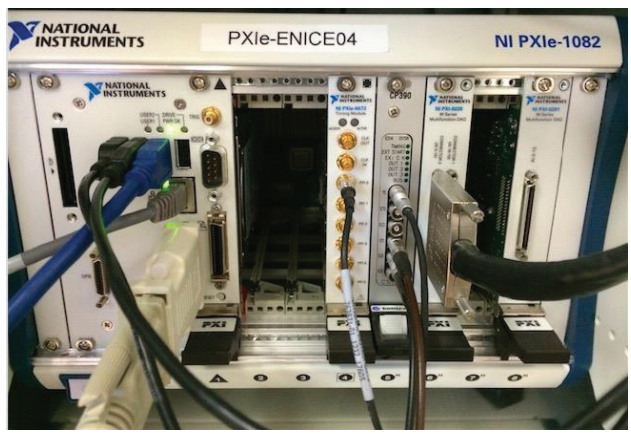


Figure 1: The GMT module installed in slot 5 of a PXI Express crate.

For the software we run a Hypervisor [10] on the PXI controller. This enabled us to run the existing timing library on Scientific Linux 6 on one core of the CPU and the LabVIEW RT system (Pharlap) on the other. In such a system the triggers generated by the GMT receiver were wired to a PXI trigger module that could provide bus triggers for commercial data acquisition cards in the rest of the PXI crate. The Hypervisor solution has also be used in other laboratories for the same purpose, such as in the Brazilian Synchrotron Light Laboratory (LNLS) [11] have used the Hypervisor to profit from the best features of two different operating systems.

WHITE RABBIT ON cRIO

The White Rabbit (WR) node is an open hardware design, accessible from CERN's open hardware repository [12]. Engineers at University of Zürich have used the WR design to build a C-series module for the cRIO platform [13], using the Module Development Kit of National Instruments. They adapted the module on FPGA code and wrote a driver in LabVIEW FPGA for the control and readout of the WR time stamp.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The creators of the cRIO-WR module have made their design available in the open hardware repository on CERN's request [14]. We have had a series of 10 modules produced by a company that had previous WR module manufacturing experience [15].

During the testing we have optimised the LabVIEW FPGA driver code for minimum delay and jitter. Figure 2 shows the cRIO-WR test setup.

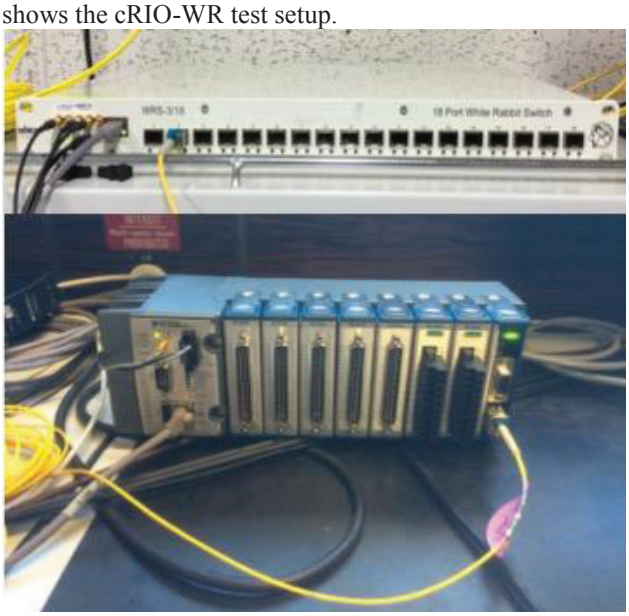


Figure 2: The test setup with the cRIO-WR module connected with the yellow fibre to the WR switch on top.

WHITE RABBIT ON PXI

A White Rabbit PXI Express module has not been manufactured yet, but preparatory work has been done and National Instruments has participated to a plugfest at the ISPCS 2015 conference [16]. National Instruments is one of the main players in the IEEE-1588 High Accuracy standardisation committee with the objective to complement the White Rabbit protocol with several other features. Once this standard has been agreed, I expect to see IEEE-1588 HA "White Rabbit" timing modules on PXIe. Such modules will be able to generate star triggers on the PXIe bus from a WR trigger, which will simultaneously trigger all modules in a crate to a high accuracy (< 0.5 ns skew).

CONCLUSION & FUTURE DEVELOPMENTS

CERN accelerator timing (GMT) modules have been custom designed for the VME, cPCI and PMC busses. This was done before the time PXI has become an interesting alternative. We have integrated the PMC version of the timing module by using a PMC carrier module for PXI and running the timing software on Scientific Linux 6 on a Hypervisor installed on a PXI controller. This allowed run the available Linux library and use all standard LabVIEW RT software and drivers on the same multi-core CPU.

We have profited from the open hardware design made by the University of Zürich of a White Rabbit timing module in cRIO. We have verified the design and had a series of modules produced using a WR partner company. This allowed us to test and optimise the FPGA driver to get optimum performance, which resulted in a minimum of 850 ns delay and 2 ns jitter.

CERN and National Instruments are two of the main players in the IEEE-1588 High Accuracy standardisation committee who's objective is to complement the White Rabbit protocol with several other features. Once this standard has been finalised, I expect to see IEEE-1588 HA "White Rabbit" timing modules on PXIe on the market.

Future developments point in the direction of having Linux RT on both cRIO and PXI platforms. For the new cRIO systems this is already the case and I expect to see Linux RT on PXI Express in the not too distant future. This operating system will allow the coexistence of commercial and open source software, which is what we have achieved with the Hypervisor, but a single OS will simplify the configuration, operation and maintenance of such systems and increase their performance.

We will pursue our objective of using commercial hardware for modules of which there is a large offer by multiple vendors and use custom designs for functions that are specific to CERN, where no commercial offer exists, or is mono vendor only.

REFERENCES

[1] Nanosecond Level UTC Timing Generation and Stamping in CERN's LHC. ICALPECS 2003
[2] https://en.wikipedia.org/wiki/PCI_eXtensions_for_Instrumentation
[3] http://pxisa.org/files/PXI%20at%2015_final.pdf
[4] http://blog.pickeringtest.com/pxi-solve-tough-test-measurement-challenges
[5] http://www.ni.com/impactawards/explore_2015.htm
[6] https://en.wikipedia.org/wiki/The_White_Rabbit_Project
[7] https://accelconf.web.cern.ch/accelconf/p07/PAPERS/MOPAN081.PDF
[8] https://ab-dep-co-ht.web.cern.ch/ab-dep-co-ht/hardware/hardware_detail.htm?name=CTRP&version=2
[9] http://www.directindustry.com/prod/kontron-america/product-29073-1104277.html
[10] https://en.wikipedia.org/wiki/Hypervisor
[11] https://accelconf.web.cern.ch/accelconf/BIW2012/papers/mopg031.pdf
[12] http://www.ohwr.org/projects/wr-node-core/wiki
[13] http://www.ni.com/compactrio
[14] http://www.ohwr.org/projects/crio-wr/wiki
[15] http://www.incaacomputers.com
[16] http://www.ispcs.org/2015/index.html

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

AN UPDATE ON CAFE, A C++ CHANNEL ACCESS CLIENT LIBRARY, AND ITS SCRIPTING LANGUAGE EXTENSIONS

J. Chrin, Paul Scherrer Institut, 5232 Villigen PSI, Switzerland

Abstract

CAFE (Channel Access interFacE) is a C++ client library that offers a comprehensive and easy-to-use interface to EPICS (Experimental Physics and Industrial Control System). Functionality is provided for the synchronous and asynchronous interaction of individual and groups of low-level control data, coupled with an abstraction layer to facilitate development of high-level applications. The code base has undergone major refactoring to make the internal structure more comprehensible and easier to interpret, and further interfaces have been implemented to increase its flexibility, in readiness to serve as the CA host in fourth-generation and scripting languages for use at SwissFEL, Switzerland's X-ray Free-Electron Laser facility. An overview of the structure of the code is presented, together with an account of newly created bindings for the Cython programming language, which offers a major performance improvement to Python developers, and an update on the CAFE MATLAB Executable (MEX) file.

INTRODUCTION

CAFE (Channel Access interFacE) [1, 2] is a modern, C++ client library that provides an intuitive and multifaceted user interface to the EPICS (Experimental Physics and Industrial Control System) [3] native C-based Channel Access (CA) Application Programming Interface (API) [4]. It allows for remote access to control data, encapsulated in Process Variables (PVs) residing in EPICS Input/Output Controllers (IOCs), while sheltering the user from the intricacies of programming with the native CA library. CAFE's conception arose from requirements foreseen by SwissFEL, Switzerland's X-ray Free-Electron Laser [5], coupled with the desire to avoid deprecated CA APIs propagating into a new project. Its development has since encompassed a renewed effort as requirements for application development for the forthcoming commissioning phase became apparent [6]. In particular, it was recognized that the effort afforded to a complete API, that further provided abstract layers for beam dynamics applications, could be readily incorporated into any C/C++ based scripting languages of choice. The main advantages to this approach are:

- The inherent simplicity and convenience of maintaining a single CA interface code. New CA functionalities from future EPICS 3 releases need only be integrated into a single base library.
- A uniform response to errors and exceptions that facilitates trace-backs.
- The CA class is well separated from the internals of the domain language meaning that bindings to other scripting and domain-specific libraries are vastly simplified.

An overview of the structure of the code is given, together with a discussion on design features that provide control over configurable components that govern the behaviour of interactions, and guarantee that the outcome of all method invocations are captured with integrity in every eventuality. Newly created bindings to Cython are then presented, and the improvement in performance yielded to Python developers is emphasized. An updated version of CAFE's MATLAB Executable (MEX) file [7], that exposes new functionalities to MATLAB [8] users, has also been made available.

CAFE C++ IMPLEMENTATION

The C++ interface to the EPICS Channel Access client library follows sound practices in CA programming [9] by placing careful attention to:

- Management of client-side CA connections.
- Memory optimization, particularly when connections are restored.
- Separation of data retrieval from its presentation.
- Strategies for converting between requested and native data types.
- Caching of pertinent data related to the channel and its state.
- Aggregation of requests for enhanced performance.
- Adaptive correction procedures, e.g. for network timeouts.

CAFE provides functionality for synchronous and asynchronous interactions for both single and groups of channels. All transactions report their data to a multi-index container provided by the Boost C++ libraries [10]. Here, the container takes ownership of instances of the "Conduit" object, each of which acts as the storage location for all data related to its associated PV. Multiple, distinct interfaces (or indices) provide convenient access to the object elements, allowing their data to be quickly retrieved or modified. The handle index (or object reference) has been configured with a unique key, and as such, acts at the definitive reference to the resource's data. To facilitate data access the underlying container is also indexed by PV, PV Alias, and the CA Channel Identifier. Callback functions have been implemented on all operations involving connection handlers, event handlers and access right handlers. Their invocation triggers their data to be written into the "Conduit" container object. In this way, the connection state of the channel, and all the channel's parameter values, are recorded with integrity. Memory to hold the channels data is allocated dynamically on first connection, and re-examined in the event of re-connection. Only one connection per channel is ever established, unless the channel is also member of a synchronous group (which itself

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

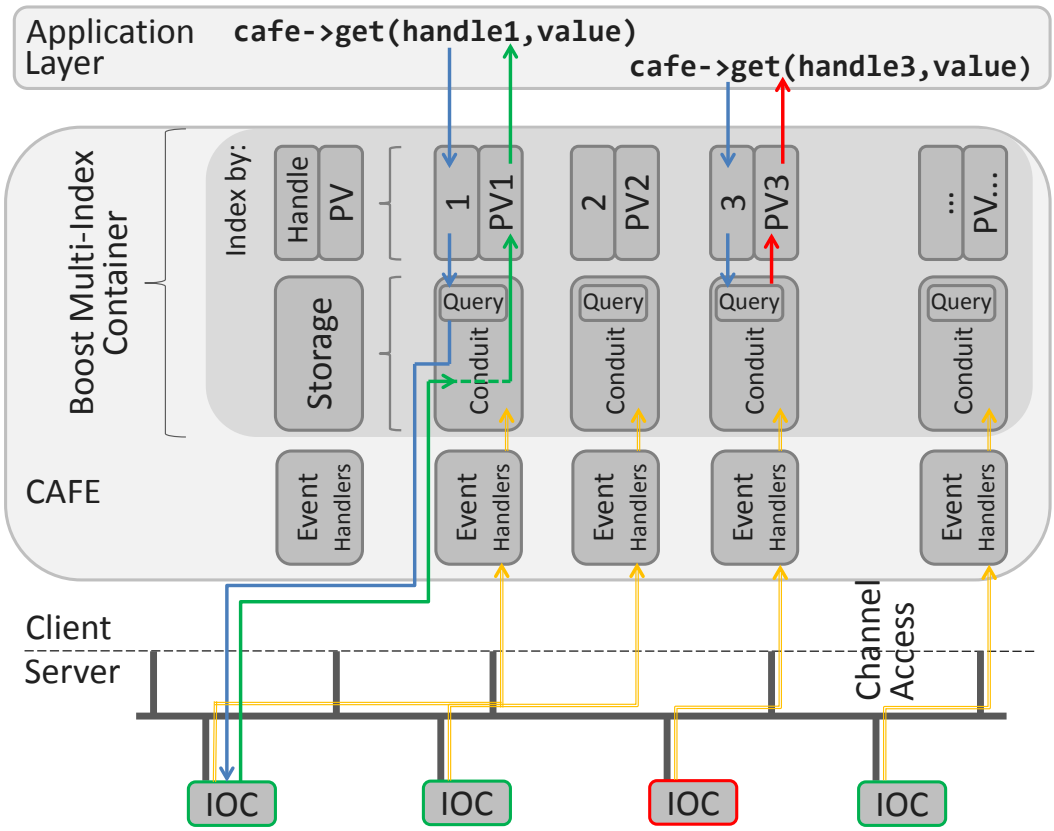


Figure 1: The information flow for a cafe method invocation in the case of a connected channel, PV1 (green), and a disconnected channel, PV3 (red). The multi-index container ("Conduit" object) serves as the data store for the full complement of the PV's data, whether static or dynamic. The handle (index) is the reference to the resource's data. PV data emitted from the IOC is recorded within the container (yellow); cafe method invocations first query the container to assess whether the prerequisites for launching a message have been met.

exists as a logical software entity), in which case a separate virtual circuit is created and a new handle assigned.

All commands first query the container to establish whether a message need be sent over the network. This is illustrated in Fig. 1, which shows the response to a method invocation on a connected channel, PV1, and disconnected channel, PV3. Only if the prerequisites for the method invocation are satisfied, e.g. the channel is connected, is the message sent to the IOC. A hierarchy of software procedures ensures that the most appropriate, and immediate, error message is reported to the client. For instance, a message to PV3 will report either that the channel is presently disconnected or has never been connected, if that be the case.

Policy classes provide control over configurable components that govern the behaviour of interactions, either on a global or individual basis. These include policies that determine procedures to establish connectivity, blocking or non-blocking method invocations, a strategy for converting between data types that are requested and offered, and allow for network time-outs to adapt dynamically.

An XML configuration mechanism provides a convenient framework for users to define and initialize CAFE groups. Transactions on CAFE group objects are invoked through simple intuitive method invocations that reference groups through their identifier.

In addition to refactoring the internal structure of the code, the CAFE interface has also been extended to facilitate bindings to scripting languages (e.g. C++ vectors map directly onto Python lists). Among the new features that have been added is a mechanism to enable and simplify multi-dimensional scan procedures.

SCRIPTING AND DOMAIN-SPECIFIC LANGUAGE EXTENSIONS

CAFE, by design, was not only intended to be sufficiently expressive to provide the required abstractions in a convenient way, but also flexible enough to act as a CA gateway for other, underlying C/C++ based programming languages. As discussed in [2], the idea of providing a single C++ library for use across a number of scripting and domain-specific languages (DSLs) enforces a logical boundary between components of the CA API and the specifics of a given domain's C/C++ extension framework. In this way, the CA classes are not confined to the system in which they execute. Code reusability avoids repetition, reduces maintenance,

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

and vastly simplifies the creation of bindings. Indeed, exposing CA functionality to a given domain begins to follow a definite and recognizable pattern, which, after careful validation of input arguments, largely reduces to a mapping of CAFE/C++ data types to their domain equivalent. The CAFE C++ library has sufficient breadth and maturity to act as a suitable host for scripting languages. Whatever it may be missing, should be typically straightforward to address. Bindings have been provided for Python and MATLAB, which are the languages of choice for physics application developers at SwissFEL.

CyCafe: Syphoning CAFE with Cython

Cython [11] is a programming language that offers a Python-like style of coding while maintaining the performance advantages of C. Its prime capabilities are to compile Python code into C or C++ source code, and to interface with external C/C++ libraries. With both the powerful Cython (version 1.22) constructs and the full Python language available for disposal, an optimized Python interface has been developed for CAFE (CyCafe). The Python extension module is named PyCafe. A number of CAFE methods exposed to PyCafe are shown in Listings 1 and 2. The improvement in performance for a single channel operation is a healthy factor of four compared to a pure Pythonic channel access operation. Some important and novel particularities of providing the CyCafe API are revealed in the following.

Python Buffer Protocol: The introduction of a new buffer protocol allows a number of Python built-in types, such as `memoryview`, the `ndarray` arrays from the much used NumPy scientific computing package [12], and other objects that implement the protocol, to share their data without the need for copying. Cython can similarly extend the buffer protocol to work with data arising from external libraries. CyCafe consequently provides interfaces that cater for the new buffer protocol. In particular, their use in reading/writing waveforms results in a marked improvement in performance.

C Function Pointers and Callbacks: Cython supports C function pointers allowing C functions, that take function pointer callbacks as input arguments, to be wrapped. This feature allows users to pass a Python function, created at runtime, to control the behaviour of the underlying C function. Using this methodology, a Python callback function may be easily supplied for any asynchronous CA interaction. This is highlighted in Listing 2, where a monitor on a PV is activated. A novel aspect of the CyCafe interface, here, is that only the handle (i.e. object reference) is, and need be, reported back to the callback function. Since the CAFE API takes the provision to cache the data in its internal storage area, defined by the multi-index container, the user may call upon any one of a number of CAFE methods that retrieve data directly from the cache, as show in line 4 of Listing 2. The precedence of sifting through Python dictionaries is obviated.

Listing 1: PyCafe Read/Write Examples

```
1  import PyCafe
2  cafe = PyCafe.CyCafe()
3  cyca = PyCafe.CyCa()
4
5  #handlePV=<handle/'pvName'>
6  #dt=<'int','float','str','native'(default)>
7  #hpvList=<hList/pvList> i.e. handle/'pvName' list
8  #s gives overall status, sList is a status list
9  pvList=['pv1','pv2','pv3','pv4']
10 try:
11   handle= cafe.open('pvName') #returns obj. ref.
12   hList = cafe.open( pvList ) #ret. obj. ref. list
13 except Exception as inst:
14   print inst
15
16 #Synchronous Single Channel Operations
17 try:
18   #get value in native type
19   value = cafe.get(handlePV)
20   #returns structured data
21   pvData= cafe.getPV(handlePV, dt='float')
22   #write operation
23   cafe.set(handlePV, pvData.value+0.001 )
24   #waveform, return list in native type
25   valList = cafe.getList (handlePV)
26   #waveform, return memoryview of floats
27   memview = cafe.getArray(handlePV, dt='float')
28   #waveform, return numpy array in native type
29   npArray = cafe.getArray(handlePV, asnumpy=True)
30   #set waveform; input [values] may be any of
31   #list, memoryview, numpy.ndarray, array.array
32   cafe.set(handlePV,[values])
33   #Get cached controls data
34   pvCtrl = cafe.getCtrl(handlePV)
35   pvCtrl.show() # print all control parameters
36   print "units = ", pvCtrl.units
37   print "precision = ", pvCtrl.precision
38   print "enum options (if any): ", pvCtrl.enum
39 except Exception as inst:
40   print inst
41
42 #Synchronous Multiple Channel Operations
43 valList,s = cafe.getScalarList(hpvList)
44 s,sList   = cafe.setScalarList(hpvList, valList)
45
46 #Asynchronous Single/Multiple Channel Operations
47 s,sList = cafe.getAsyn(hList)
48 s,sList = cafe.waitForBundledEvents(hList)
49 pvData  = getPVCache(hList[0])
50
51 #Synchronous Groups
52 #gHandleName=<groupHandle/'groupName'>
53 s = cafe.defineGroup('groupName', pvList)
54 gHandle = cafe.openGroup('groupName')
55 valList,s,sList = cafe.getGroup (gHandleName)
56 s,sList = cafe.setGroup (gHandleName, valList)
57 #returns list of structured data
58 pvgList = cafe.getPVGroup(gHandleName,dt='str')
59
60 cafe.terminate() #tidy up
```

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Listing 2: PyCafe Monitor Example

```
import PyCafe
cafe = PyCafe.CyCafe()
cyca = PyCafe.CyCa()

#Callback function
def py_callback(handle):
  #Any method that retrieves data from cache
  pvData=cafe.getPVCache(handle)
  pvData.show()
  return

#Start Monitor
monID=cafe.monitorStart(handle, cb=py_callback,
    dbr=cyca.CY_DBR_TIME, mask=cyca.CY_DBE_VALUE|
    cyca.CY_DBE_ALARM)
...
status=cafe.monitorStop(handle, monID)

cafe.terminate() #tidy up
```

Global Interpreter Lock (GIL): Cython uses the CPython/API to access C-level code. CPython's memory management is not, however, thread-safe, necessitating a dedicated mutex, the Global Interperter Lock (GIL), to ensure that only one native thread executes Python bytecodes at any given time. External C code that does not interact with Python objects can, however, be executed without the GIL in effect, and thus achieving thread-based parallelism. All methods that access the low-level hardware through CA are done so in a non-GIL context. Without taking this necessary step of releasing the GIL, PyCafe will otherwise hang, particularly in cases where callbacks are involved. A monitor callback can still be invoked even while another CAFE operation is being invoked. All CyCafe methods that interact over the network are consequently called in a non-GIL context, i.e. with the GIL released.

MOCHA: A MATLAB Executable File for CAFE

MOCHA (MATLAB Objects for CHannel Access) is a MATLAB Executable (MEX) file that provides CA functionality to MATLAB through CAFE. MOCHA supports all MATLAB data types, and benefits from MATLAB's 64-bit indexing functionality, granting cross-platform (32/64-bit) flexibility. MOCHA methods may be invoked directly without the need to run additional MATLAB scripts.

The package has been put into good effect at the SwissFEL Injector Test Facility (SITF) [13], where it proved to be a stable and robust API that served the extensive needs of applications developers. The MOCHA MEX file, previously presented in [7], has since been consolidated with the updated CAFE library. As is the recommended practice, a separate MEX file has been compiled for each release version of MATLAB.

SUMMARY

The CAFE C++ Channel Access interface library has been refactored, extended and consolidated, in preparation to serve as the Channel Access host to scripting languages in use for beam dynamics applications at SwissFEL. In particular, CAFE's new Cython interface exposes an extensive set of CA functionality to Python developers and offers a significant improvement in performance. The entire CAFE package may be downloaded from the CAFE website [1]. It would be a relatively straightforward task to extend the scope of this work to other C/C++ based languages once the specifics of the domain's C/C++ extension framework have been grasped.

REFERENCES

[1] CAFE, http://ados.web.psi.ch/cafe/.

[2] J. Chrin and M.C. Sloan, "CAFE, A Modern C++ Interface to the EPICS Channel Access Library", in *Proc. 13th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'11)*, Grenoble, France, Oct. 2011, paper WEPKS024, pp. 840–843.

[3] EPICS, http://www.aps.anl.gov/epics/.

[4] J.O. Hill and R. Lange, "EPICS R3.14 Channel Access Reference Manual", http://www.aps.anl.gov/epics/docs/ca.php

[5] "SwissFEL Conceptual Design Report", R. Ganter, Ed. PSI, Villigen, Switzerland, Rep. 10-04, Version Apr. 2012.

[6] T. Schietinger, "Beam Commissioning Plan for the SwissFEL Hard X-ray Facility", presented at the 37th Int. Free-Electron Laser Conf. (FEL'15), Daejeon, Korea, Aug. 2015, paper MOP017.

[7] J. Chrin, "MATLAB Objects for EPICS Channel Access", in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper MOPPC146, pp. 453–456.

[8] MATLAB®, http://www.mathworks.com/products/matlab/.

[9] D. Zimoch, "Channel Access Client Programming", EPICS Collaboration Meeting, 27-29 Jul. 2009, NFRI, Daejeon, Korea, http://www.aps.anl.gov/epics/meetings/2009-07/.

[10] Boost Multi-index Containers Library, http://www.boost.org/libs/multi_index/.

[11] Cython C-Extensions for Python, http://cython.org/.

[12] NumPy, http://www.numpy.org/.

[13] T. Schietinger *et al.*, "Commissioning Experience and Beam Physics Measurements at the SwissFEL Injector Test Facility", in preparation.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TINE STUDIO, MAKING LIFE EASY FOR ADMINISTRATORS, OPERATORS AND DEVELOPERS

P. Duval, M. Lomperski, DESY, Hamburg, Germany
J. Bobnar, Cosylab, Ljubljana, Slovenia

Abstract

A mature control system will provide central services such as alarm handling, archiving, location and naming, debugging, etc. along with development tools. It has become common to refer to the collection of these services as a 'studio'. Indeed Control System Studio (CSS) [1] strives to provide such services independent of the control system protocol. Such a 'one-size-fits-all' approach is likely, however, to focus on features and behavior of the most prominent control system protocol in use, providing a good fit there but perhaps offering only a rudimentary fit for 'other' control systems. TINE [2] is for instance supported by CSS but is much better served by making use of TINE Studio. We report here on the rich set of services and utilities comprising TINE Studio.

INTRODUCTION

What one ends up referring to as a control system *studio* is primarily all about *the user*. In a perfect world *control* of a machine could be *blind*, where input simply leads to output and there is no necessity of involving something as grandiose as a *studio*. In reality, this will never be the case. Namely, a user operating a machine will inevitably encounter problems which need to be quickly diagnosed and corrected. A user studying the machine will need to readily access all machine parameters. A user developing controls for a machine will also need to produce robust, functional and intuitive applications in a timely manner. A user administering a machine will need to be able to check the integrity of the controls and make modifications where necessary. We refer to the tools the user has at his disposal toward these ends as a *studio*, and where the control system is TINE, as TINE Studio.

The operator and machine physicist will make good use of TINE central services. Namely, the archive and alarm systems and their peripheral applications are major workhorses in machine control.

Likewise, developers will make use of server *wizards* and configuration editors, where front-end and middle layer servers are concerned, and rapid application development (RAD) components and GUI builders, where client applications are concerned.

Finally, control system administrators will profit from the many database management tools, report generators, and monitoring tools available.

In each case the utilities available strive to present an intuitive *cut-to-the-chase* application and take pains to avoid *popup pollution* and needless searching for results in arcane tree structures.

CENTRAL SERVICES

Archive System

An archive system can mean different things to different people. In the context of machine control, we will not refer to the data acquisition system necessary to process the myriads of data taken at an experiment. Instead we refer to the archiving of machine data versus time or versus event. The trend of relevant machine parameters can help diagnose operational problems when it is easy to browse and correlate among them. Likewise, investigating data and settings at a specific event, be it post-mortem, such as an RF trip, or otherwise is also invaluable.

TINE allows the central archiving of any machine parameter with optional storage criteria and does not modify, reduce, or expunge the data in any way after storage. A TINE server may also provide *local* archiving of any parameter.

The TINE archive viewer uses sparse (raster) data with points of interest for rapid browsing over very large time ranges. It also makes use of *optical zooming* whereby a new data request is made whenever a zoom operation over the time interval is made: the finer the zoom, the finer the raster. At any stage the number of points in the selected time range plotted versus the number of points available is easily seen. The archive viewer also decides on the optimal source (central or local) from which to retrieve the archive data.

An archived parameter can be of any available format, be it scalar, multi-channel array, waveform trace array, or video image. The primary archive view is, however, a trend, i.e. a value versus time. Consequently if an array element is selected for viewing then a trend of that element is displayed. It should be noted that the TINE archive viewer can also display the trend versus a system stamp (which might be a pulse or cycle number) instead of a time stamp. In any case, a secondary chart is able to display a snapshot of the entire array (multi-channel or waveform) at the selected (clicked on) time in the trend chart. Similarly, video images will display a frame count in the primary trend chart and the entire image in the secondary chart.

Figure 1 shows the trends of selected parameters in PETRA, including the horizontal tune, over a 24-hour period. The secondary plot shows horizontal tune at the time (green line) clicked on by the operator.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

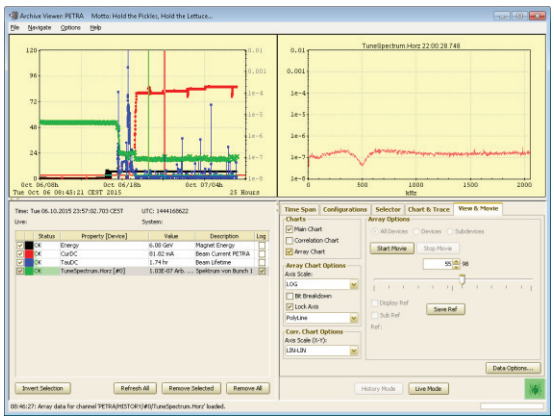


Figure 1: Typical Archive Viewer display.

A third *correlation* chart is also available, where any two parameters can be plotted against one another (and even after applying formulae). Frequently the simple correlations the user makes with his eye in the primary trend chart are all that is needed to pinpoint operations problems.

The TINE event system allows specific data retrieval and storage based on an event trigger. This is most often employed in the case of transient recorders which have a large amount of data in memory following a post-mortem event. A trigger can then instruct the event system to lock and acquire the data and then release the lock when finished.

TINE Studio provides a generic Event Viewer for the general browsing of all events. But as transient recorders tend to be the principal event use-case, a specific Transient Recorder Viewer is also available.

Alarm System

The TINE Alarm system will in many cases provide the first indication of incipient problems and perhaps the smoking gun in cases of operations failures. An *alarm* is in general an entity that belongs to a device. It has a severity and other meta-information and might contain specific data. It also has a duration. Alarms originate at a front-end or middle layer server and are pulled by the central alarm server, where they are also archived.

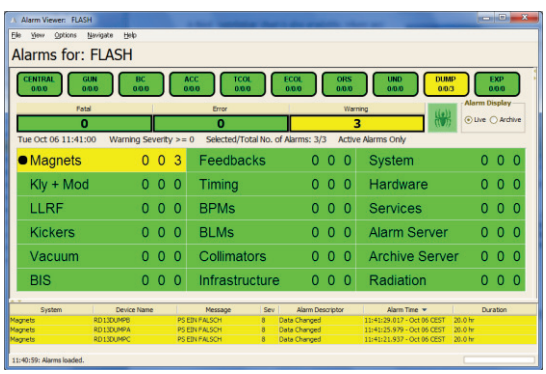


Figure 2: Typical Alarm Viewer display.

Upon start-up, the TINE Studio Alarm Viewer displays a grid-like view of the available alarm systems, showing the number of active *fatal*, *error*, and *warning* level alarms in each, as well as *regional* information. Should an alarm appear in any subsystem, a list of active alarms can be exposed by simply clicking on the alarm subsystem. Specific alarm information can then be obtained by clicking on an alarm in the alarm list (see Figure 2 above).

Operators have the ability at any time to disable an active alarm (or re-enable an alarm), as long as they provide a user name and reason in the *disable alarm* dialog.

Operation Statistics

The TINE Studio Operation History Viewer makes use of the archive and alarm systems as well as the TINE State system. If the machine states are suitably defined and declared, then the amount of time spent in any one machine state can be archived. Likewise the *subsystem-is-available* state (based on there being no fatal alarms in the subsystem) can also be archived. Together this information is used to feed the Operations History Viewer. Here the operators and machine coordinators can easily see how much time a machine has spent in any particular state over any selected time range. Likewise, the availability can easily be examined. In particular, if the machine was not available at some time, the *reasons* can be gleaned by examining the fatal alarm list during the time the machine was not available. In Figure 3 we see that the 24 minutes of non-availability were due to problems in the HF system.

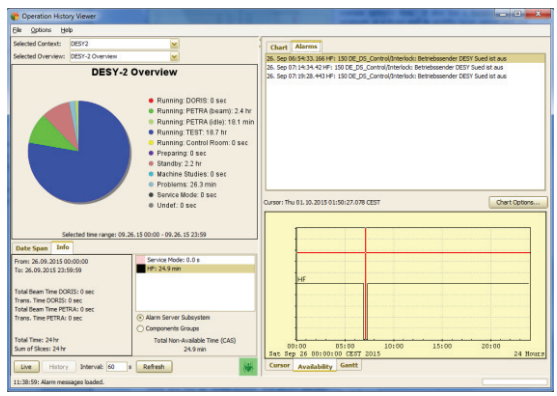


Figure 3: Typical Operation History display.

Generic Viewers

TINE Studio offers several other generic viewers, which have proved their value over the years.

Although the Archive Viewer itself can be switched to *live mode*, a trend-chart viewer, where the live trends of multiple machine parameters can be followed, is also part of the package.

Many relevant control system elements are available as *multi-channel arrays*. These are parameters representing a vector of instances of *the same thing*, with the same units and display settings, such as BPM or BLM values, vacuum pressure, etc. Such control system elements can be easily viewed, browsed, and investigated with the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TINE Studio Multi-Channel Analyser (MCA). Here both live and archive data can be correlated. Figure 4 shows the MCA viewer in history mode.

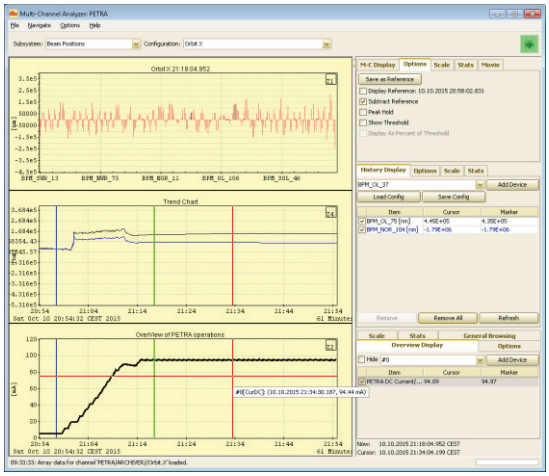


Figure 4: An MCA display. Here: the PETRA Vacuum.

A machine control system typically features a number of servers providing oscilloscope like traces or waveforms. These are best viewed with the TINE Studio Scope Trace Viewer. The user can easily compare live data with data retrieved from either the archive system or the event archive system. See Figure 5 below.

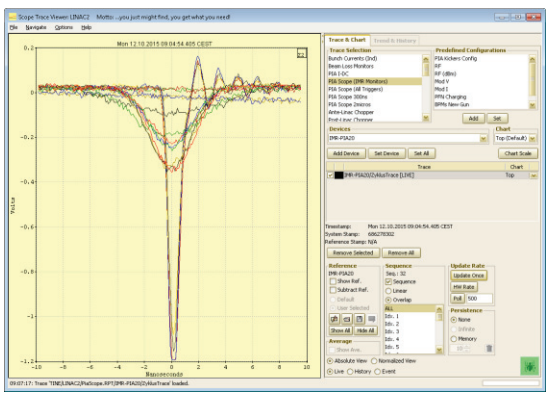


Figure 5: TINE Studio Scope Trace Viewer.

The most generic of all viewers is perhaps the TINE Instant Client. This one tool allows browsing the entire control system of all elements in the entire TINE site. Virtually all manner of control elements can also be viewed with the Instant Client (Figure 6 below), from structured data to video.

ADMINISTRATION

Database Management Utilities

Most TINE central servers make use of a local database to provide their data access instructions. This is true of the archive server, the alarm server, the event server, and the state server. In each of these cases, TINE Studio provides a database management utility, which checks consistency and simplifies the decision making process an administrator faces when he makes editorial changes.

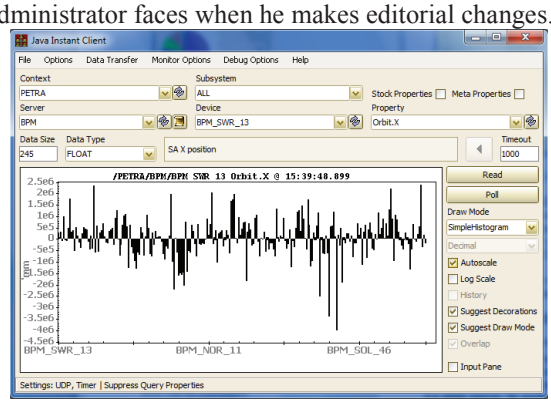


Figure 6: The TINE Instant Client.

Naming Services Utilities

TINE is a plug-and-play system. A server will automatically publish its registered information to the TINE equipment name server (ENS) as well as the TINE group equipment name server (GENS), if the server is registered as a member of a server group (a class of servers with similar devices). From time to time an administrator might wish to update the plug-and-play database maintained by the ENS. Indeed, only an administrator is allowed to assign 'importance' to a control system server. This is best accomplished by the ENS Administration Utility.

System Monitoring

It is generally the task of the control system administrator to monitor the integrity of the control system elements at large. Concerning servers, this task is itself fulfilled to zeroth order by the TINE Watchdog, which is itself a TINE server, responsible for starting, monitoring, and re-starting if necessary all other servers on the same host computer. The watchdog control GUI, shown in Figure 7, provides an interface to all watchdog servers.

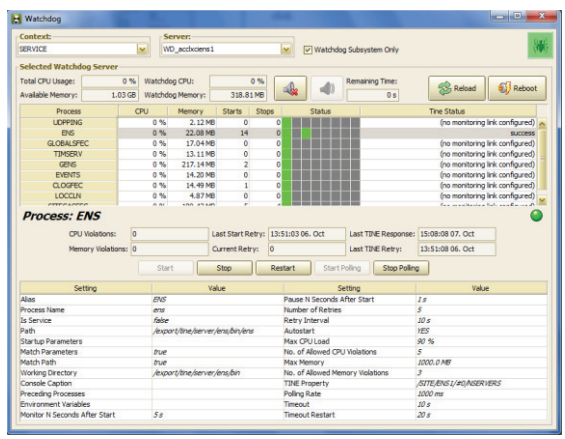


Figure 7: Watchdog control GUI.

Of major benefit to the administrator is the FEC (Front End Controller) Remote Panel. This single panel displays the running status of all servers in a particular TINE

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

context. Log files, alarm and other meta- information, can trivially be obtained for any selected server.

A report of the characteristics (version number, operating system, daemon statistics, ping results, etc.) of all servers in the selected context can be generated with a single button click. Figure 8 shows a typical view of the FEC remote control panel.

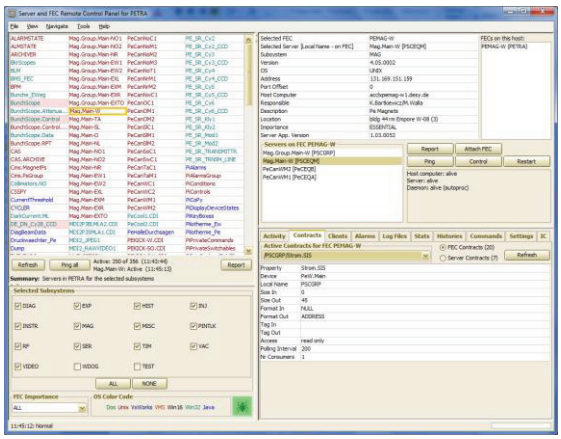


Figure 8: TINE Studio FEC Remote Control Panel.

DEVELOPMENT

Good development tools which go beyond a C or Java API are a godsend. But here we should distinguish between *server* development, where display is not an issue and *client* development, where it generally is.

Server Development

At the hardware level a TINE server can under many circumstances simply be configured as a CDI (Common Device Interface [3]) server. If this is the case then the TINE Studio CDI editor can be used to configure (and deploy and maintain) the necessary CDI database.

Often, a front end server or middle layer server will need to apply additional logic to obtain the finished data sets published by the server. Here the use of the TINE server wizard can be used to generate the configuration files and/or code in selected languages. The developer can then concentrate on making the relevant changes to the generated code.

Rapid Application Development Tools

At the client side GUI (Graphical User Interface) applications are of primary concern. Here we should also distinguish between rich clients, where programming logic (often in a high level language) enhances computation and display, and simple clients, where widgets on a panel are connected to control points but have limited interactions among themselves.

To be sure, TINE offers interfaces to high level scripting languages such as Python and MatLab, which are popular with machine physicists and are necessarily *rich*. On the widget side of the coin, there are also TINE plugs to popular panel builders such as CSS or jDDD [4]. TINE Studio itself, however, does not offer a panel builder per se. Instead GUI developers are encouraged to make use of ACOP (Advanced Component Oriented Programming [5]) components, which can be used in a number of frameworks and languages with minimal (and sometimes no) coding. For instance, *acopbeans* can easily be incorporated in java projects in either eclipse or NetBeans (see Figure 9 below). And ACOP.NET, the successor to ACOP ActiveX, can easily be added to Visual Studio. In such cases an application can become a rich client when necessary but otherwise remain a simple application created by dropping graphical widgets on a panel in a graphical development environment (and connecting them to control system endpoints).

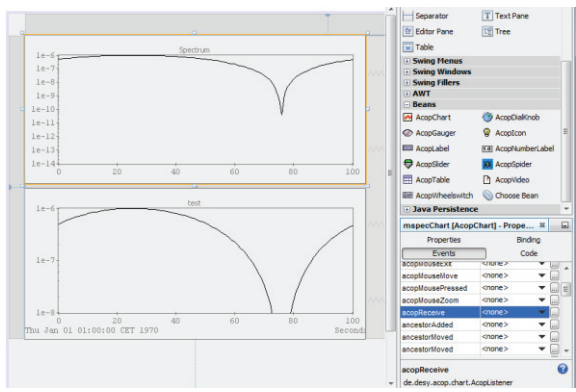


Figure 9: Using ACOP in a NetBeans project.

And in the case of the COMA (Container Object MAnager [6]) class in Java, simple clients can even be created *on-the-fly* without invoking a framework.

CONCLUSION

A *studio* should strive to meet the expectations of all of categories of users, without restricting functionality or performance. This includes the developer along with the operator and the machine physicist and engineer as well as the administrator. The utilities presented to all categories of user should not only provide an intuitive interface but make available the information the user is most likely to need with a minimum of effort. After years of interviews, interactions, and feedback with precisely this set of users, TINE Studio is well on its way to realizing these goals.

REFERENCES

[1] CSS website: http://controlsystemstudio.org/
[2] TINE website: http://tine.desy.de.
[3] Duval, et al., “Common Device Interface 2.0”, PCaPAC 2014, Karlsruhe, Germany, (2014) .
[4] jDDD website: http://jddd.desy.de.
[5] acopbeans website: http://public.cosylab.com/acop/site/
[6] Bartkiewicz, et al., “The Run-Time Customization of Java Rich Clients with the COMA Class”, ICALEPCS 2007, Knoxville, USA, (2007).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

APPLYING SOPHISTICATED ANALYTICS TO ACCELERATOR DATA AT BNL'S COLLIDER-ACCELERATOR COMPLEX: BRIDGING TO REPOSITORIES, TOOLS OF CHOICE, AND APPLICATIONS *

K.A. Brown†, P. Chitnis, T. D'Ottavio, J. Morris, S. Nemesure, S. Perez, D. Thomas
Collider-Accelerator Department., BNL, Upton, NY

Abstract

Analysis of accelerator data has traditionally been done using custom tools, either developed locally or at other laboratories. Much of the data analysis is done in real time when the data is being logged. However, sometimes users wish to apply improved algorithms, look for data correlations, or perform more sophisticated analysis. In recent years we have investigated the use of tools to bridge standard analysis systems, such as MATLAB®, R, or SciPy, to the controls data repositories. In this report we will discuss the tools used to extract data from the repositories, tools used to bridge the repositories to standard analysis systems, and directions we are considering for the future.

INTRODUCTION

Historically our data repositories have not been that large or complex. Over time these repositories have grown and the kinds of data have become more complex. Today it is not unusual to be working with multidimensional arrays or images, as well as simple scalar data. But more significantly, we store far more data than we analyze [1]. If we want to mine that data and look for particular patterns or do complex correlations, our legacy tools are inadequate.

The development of more sophisticated data collection and analysis systems started after 2000, once the Relativistic Heavy Ion Collider (RHIC), within the Collider-Accelerator Complex (C-AD) at BNL, became operational [2]. The new systems that were put in place standardized the data formats across all data sources and placed the data into well-defined filesystem hierarchies. General tools were built to access all data in these repositories.

The operation of RHIC and associated accelerator systems requires comprehensive data logging systems, collecting and storing measurements from many physical systems. Analysis of this data is a critical part of any troubleshooting process and involves the detection and study of composite behavior patterns. Since most of the tools in place to do this are custom built applications with limited functionality, bringing in other systems, such as MATLAB® or R, allows more sophisticated analytics to be applied to the data. However, the main issue isn't so much getting systems that can do sophisticated analytics, but easily linking such systems to the data that has been stored.

* Work performed under Contract Number DE-SC0012704 with the auspices of the US Department of Energy.
† kbrown@bnl.gov

Data Collection and Storage

One step towards making use of other analysis tools is to get the data already collected and stored to be available to any application. To do this we make use of HTTP protocol multithreaded data servers [3].

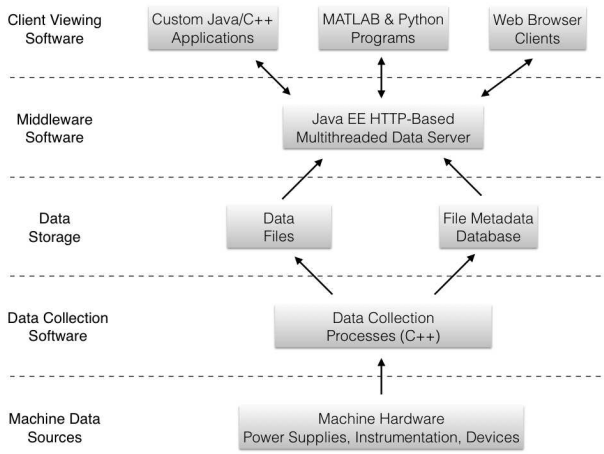


Figure 1: Three tier data layers for Accelerator Data.

Figure 1 shows how data gets collected and stored and then accessed using a data server. All the systems used at the *Machine Data Sources*, *Data Collection Software*, and *Data Storage* layers are legacy systems, for the most part, and serve the purpose of writing data into the filesystem. The *Data Server* provides a simple http-based interface to that data. We also have library level tools that provide this access and tools for obtaining live data.

Data Formats

The RHIC file storage and retrieval systems were designed to make use of the Self Describing Data Sets (SDDS) system developed at Argonne [4]. Using this file format a directory tree hierarchy was built, using a (standard relational) database to facilitate retrieval.

Today, the SDDS format remains the system used to store data, but that choice in format is now a constraint to new design choices. This is an issue in places where performance improvements need to be made. A lesson we seem to keep learning is that the opportunities of one era become the liabilities of a later era. Every custom design eventually becomes either a maintenance burden or a hurdle to progress [5].

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

MODERN TOOLS

The growth of analytics in industry opens opportunities for improving the mining and analysis of accelerator data. However, accelerator and scientific data can possess features that present challenges in using some of these tools. In the accelerator environment there is often significant support for in-situ processing, combining metadata and indexing with large-scale time series and multidimensional arrays. This is one of the challenges towards deploying data crawling techniques, for example, where complex correlations need to be performed.

In this section we will briefly discuss a few potentially beneficial technologies, including some that are in use in accelerator environments.

HDF5

The Hierarchical Data Format (HDF) [6] was designed to be an architecture-independent library and file format for scientific data. Today it is extensively used in many areas of both science and industry.

HDF is not used at C-AD, but we are researching ways to take advantage of this technology. HDF5 is particularly appealing since it is specifically designed to help manage extremely large and complex data sets, implementing an API with C++ and Java interfaces.

The advantages of HDF5 are the versatility of the data model, the performance features that provide access time and space optimizations, and the tools for managing, manipulating, viewing, and analyzing data.

Distributed Databases

Distributed databases are Hadoop [7] related or inspired systems such as Cassandra [8] and MongoDB [9]. These systems allow data to be stored across multiple computers systems where data locality is spread over all systems in the database. The presentation of the data is independent of the actual locality of the data. The data can be spread out over multiple networks, can be joined and updated from multiple tables on different systems, and by distributing the data over multiple systems provides high performance, high data integrity access to large data sets.

We are not using distributed databases in our systems, but we are researching these technologies.

Many laboratories are moving towards distributed database systems using high-performance asynchronous messaging to build scalable high performance data streaming to repositories [10–14]. Parallel computing in the controls network is not common, but laboratories are finding they need to exploit parallelism to gain performance, such as at Spring8 and RHIC [15, 16]. Combining parallel computing technologies with distributed database technologies is a natural extension of both ideas. We are researching such systems, for example Apache Spark [17] bridges these two technologies to gain high performance in data mining and analysis [18].

MATLAB®

MATLAB® [19] is used in our systems largely for offline data processing. Although it is an environment for doing simulations, data analysis, data visualization, as well as providing an interface to other devices and instruments, we mostly make use of the analysis and visualization features in the controls system.

In the MATLAB® environment, scripts are a series of statements that operate on data in the workspace. They do not accept arguments. Functions accept input arguments and return output values. Internal variables are local to the function. Using scripts and functions we can build fairly well featured interfaces. However, MATLAB® is intended to be used from its own environment, so it is not ideal for building full featured applications.

What makes MATLAB® particularly powerful is its extensibility. There are many Toolboxes, as these extensions are called, from which to select. Toolboxes don't just provide greater capabilities, they also allow more complex tasks to be done more easily.

R

R is a free software project aimed at statistical computing and visualization [20]. In some respects much of what can be done in R can be done in MATLAB®. However, there are some use cases where R is more efficient. R also provides the capability to build scripts to simplify the access to the data through the server interface. We have used R only for a small number of off-line studies.

With R we can bring advanced statistical models into accelerator data analysis. Since R is an environment and a scripting language it is also a way of fast prototyping new methodologies in analysis.

There also exists an open source graphical environment for R, RStudio [21].

SciPy

Python [22] is quickly growing in popularity at C-AD. This is partly due to the ability to rapidly prototype new software, but also because Python is highly extensible with a very active user community. One very popular module distribution (a collection of python modules) is SciPy [23], an open source collection of tools for math, science, and engineering.

Much of what can be done by MATLAB® and in R can be done from SciPy. The distribution also includes the h5py package, a Python interface to HDF5. Many accelerator applications are being developed using Python and SciPy at C-AD. For access to the controls data, the HTTP interface data server is used, as well as control systems functions for accessing live data.

Since Python is meant to be a scripting language, as opposed to a scripting environment, full-featured applications can be built. In addition, the community puts significant focus on performance. This makes Python much more com-

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

petitive as a resource for building true controls applications, often replacing Java or C++ as the tool of choice.

EXAMPLES

MATLAB® Controls Data Viewer

The data viewer for the C-AD controls data is a custom built C++ application, called LogView. To offer improved offline analysis to the data we wrote a collection of MATLAB® functions and scripts. These scripts provide an interface to select and display data from the data server using simple menu selection tools. The key to making this tool useful is to package the data in such a way that end users can apply other MATLAB® functions to the data that was selected and displayed (Fig. 2).

The data servers are http protocol based servers. Requests are sent through http messages. The data returned is contained in an xml formatted message. In principle, one can send requests and get data through a normal web browser interface. To do this within the MATLAB® environment we can use the "urlread()" and "urlwrite()" functions. However, these are not the most convenient or efficient functions to use when the data returned is in xml format. Fortunately, there exists a useful function, called "xmlreadstring()", which returns a Document Object Model (DOM) node. Processing the data is simply a matter of working with the DOM package methods. These methods are well documented on the Oracle Java site [24].

To make the data selected for viewing useful it needs to have names the users understand. In MATLAB® all variables are arrays. MATLAB® uses data containers called cell arrays, or cells, to represent variables. Cells have the ability to hold complex representations of various data sets. But to have named variables, MATLAB® provides Structures, which are Cells where each field is named.

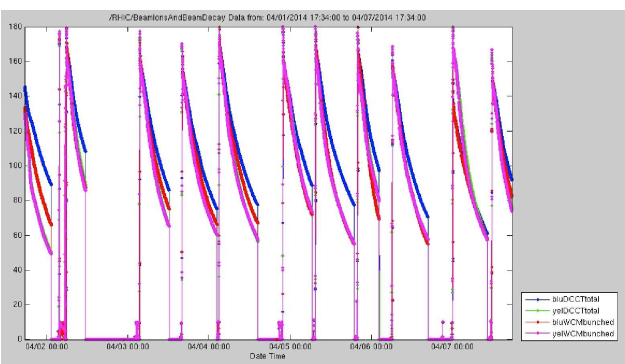


Figure 2: Data collected from the MATLAB® data viewer.

Analyzing Machine Protection Data

The Beam Permit System (BPS) of RHIC plays a key role in safeguarding against the faults occurring in the collider. Over the course of its 15 years of operation, RHIC has accumulated operational failure data. By integration of earlier reliability calculations with operational failure data using Bayesian analysis, the BPS reliability can be quantified using a two-parameter Weibull survival model, with unknown scale and shape parameters. As the joint posterior distribution (Fig. 3) for Weibull with both parameters unknown is analytically intractable, the Markov Chain Monte Carlo methodology with Metropolis-Hastings algorithm is used to obtain the inference [25].

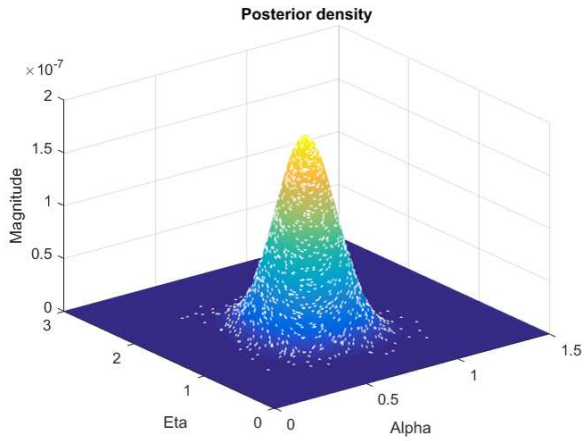


Figure 3: Using MATLAB® to apply advanced statistical techniques to quantify the reliability of the RHIC BPS. Shown here is the posterior density distribution from a Bayesian model of BPS failure.

Developing Better Quench Detector Tables

Due to the limitations of the electrical model used in the RHIC quench detector systems, many false quench events are generated that affect the RHIC availability. The nonlinear electrical behavior of the super conducting magnets was analyzed. The new models include eddy current components and parasitic capacitances, as opposed to simple inductance and lead resistance in the older model. Many data cleaning techniques were employed to reduce the noise in the observed data. Piecewise regression was used to examine the saturation effects in magnet inductance (Fig. 4). The goodness-of-fit of the models was assessed by comprehensive residual analysis. This analysis was done using both R and MATLAB® [26].

Neural Network and Markov Process Models

For this work we have been using NumPy [27] along with SciPy. For neural network analysis (reinforcement learning in the form of a Markov decision process) we use the optimize library for gradient descent to minimize cost functions; specifically the Broyden-Flectcher-Goldfarb-Shanno algorithm (BFGS). This is an interactive method for solving unconstrained non-linear optimization problems, where the Hessian doesn't need to be evaluated directly. In SciPy we can use Levenberg-Marquardt, for doing back-propagation.

Inspection of past data allows development of a model that will provide a prediction of future behavior. This can constantly be adapted as new data is acquired. If something

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

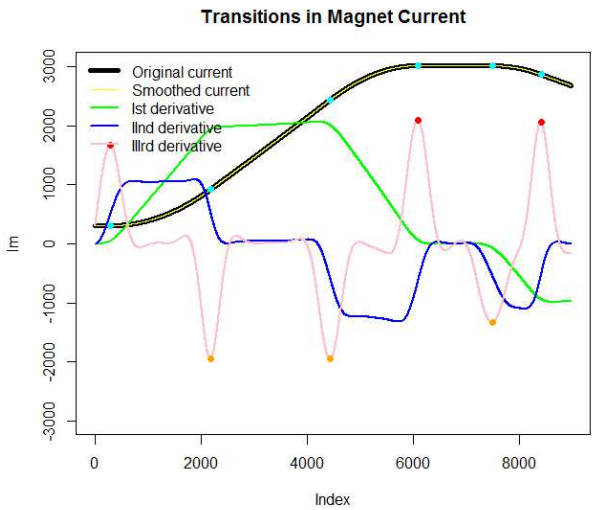


Figure 4: Segmenting data using R to apply different models for deriving superconducting magnet inductance.

changes in all the processes that influence the data being analyzed, then the prediction will deviate. Pattern recognition of past deviations could be used to determine off normal events (Fig. 5).

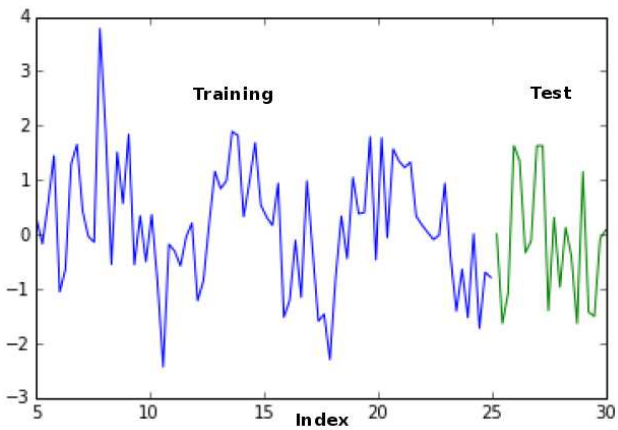


Figure 5: An example of predictive analysis using a neural network derived model of past data.

FUTURE EFFORTS

The ability to efficiently mine and crawl through data to find aberrations and complex correlations remains a future effort. The architecture of the underlying data repository systems needs to be evaluated and methods to mine and crawl the data need to be developed. The classic approaches to data mining of defining pattern associations, path and sequencing analysis, and classification of patterns can be adapted and we could leverage into using existing tools from open source projects or industry (e.g., rapidminer [28], WEKA [29], R [20], Orange [30], KNIME [31], NLTK [32]). These include predictive analytics, machine learning, Bayesian and Markov models, informatics, and natural language processing.

REFERENCES

[1] K. A. Brown et al., ICALEPCS 2013, MOMIB03, http://jacow.org

[2] J. Morris, T. D'Ottavio, ICALEPCS 2001, TUAT002, http://jacow.org

[3] T. D'Ottavio et al., ICALEPCS 2011, MOMAU002, http://jacow.org

[4] M. Borland, PAC 1995, WAE11, http://jacow.org

[5] J. Skelly, J. Morris, ICALEPCS 2001, WC1P05, http://jacow.org

[6] https://www.hdfgroup.org/

[7] https://hadoop.apache.org/

[8] http://cassandra.apache.org/

[9] https://www.mongodb.org/

[10] M. Kago, A. Yamashita, ICALEPCS 2013, TUMIB06, http://jacow.org

[11] S. Marsching, ICALEPCS 2013, MOPPC099, http://jacow.org

[12] N. Kikuzawa et al., ICALEPCS 2013, TUPPC017, http://jacow.org

[13] K. Fuchsberger et al., ICALEPCS 2013, TUPPC026, http://jacow.org

[14] T. C. Shen et al., ICALEPCS 2013, WECOBA06, http://jacow.org

[15] M. Ishii et al., ICALEPCS 2013, MOPPC128, http://jacow.org

[16] B. Frak et al., ICALEPCS 2013, MOPPC157, http://jacow.org

[17] https://spark.apache.org/

[18] N. Malitsky, these proceedings, ICALEPCS 2015, WEPGF058, http://jacow.org

[19] http://www.mathworks.com/products/matlab/

[20] https://www.r-project.org/

[21] https://www.rstudio.com/

[22] https://www.python.org/

[23] http://www.scipy.org/

[24] http://docs.oracle.com/javase/6/docs/api/org/w3c/dom/package-summary.html

[25] P. Chitnis et al., these proceedings, ICALEPCS 2015, MOD3I01, http://jacow.org

[26] P. Chitnis et al., these proceedings, ICALEPCS 2015, MOM310, http://jacow.org

[27] http://www.numpy.org/

[28] https://rapidminer.com/

[29] Mark Hall et al., SIGKDD Explorations, Vol. 11, Issue 1 http://sourceforge.net/projects/weka/

[30] http://orange.biolab.si/

[31] https://www.knime.org/

[32] http://www.nltk.org/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

USING THE VAADIN WEB FRAMEWORK FOR DEVELOPING RICH ACCELERATOR CONTROLS USER INTERFACES*

Wenge Fu#, Kevin Brown, Ted D'Ottavio, Seth Nemesure, Enrique Schuhmacher
Brookhaven National Laboratory, Upton, NY 11793, USA

Abstract

Applications used for Collider-Accelerator Controls at Brookhaven National Laboratory typically run as console level programs on a Linux operating system. One essential requirement for accelerator controls applications is the bidirectional synchronized IO data communication. Several web frameworks have made it possible to develop web based Accelerator Controls applications that provide all the features of console based user interface applications. Web based applications give users flexibility by providing an architecture independent domain for running applications. Security is established by restricting access to users within the local network. Additionally, the web framework provides the opportunity to develop mobile device applications that makes it convenient for users to access information anywhere and anytime. The Vaadin Java Web Framework is a tool kit being used to develop client side web interfaces. Vaadin provides Java developers a short learning curve overhead. Most Java Technologies, including JavaEE and third party packages work well within the Vaadin framework. This paper explores the feasibility of using the Vaadin web framework for developing UI applications for Collider-Accelerator controls at Brookhaven National Laboratory.

INTRODUCTION

"*Vaadin Framework is a Java web application development framework that is designed to make creation and maintenance of high quality web-based user interfaces easy*"[1]. First released in 2009, the Vaadin web application framework has been a fast growing API in terms of popularity among web developers for its rich functionality. The Vaadin framework has an advantage over other web development technologies because it uses the Java programming language which is more familiar to the application development community.

The Vaadin application framework provides two programming models: server side (Java) and client side. The client side framework is backed by the Google Web Toolkit (GWT). Program code is written in Java and resides on the server side. Server side program code helps make web applications more secure. Vaadin has a rich set of UI components. The server side and client side communicates via HTTP (or TCP when websockets are used) protocol and transfers data in JSON. The Vaadin framework supports all major web browsers without additional plugins[2] and works well with major IDEs. This makes the web application coding and debugging easier. In Vaadin, the look and feel of the web application is controlled by CSS themes. This makes web application GUI richer, and more configurable; Vaadin is best used for designing single page web applications which typically work like console level UI applications.

* Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy.

fu@bnl.gov

VAADIN FOR ACCELERATOR CONTROL APPLICATIONS

Controls applications used in accelerator controls systems have many common characteristics, such as:

- They mostly require fast live bidirectional communications with many different systems such as hardware controllers, database servers, file servers, and other legend systems on different platforms (Unix, Linux, Windows etc.);
- Requires fast UI and interactive responsiveness.
- Rich UI for control data visualization for single or multiple GUIs.

These features can be relatively easy to implement with traditional languages such as C++ and Java. As Vaadin uses the Java programming language, server side java JDK (or JavaEE) APIs, third party APIs and jar packages can be used directly. This makes Vaadin a favorable choice when choosing a web application framework.

In a Vaadin web application, the server side programs (written by developers) and client side code (generated by Vaadin from server side code) have a common shared state, which helps enhance UI responsiveness and overall performance. For web based accelerator controls applications, the UI design and GUI layout are relatively easy to implement. It is critical for the client-server bidirectional communication layer to be effectively managed. Vaadin data push features make this kind of bi-directional communication easy to setup.

Vaadin push can be configured with Java annotations: @Push(PushMode, Transport) in program code or with a configuration file. There are three push modes:

- AUTOMATIC (default)
- MANUAL
- DISABLED

and 3 transport methods for communication in the request/response cycles:

- LONG_POLLING
- STREAMING
- WEBSOCKET

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

These push modes and transport methods can be either set in the program or set in a configuration file (e.g. web.xml).

Since STREAMING is very similar to LONG_POLLING and has been deprecated since V7.5.0, this paper just focuses on the long_polling (the "fake" push) and websocket (the real push) methods.

Long polling is a process whereby clients send requests to the server, and the server receive the requests, but holds it for a set period of time. The response happens when new data is available within the time period. Meanwhile, the client keeps the connection open and ready to receive data from the server. In this method, although all requests are initiated from client side, it effectively achieves a real time server push-like bi-directional communication.

Websocket, on the other hand, is a full duplex TCP connection that is independent of the HTTP request/response cycle. Once the connection is established, it is a true real time bi-directional communication.

For accelerator controls applications, both methods work well. But they do have some differences because of the nature with which the communication is established. The long_polling method is easy to setup and is supported by most web browsers. However, there may be a short delay in data transport, while requiring more server resources such as memory. Websockets use dedicated connections between server and clients, use fewer resources and is capable of handling large amounts of client/server connections. Figure 1 diagrams the relationship between web application (client), Vaadin application server and the back end accelerator control system.

VAADIN CONTROL APPLICATION EXAMPLES

Vaadin is primarily designed for single-page web applications which work like desktop applications. The common life cycle for developing Vaadin web applications include, designing and writing web application UI Java code, tuning the look and feel of the UI in CSS, deploying the application to a web container (such as Glassfish server) and testing the application in a web browser. Since the UI design is very similar to strategies used with other languages, the focus of our tests addressed synchronized IO communication with simple GUIs and default CSS settings. We tested Vaadin with 3 different types of controls applications:

1. A single page application with no push (data polling , HTTP).
2. A single page application with manual push in long_polling. (HTTP)
3. A single application with automatic push in websockets. (TCP)

The Glassfish server v4.1 (and v4.0) are used as the application servers.

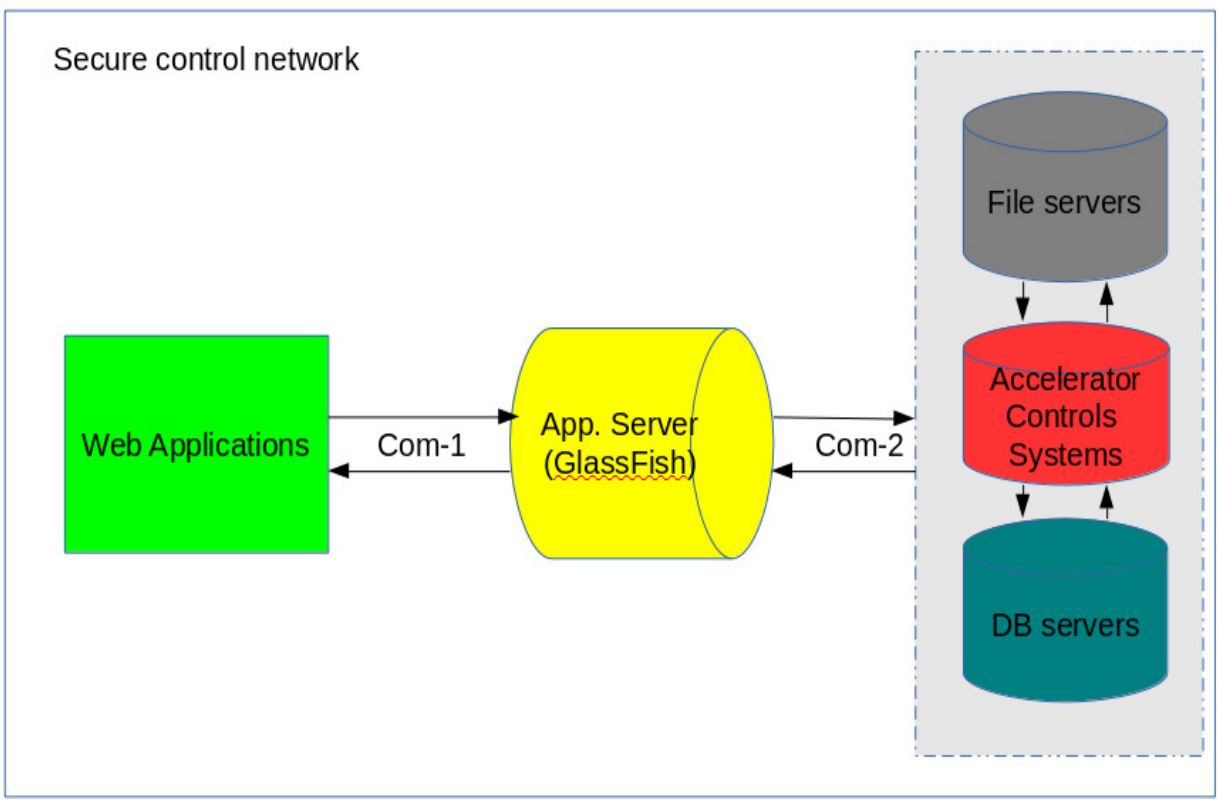


Figure 1: Diagram of Vaadin for accelerator control application

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

In the first case, we converted a C++ based controls application into a Vaadin web application. The application's name is SystemViewer. The basic function of this simple program is to display the current statuses of all monitored control systems based on live data in a back end database, and to highlight any system which may have problems or need System Administrators' attention. This program also displays the detailed system data for any monitored systems, and is capable of launching

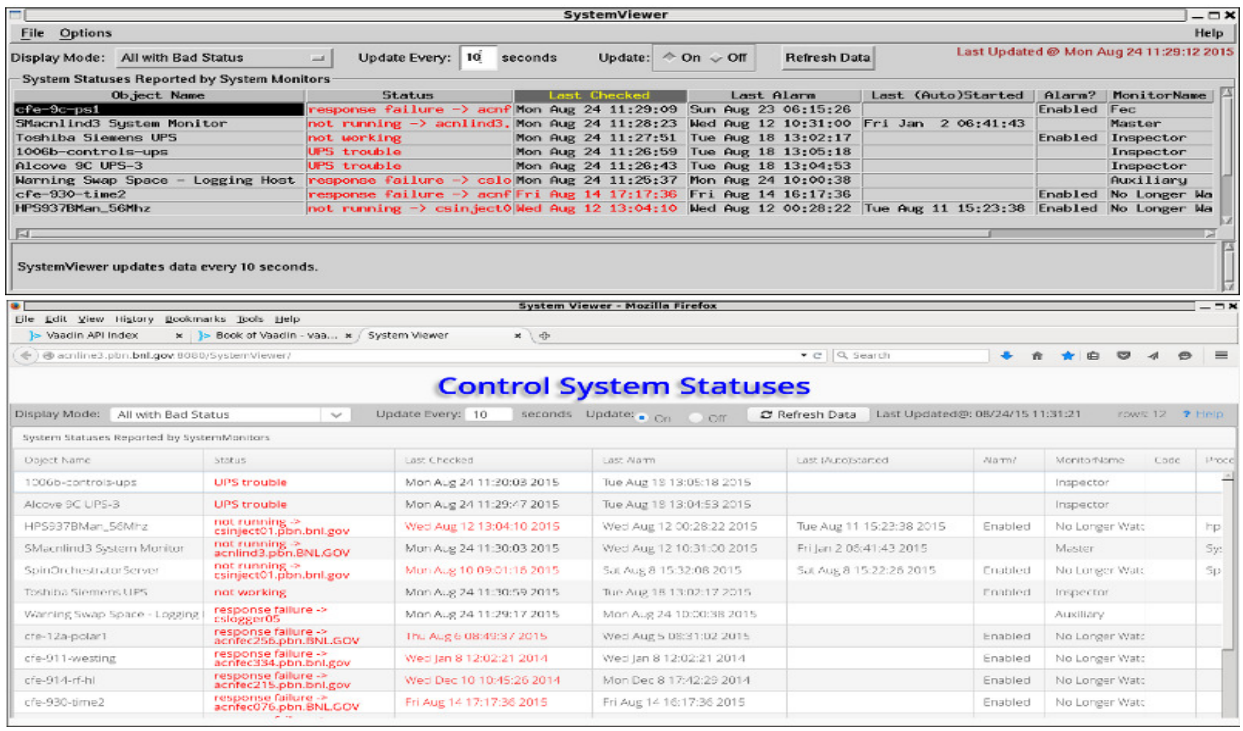


Figure 2: SystemViewer application: C++ (top) vs Vaadin (bottom)

diagnostic tools (other GUI programs) with related context directly from the web application. All features of the C++ version were implemented in the Vaadin version. This makes the program available anywhere within the security network. In this case, all data are periodically polled from the client side and the program works just as fast and as reliable as the C++ version. In fact, the Vaadin version includes more features such as allowing hiding and showing columns in the GUI. Figure 2 shows the GUI of the C++ vs the Vaadin web GUI.

In the second and third cases, we converted a GUI instance of a C++ application called PET (Parameter Editing Tool) into a Vaadin web application and use the data push approach with long_polling and websockets, respectively. In these two cases, the programs connect to control devices and display (or change) the live setting or measurement values of these devices. Two of the devices have data updating at various frequencies ranging from 1Hz to 1000Hz. In both cases, we found that the Vaadin web application works well when the data frequency development. It is recommended that websocket based push is used for long running web applications(>24 hours). Long_polling is supported by most of existing systems as it is an HTTP based "fake" push technology. Websocket is a relatively new technology, and requires new versions of application server software. Websockets are the recommended push technology for web based control applications.

In our Control System, we have successfully developed a DashBoard web application system using the Vaadin technology. It provides a flexible system for quickly setting up web based controls applications with a rich GUI to monitor accelerator operations.

Testing has uncovered some common problems with data push in Vaadin:

- After an application is running for some period of time, a "UIDetachedException" error occurs causing the web application to stop updating IO data. When this happens, re-loading page doesn't always help. The application server requires a restart. This problem may be caused by user session time outs.
- The Web UI seems unable to handle high frequency IO data with data update rates >

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

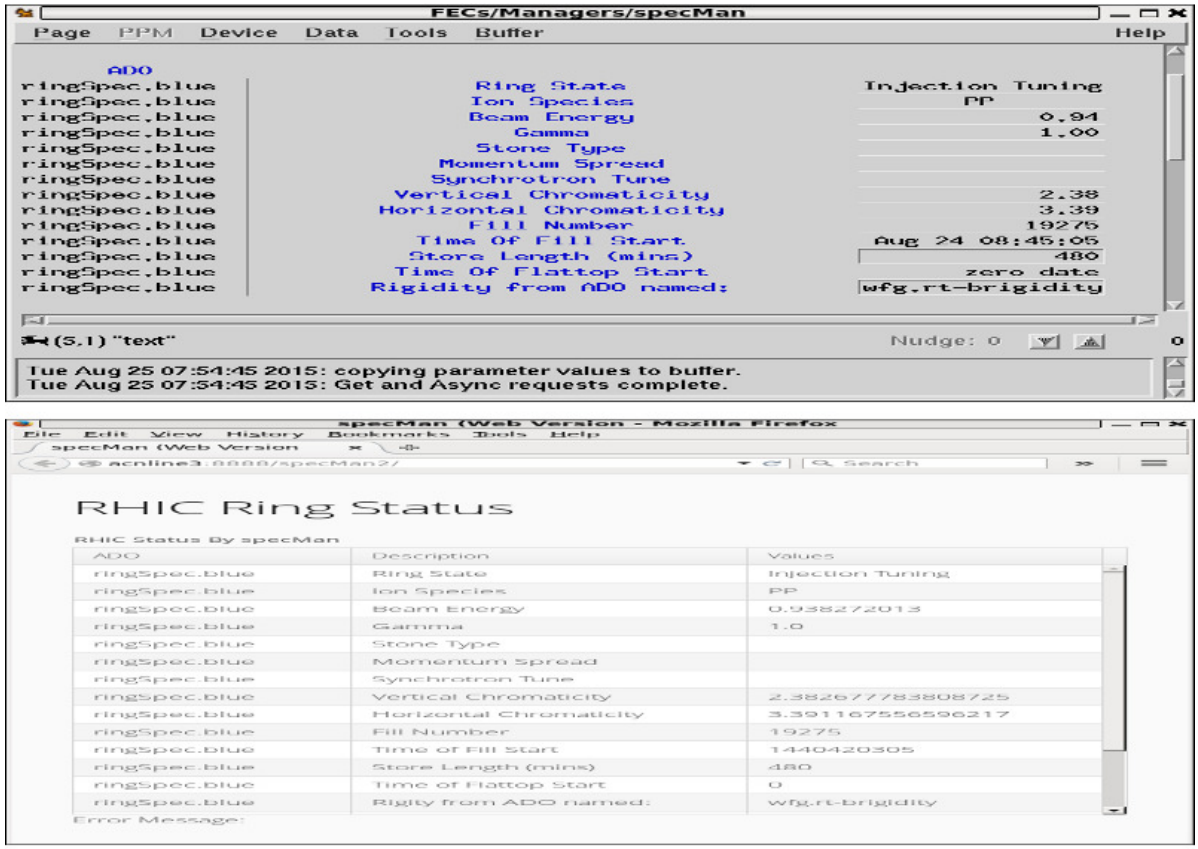


Figure 3: specMan: C++ (top) vs Vaadin (bottom)

~100Hz. Developers need to keep this in mind when deciding if Vaadin is good for the type of target applications which requires high frequency data updates.

- The IO connections between front end application and Vaadin application server, and the IO connections between Vaadin application server and the rest of control system have to be properly controlled and coordinated, otherwise, the control system may be over burdened.

Some of the problems can be resolved by using proper system settings and program logic controls; Others may require software updates. For example, after upgrading from Glassfish 4.0 to v4.1, the UIDetachedException problem disappeared.

It is noticed that, with Vaadin, we can develop web applications without explicit knowledge of HTML and Javascript. However, since all client side web applications are essentially based on HTML, Javascript and CSS, familiarity with HTML, Javascript and CSS will help developers make web applications more flexible.

SUMMARY

The Vaadin web application frame work offers web application developers rich sets of UI components[3], add-ons, Java APIs, and powerful console level application like features. It makes accelerator control web application development simpler for Java developers without the prerequisite of HTML and Javascript knowledge. The key aspects of accelerator control applications include a fast and responsive bi-directional IO connection and UI interactions on web GUIs. The Vaadin web framework does a good job in this regard with server push features that support long_polling or websockets. The testing of accelerator control web applications developed with Vaadin technology shows that, it is easy to convert console level control applications to HTML5 supported web platforms, and the web applications can be just as robust as the OS console level applications. As Vaadin technology and application server technology evolves, this web application framework will become more reliable. These technologies will aid in making web based accelerator control application development easy, powerful and more convenient to end users. The testing work with Vaadin showed some problems that needed attention during application development. Avoiding these pitfalls will help to develop robust and high performance web applications.

REFERENCES

[1] Book of Vaadin: https://vaadin.com/book/
[2] Wikipedia: https://en.wikipedia.org/wiki/Vaadin
[3] Vaadin: http://demo.vaadin.com/sampler/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DEVELOPMENT OF iBEACON BASED EQUIPMENT INVENTORY SYSTEM AT STAR EXPERIMENT

J. Fujita, M. Cherney,
Department of Physics, Creighton University, Omaha, NE 68178, USA

Abstract

An inventory system using iBeacon technology has been developed. Using a specially written iOS app, makes the location of the equipment easier to a workers during the routine access to the experiment. The use of iBeacons and iOS devices allow us to distinguish one equipment rack from another very easily. Combined with 2D barcode, the use of iBeacons may provide better inventory management of the equipment for experiments.

INTRODUCTION

STAR Collaboration (Brookhaven National Laboratory, Upton, NY) composed of 56 institutions from 11 different countries. During an annual data-taking period, nearly a hundred different members in the collaboration will come to participate in taking data each year. During the data-taking period, it sometimes is necessary to perform a hardware reset. While the experts of each subsystem are typically present during the data-taking period, not every expert is available for emergencies, and the reset must be handled by the workers. Not all workers are familiar with the hardware, and they may not be aware of where the hardware equipment is located. Having a method to guide them to the hardware which needs to be reset is be critical.

To accomplish this, we have briefly looked into the possibility of using GIS. In ICALECPS 2005, Larrieu et al. [1] and in ICALEPCS 2007, Yamashita et al [2] presented the potential use of GIS for accelerator site management. Unfortunately, GIS is not always useful in three-dimensional space in close proximity such as a high energy physics experiment detector. The idea was quickly abandoned.

Recently, with the widespread use of mobile phone with Bluetooth Low Energy technology, utilizing iBeacons can be used to locate equipment in the close proximity. We have developed a special iOS app using iBeacons as well as a 2D barcode.

THE SYSTEM

For the iBeacons, we deployed several Raspberry Pi [*3*] computers equipped with USB Bluetooth 4.0 dongles.

SYSTEM OVERVIEW

iBeacon

iBeacon is the technology standard developed by Apple, which allows mobile apps to listen for signals from beacons in the physical world and react accordingly. Essentially, iBeacons behave as lighthouses. Using Bluetooth Low Energy technology, it sends out a small string of information to identify the beacon. This information is received by a specifically written app for the beacon to show the information as well as measure the distance from the beacon to the mobile phone.

iOS Application

In order to effectively use iBeacons, a special app that has the beacon information has to be written. In our case, the equipment rack information is associated with the beacon information. Once the worker enters the proximity of the equipment rack with an iBeacon, the app will notify the worker what equipment are available in that particular rack. This will let the worker know that he/she is standing in front of the correct set of equipment.

QR Code

While iBeacons will help the users to know if they are in the proximity of where they need to be, it will not help them to identify the equipment in question. To deal with that, QR codes are used. The information on the QR code is kept relatively short, allowing us to miniaturize the QR code size, which is necessary to fit it on a VME face plate. The built-in camera on an iOS device functions as a QR code scanner. Figure 1 shows an example use of the QR code.

Figure 1: An example of equipment with the QR code.

This implementation provided both flexibility and the cost effectiveness.

The iBeacons were placed throughout the equipment platform. They are set to transmit slightly different

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

information allowing one to be distinguished from one another. The information that each iBeacon transmits are coded into each location of the platform and equipment in that location. The Figure 2 shows the actual iBeacon setup.

Figure 2: iBeacon hardware setup.

The app is written such that once a worker is are in the proximity of an iBeacon, it will beep and display the synaptic view of that location with all the inventoried equipment in that location. The user can tap each item on the mobile device screen to see more detailed information on the equipment.

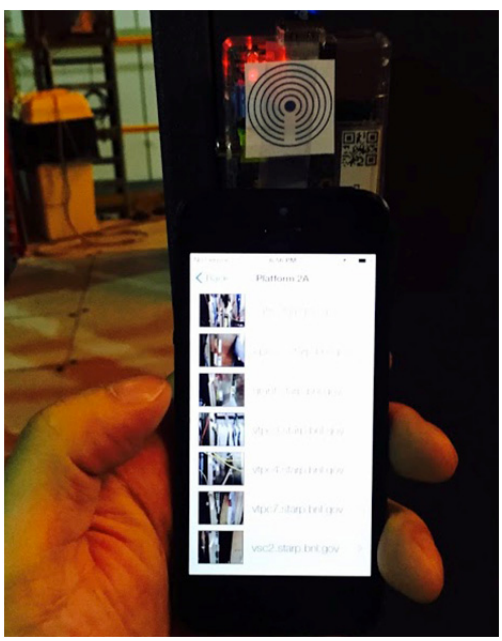


Figure 3: iBeacon setup and the iPhone app.

The app is also equipped with QR code scanner. If a worker scans the QR code attached to each pieces of equipment via built-in camera, the detailed information of the equipment is displayed on the mobile device.

The app was written initially in Beacondo Designer [4]. This toolkit allowed us to write a simple iBeacon aware iOS app very quickly without writing a single line of code. Once the prototype app was developed in Beacondo, then, the project was built for Xcode for tweaking as well as to produce a binary application to evaluation to a real iOS device.

The app was tested on an iPhone 5 as well as an iPhone 6. Although not tested, the app should run on other Bluetooth 4 equipped iOS devices as well. Figure 3 shows the iBeacon and the iPhone app during the performance testing at STAR Experiment. Figure 4 and 5 shows the screen shot of the app.

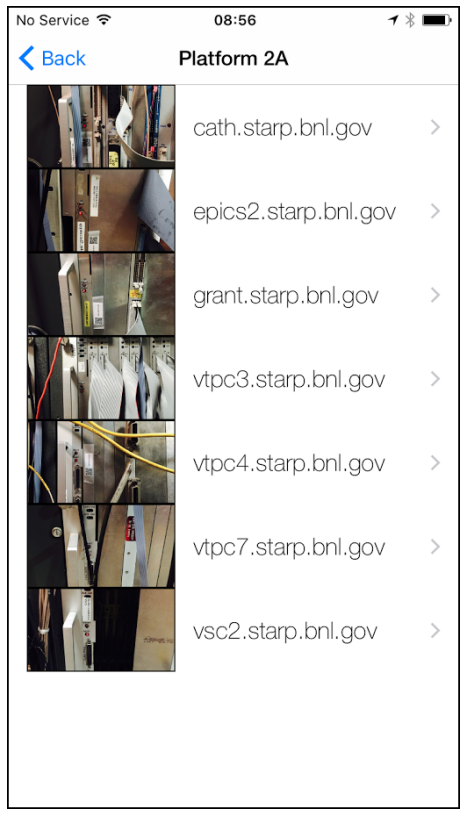


Figure 4: Screenshot of the iBeacon app.

Future Plan

Currently, due to the beacon power issues, the user is required to hold the iOS device fairly close to trigger the beacon information. The beacon power/distance issue must be carefully studied.

The same template generated by Beacondo app is capable of producing Android version of the app, but this has not yet been tested.

Finally, while preparing for the actual test at STAR Experiment, Google announced its "Eddystone" [5] beacon. We have briefly looked into the technology. While we have found some strength in Eddystone technology, it is too early to determine if Eddystone will be widely accepted or not.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

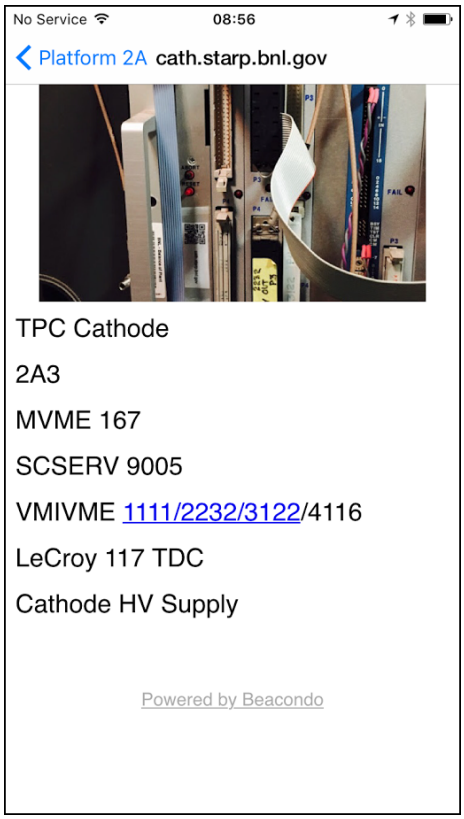


Figure 5: Detailed information on a single VME processor as displayed in the iOS app.

CONCLUSION

The prototype system was tested at STAR Experiment. The limited test performed at STAR Experiment at Brookhaven National Laboratory shows that the system is capable of correctly identifying the equipment nearby. It has been determined that he system should help guiding the workers identify the necessary hardware equipment easily.

ACKNOWLEDGMENT

This work was supported in party by the Office of Science of the United States Department of Energy and the College of Arts & Science of Creighton University.

REFERENCES

[1] T. Larrieu et al., "Evaluating the Potential of Commercial GIS for Accelerator Configuration Management", ICALEPCS 2005, Oct 2005, Geneva.
[2] A. Yamashita et al., "Development of Accelerator Management System with GIS", ICALEPCS 2007, Oct 2007, Knoxville.
[3] Raspberry Pi – Teach, Learn, and Make with Raspberry Pi; https://www.raspberrypi.org/
[4] Beacondo: Build great iBeacon app for retail, restaurant and more; http://www.beacondo.com/
[5] Beacons | Google Developers; Mark up the world using beacons; http://www.beacondo.com/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ADOPTING AND ADAPTING CONTROL SYSTEM STUDIO AT DIAMOND LIGHT SOURCE

M. Furseman, N. Battam, T. Cobb, I. Gillingham, M. T. Heron, G. Knap, W. Rogers,
Diamond Light Source Ltd, Oxfordshire, UK

Abstract

Since commissioning, Diamond Light Source has used the Extensible Display Manager (EDM) to provide a GUI to its EPICS-based control system. As Linux moves away from X-Windows the future of EDM is uncertain, leading to the evaluation of Control System Studio (CS-Studio) as a replacement. Diamond has a user base accustomed to the interface provided by EDM and an infrastructure designed to launch the multiple windows associated with it. CS-Studio has been adapted to provide an interface that is similar to EDM's while keeping the new features of CS-Studio available. This will allow as simple as possible a transition to be made to using CS-Studio as Diamond's user interface to EPICS. It further opens up the possibility of integrating the control system user interface with those in the Eclipse based GDA and DAWN tools which are used for data acquisition and data analysis at Diamond.

INTRODUCTION

Diamond Light Source is a third generation light source, comprising of an injection chain of 100 MeV linac, 3 GeV booster ring and a 561.6 m 3 GeV storage ring [1]. There are currently a total of 33 photon beamlines which are either completed, in construction, or planned. All control system parameters are exposed via the Experimental Physics and Industrial Control System (EPICS) [2], and GUIs for these are realised with the Extensible Display Manager (EDM) [3], which is widely used in the accelerator community for monitoring and controlling live process variables (PVs).

EXISTING INFRASTRUCTURE

At Diamond Light Source we use EDM extensively across the Accelerator complex and Photon Beamlines, which allows us to provide a common interface to many core components such as vacuum systems and motors. Typically a user will access synoptic overviews of a beamline or the machine areas from the Diamond Launcher, an application that resembles a 'start menu' with Diamond specific content. From the synoptic screen a user will access more detailed information by clicking through a series of panels.

All the panels are stored on a read only file system which contains multiple versions of releases for different modules. EDM is started from a script invoked by the launcher which sets environmental variables to point at the correct versions of dependencies for any particular synoptic screen. A new instance of EDM is run for each synoptic allowing them to depend on different modules. In total around 7000 EDM screens are installed at Diamond with many of these being auto-generated.

Not all displays in Diamond are thin clients that interact only with EPICS PVs; some require additional processing to provide the user with interactive information or need to calculate values based on a physics algorithm. It is not plausible to do these processes in EDM and is sometimes difficult to achieve them in an IOC. These are typically implemented in PyQT and interact with EPICS using the Cothread Python library [4]. Developing these GUIs is time consuming and can provide an inconsistent feel to the overall operator interface.

MOTIVATION FOR MOVING TO CONTROL SYSTEM STUDIO

While Diamond has been using EDM since it began operation, the long term prospects for the application are uncertain. As Linux distributions move away from X to Wayland, EDM will lose support for Motif, the widget toolkit it is built on. While it is plausible that EDM could be written to use a more modern toolkit, this transition provides the opportunity to look for a new GUI tool for creating operator interfaces that are based on more modern technologies and provide more features.

Control System Studio [5] (CS-Studio) is written in Java and based on the Eclipse Rich Client Platform (RCP). It provides a modular plugin architecture that allows extensions to be easily contributed in the form of plugins. This architecture has given rise to a number of features that can be bundled with the application to tie together a suite of control system tools into a single interface; an example of one of these plugins is the Data Browser, which can be used simultaneously as a replacement for the 'Striptool' and the 'Archive Viewer'. Eclipse RCP is capable of running on multiple platforms including Linux and Windows. While our control interfaces run on Linux workstations, this does remove a barrier in providing control system information to office users with Windows workstations.

CS-Studio is the choice for many new sites [6–8]. This increases confidence that the application will benefit from community maintenance in the future. Indeed, there has been an average of 7.82 commits a day to the master branch since 2007 [9].

Yet another benefit of moving to CS-Studio is that an automated conversion framework already exists for converting EDM layout files (EDL) to CS-Studio's XML based layout files (OPI). With so many EDL files automated conversion is a necessity if Diamond is to transition its entire GUI infrastructure in a reasonable time scale. The conversion framework is also easily extensible allowing developers to add unimplemented features.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

On photon beamlines, Diamond uses tools developed in house that provide Generic Data Acquisition (GDA) and a Data Analysis Workbench (DAWN) [10]. These tools are developed for external users coming to site to be able to collect and analyse their data without having to understand how to interact with the underlying control system. Both applications provide a GUI which is also implemented with the Eclipse RCP; due to its modular nature it is easy to share components between applications. Using CS-Studio as an interface to the control system further affords the possibility of developing shared software.

CHALLENGES IN SWITCHING

While the decision to use CS-Studio on its merits seems valid, it was necessary to determine if there were any issues that would prevent successful adoption.

Performance

CS-Studio was expected to run more slowly than EDM, which was written to run on computers with significantly fewer resources and less power. To determine if there is a big enough difference in performance to cause issues for users, a series of tests were carried out to asses this and other issues of usability.

In general the time taken to open screens was perceived as acceptable with a few notable exceptions; screens which had been using the 'Symbol' widget in EDM were very slow to open. To work around this Diamond has implemented a custom CS-Studio Symbol widget that has much better performance, but is limited to rendering static rasterized images. Producing those images is part of the post-process conversion process which is described later. In order to achieve this speed CS-Studio does use multiple cores, but this is acceptable given the computers in use at Diamond.

CS-Studio uses considerably more memory than EDM. The Java Virtual Machine (JVM) imposes a fixed upper limit on the available memory when the application is launched. In practice CS-Studio quickly consumes all the memory available to it and then relies on Garbage Collection (GC) to free memory when required. This presents a problem only when the required memory exceeds the allowance at which point CS-Studio will consume a lot of CPU resource before eventually crashing the JVM. This is satisfactorily managed by setting an appropriate value for the memory usage; 2 GB is sufficient for most use cases at Diamond.

A machine operator's shift will often cover 8 hours of continuous GUI use. Due to the complexity and time critical nature of machine control it is not acceptable for the application to crash during this period. Long duration tests have been completed that give us confidence that an instance of CS-Studio can be left running over multiple days without crashing or slowing down. CS-Studio does crash occasionally, however these crashes do not seem to occur more frequently than were experienced with EDM.

Technology and Interface

CS-Studio is based on the Eclipse RCP, which is itself built on a number of other technologies which support the modular plugin architecture such as OSGi. This framework is powerful, but it is also dense and complex making it difficult to understand. Attempting to use the framework to do things outside the Eclipse paradigm is tricky. The Maven [11] based build system also provides those who want to do command line builds of CS-Studio with a steep learning curve.

As SWT is responsible for the widget toolkit in Eclipse RCP, the overall look and feel of the application is defined by the native widgets that it is built upon. Many of the widgets available for users to create their own displays in CS-Studio are implemented in Draw2D and therefore do not share the same appearance as the framework that surrounds them. This has the advantage that displays look very similar on different platforms, but the disadvantage that panels may not be consistent with native applications.

So far, none of the issues described have been sufficient to prevent Diamond's adoption of CS-Studio. However we have many users of EDM whose workflow uses stand-alone windows that are handled by the Linux window manager. CS-Studio presents users with a workbench, a single large window with tiled panes that can contain other windows as shown in Fig. 1. As we are automatically converting our panels they will still retain the EDM layout, which has been designed to be used as a proliferation of many small panels. While it is possible to detach one of these panels in CS-Studio, the new window behaves differently to the EDM windows we have now. There are distinct differences in behaviour with regard to Linux virtual desktops; these are heavily used in the standard workflow of the Diamond operators. The workbench based layout of CS-Studio is also restrictive to those who wish to move panels across multiple physical displays. These limitations, combined with the poor display of many of our existing screens due to their small size, present an issue that needs to be resolved before Diamond could adopt CS-Studio for our operator interfaces.

MODIFICATIONS TO CS-STUDIO

The main purpose behind modifying CS-Studio was to present an interface that would be familiar enough to our users that they could transition from EDM painlessly. To do this the core modification we have made is the ability to display CS-Studio panels in an SWT Shell, which is not integrated into the Eclipse window and so can be manipulated in the way we require. These changes have now been merged into the master branch of CS-Studio and will be available following the 4.2 release [12]. The changes provide users of CS-Studio with the ability to choose a 'Standalone Window' either when right clicking on an OPI file or creating screens which open related displays.

Although not directly a part of Eclipse RCP, it is still possible to tie elements such as the right click menu to the standalone window, allowing it to interact with the workbench window. This is useful if, for instance, a user wishes

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

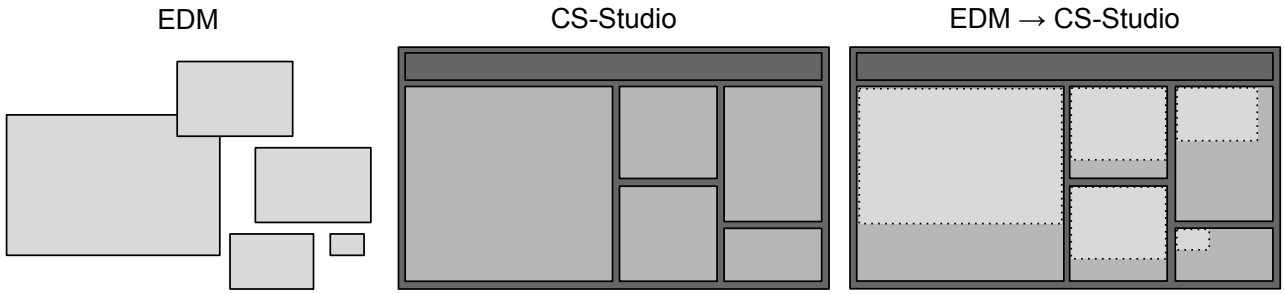


Figure 1: Converted EDM panels do not fit optimally into the default CS-Studio workbench.

to see more detailed information about a process variable that would normally open in the workbench.

There remain a few issues with the standalone window framework as it exists, and we hope to find mutually acceptable solutions to these for the variety of use cases CS-Studio enjoys at different sites. It is possible to open workbench based operator panels and also standalone panels in the same instance of CS-Studio. This could be confusing for a user who may not understand why two windows which appear similar behave in different ways. One possible solution to this is to restrict all open panels to either the view or shell-based method based on a runtime preference. Another solution is to contribute the shell as an optional 'feature' which allows maintainers of CS-Studio at each site to decide if they would like this functionality when building their Eclipse product.

CONVERSION OF EDM SCREENS

Our entire transition process is based around the ability to run existing EDM screens in CS-Studio without significant human intervention. To do this we require a good quality automated conversion from EDL files to the new OPI file format. CS-Studio includes a framework for a widget-by-widget conversion of EDM files, which we have extended to support more widgets and improve accuracy. Some logic contained in EDM screens cannot be reproduced by this method, so we have written post-processing scripts to alter the produced XML files. We have also created two specific EDM compatibility widgets to extend the CS-Studio toolset.

Symbol Widget

The Symbol widget shows a section of embedded panel depending on the value of a PV. Diamond has a site wide temperature monitoring GUI that has 1605 symbol widgets. With EDM the performance of these panels is acceptable, however with CS-Studio it is exceptionally poor. By creating a custom widget able to show sections of a rasterized image we have been able to demonstrate significantly better performance. There are a few edge cases that do not work with this approach, e.g. animated symbols, or embedded symbols that are very different resolutions to their containers, but for the vast majority of symbols this widget works well.

Menu Mux Widget

The Menu Mux widget in EDM changes one or more dynamic macro values in response to a user's control value selection. A new widget, based on the SWT combobox, was created for CS-Studio which offers similar functionality using local PVs. CS-Studio does not support dynamic macro values so these must be substituted with local PVs. The converter is unable to determine which macros referenced in a Menu Mux widget need to be changed to local PVs so they can be used by other controls on the OPI, therefore this update must be performed as part of a whole-file post-process step.

Post-Processing

Due to the difference in behaviour between CS-Studio and EDM not all widgets can be converted directly; their context in the panel determines how they should be converted. Any part of the conversion that has knowledge about the interaction between widgets is not supported by the CS-Studio converter so we have implemented this in a post processing Python script. As well as the Symbol and Menu Mux widgets there are a few other differences in behaviour we address with post processing: CS-Studio requires that a widget with a click action be above all other widgets in order to receive the click event; it also clips widgets that lie outside of a grouping container. We solve these by adding an invisible click widget on the top of the stack, and resizing the grouping container to fit the widgets respectively.

FAST ACQUISITION DATA-BROWSER PLUGIN

Diamond has a fast acquisition (FA) data stream that records data from 255 BPMs at 10 kHz into a 30 TB ring buffer. The data is accessed via a socket with clients request decimated data in bins of either raw, 64 or 16384 decimations [13]. To get this data into CS-Studio a FA data source was added which allows us to query the status of live BPM data by simulating a PV as well as historic archived data. The decimated data is returned with mean, minimum and maximum values, the display of which is supported by Data Browser. Using this setup it is now trivial to plot slow acquisition EPICS data alongside fast acquisition data, something

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

that is very useful when determining how fast beam motion is affected by other parameters.

BEAMLINE USERS

On photon beamlines workstations Diamond hosts a variety of internal and external scientists with a range of experience using controls GUIs. Often EDM's usage paradigm can create a confusing proliferation of windows which are difficult to organise once they cover the available monitor real estate, especially as a beamline typically wants to control the state of multiple items of hardware simultaneously. For this reason we are currently developing a beamline GUI framework that takes advantage of CS-Studio's ability to control the layout of panels in line with the more conventional way of using an Eclipse RCP based application. Shared components between photon beamlines and accelerators such as vacuum and motors will be required to work appropriately in either usage paradigm.

PLANNED CS-STUDIO INFRASTRUCTURE

Any production release used for controls at Diamond must be built from a script without an internet connection, ensuring repeatable builds in the future. It is not practical or even permissible to download all the p2 repositories that the build depends on due to their large size. Instead, this has been achieved by archiving a clean Maven repository used for the build in order to gather the relevant dependencies. This results in a minimal set of dependencies required for an offline build.

All production modules in Diamond are placed in a read only file system with versioned releases. Currently module maintainers manage environmental variables which allow EDM to pick up different versions of EDL files they have dependencies on; each set of dependencies requires a new instance of EDM. Due to the large footprint when running CS-Studio we can not start a new instance for every set of dependencies, fortunately Eclipse provides a method of redirecting paths within the RCP environment to filesystem paths which are called 'share_links'. Part of our modifications to CS-Studio have been allowing additional command line parameters to modify the links in a running instance of CS-Studio, thereby allowing a module owner to keep a list of dependencies which is automatically updated each time a user opens a newly released panel.

INTEGRATION WITH DAWN AND GDA

Running CS-Studio on photon beamline workstations opens a significant avenue of collaboration opportunities with the other software groups at Diamond who are also developing Eclipse RCP applications for use on data acquisition and analysis [14]. At present the Diamond build of CS-Studio contains a version of the graphing widget used in DAWN which has been integrated for use in CS-Studio displays.

ACKNOWLEDGMENTS

The authors would like to thank Friederike Jöhlinger for building the fast archiver plugin as a summer placement project. We would also like to thank the CS-Studio collaboration for their receptiveness to the changes we have made to their application.

REFERENCES

[1] R. P. Walker, "Commissioning and Status of the Diamond Storage Ring", APAC (2007).

[2] M. T. Heron et al., "The Diamond Light Source Control System", EPAC (2006).

[3] J. Sinclair, `http://ics-web.sns.ornl.gov/edm/`.

[4] M. G. Abbott et al., "Diverse Uses of Python at Diamond", PCAPAC (2008).

[5] J. Hatje et al., "Control System Studio (CSS)", ICALEPCS (2007).

[6] F. Arnaud et al., "ITER Contribution to Control System Studio (CSS) Development Effort", ICALEPCS (2013).

[7] M. Giacchini et al., "The Control System of Spes Target: Current Status and Perspectives", ICALEPCS (2009).

[8] T. Satogata et al., "ESS Controls Strategy and the Control Box Concept", PCaPAC (2010).

[9] `https://github.com/ControlSystemStudio/cs-studio/graphs/contributors`

[10] M. Basham et al., J. Synchrotron Rad. **22**, 853-8, (2015).

[11] `https://maven.apache.org/`.

[12] `https://github.com/ControlSystemStudio/cs-studio/pull/1066`

[13] M. G. Abbott, "A New Fast Data Logger and Viewer at Diamond: the FA Archiver", ICALEPCS (2011).

[14] M. W. Gerring, "Open Source Contributions and Using Osgi Bundles at Diamond Light Source", WEB3O01, *these proceedings*, ICALEPCS'2015, Melbourne, Australia (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TOOLS AND PROCEDURES FOR HIGH QUALITY TECHNICAL INFRASTRUCTURE MONITORING REFERENCE DATA AT CERN

R. Martini, M. Braeger, J. Salmon, A. Suwalska CERN, Geneva, Switzerland

Abstract

The monitoring of the technical infrastructure at CERN relies on the quality of the definition of numerous and heterogeneous data sources. In 2006, we introduced the MoDESTI procedure for the Technical Infrastructure Monitoring (TIM) system to promote data quality. The first step in the data integration process is the standardisation of the declaration of the various data points whether these are alarms, equipment statuses or analogue measurement values. Users declare their data points and can follow their requests, monitoring personnel ensure the infrastructure is adapted to the new data, and control room operators check that the data points are defined in a consistent and intelligible way. Furthermore, rigorous validations are carried out on input data to ensure correctness as well as optimal integration with other computer systems at CERN (maintenance management, geographical viewing tools etc.). We are now redesigning the MoDESTI procedure in order to provide an intuitive and streamlined Web based tool for managing data definition, as well as reducing the time between data point integration requests and implementation. Additionally, we are introducing a Class-Device-Property data definition model, a standard in the CERN accelerator sector, for a more flexible use of the TIM data points.

INTRODUCTION

Computerised monitoring of the technical infrastructure at CERN goes back to the early 1980s when a DEC DPD11 was used to display around 2000 alarms onto 2 screens in a dedicated control room. When this system was migrated to a distributed solution based on Unix PCs the decision was taken to hold monitoring definition data on an external database from which the monitoring system would be configured.

This strategy was of great benefit as the monitoring system evolved first to the Technical Data Server – (TDS) in the mid 1990s and finally to the Technical Infrastructure Monitoring (TIM) system [1]. Throughout this time the number of data points grew to reach around 100K data points today, and there is a constant demand to incorporate new monitoring from different services installing new equipment and upgrading existing installations.

For all these monitoring systems the issue of ensuring that data were correctly defined was paramount in determining satisfactory performance. The specification of the monitoring data is the responsibility of the different services whose equipment is being monitored, but the data must be validated by the operators who monitor the alarm screens as well as those responsible for the monitoring system itself.

To ensure that the declared data are correct, complete and follow established standards, the Monitoring Data Entry System for Technical Instructure (MoDESTI) procedure [2] was devised and deployed shortly after the implementation of TIM in 2005.

MoDESTI has evolved over the years to handle the many different scenarios relating to the maintenance of the data that defines TIM; however it is cumbersome to use and difficult to maintain when changes in the monitoring system need to be covered. For this reason a complete reworking of the tool was proposed in 2014 and is now in development.

MODESTI PRINCIPLES

Templates Monitoring data includes descriptive information such as location, equipment concerned, person responsible, information in dealing with an alarm (priority, causes, consequences, actions to take), applications that use the data point (synoptic views, logging, external systems), and configuration data that indicates which acquisition units handle the data points as well as how they are communicated to and from TIM. In this way the reference database contains an overview of all that is monitored by TIM and can be consulted by any authorised user.

Changes will occur during the lifetime of a data point, either through corrections to inaccuracies or due to changes in the way the data point is monitored. Furthermore data points are not monitored indefinitely, when an installation is dismantled, the related data points must be removed from the monitoring system to avoid clutter and possible confusion. These events are also managed by MoDESTI so that the monitoring systems always keep up with the latest operational changes.

Since the aim of MoDESTI was to promote data quality in monitoring, the tool was adapted for the launch of the CERN Safety Alarm Monitoring (CSAM) system. This is a separate system from TIM but CSAM is configured with data defined through the MoDESTI procedure held on the reference database. More recently, the capacity of MoDESTI in enforcing standard definitions and rules relating to alarms was harnessed by CERN's WinCC application for alarms sent to the CCC.

In order to declare data points for monitoring by TIM users must create a data integration request and submit their data in a standard way. Depending on the nature of the data points being declared, specific actions must be taken by the different agents involved in the monitoring process. Alarms must be validated by CCC operators to verify that they are correct, complete and comprehensible

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

when they arrive on screen. Some data points may need cabling to PLCs, those who carry out this work will be notified by the MoDESTI procedure when this is required and will be able to inform the system when this has been carried out. Those responsible for the CSAM system are notified when changes to their alarms are implemented so that they can synchronise their data. Finally, data points are configured onto TIM by the system administrators.

The current implementation of MoDESTI uses CERN's Engineering Data Management System (EDMS) to create data integration requests and then manage the workflow. Data is entered on pre-formatted Excel data sheets where simple checks are carried out by Visual Basic modules before these files are linked to the EDMS request. Data is then loaded onto the reference database where fuller validation checks can be carried out prior to configuration in TIM.

ENTEPRISE LOGIC

Looking back at the last ten years of experience with the TIM monitoring service we can confirm that investing in a reference – configuration database was the correct approach. The complete review of the data model, data validations, integration and configuration processes in 2005 has guaranteed a stable and reliable configuration data source for the clients of our main systems TIM and CSAM.

Data Validation Principles

The reference data checks start with the light validations as mentioned above. More complex and thorough business logic validations with detailed cross checking of the declared data against existing monitoring data are then carried out. Great care is taken to verify that the integrated data do not lead to non-unique point declarations, incomplete, non-standard, or ambiguous definitions. Alarm points must pass additional validations to ensure that once operational the CCC team have access to all details to react efficiently once the alarm is activated.

TIM is tightly coupled with other CERN technical databases: INFOR EAM for equipment maintenance, GEOSIP for locations, LASER for alarm definitions, TIMBER for long term logging, and FOUNDATION for personnel. Verification procedures must ensure that any TIM data referencing the external systems or any TIM processes injecting data in these systems (e.g.: data logging) are validated against the parent source. This is achieved by implementing off-line synchronizations of the lookup data, thus guaranteeing that our operational environment remains independent at all times from the operational constraints of the third-party databases. The same principle is applied to systems which declare data in TIM. The PL/SQL modules that carry out the validations for MoDESTI are also called by external systems in order to protect the TIM configuration data against any unverified modifications. The MoDESTI interface and the workflow are decoupled from the data business logic and share only the data validation APIs. These APIs are exposed as externally accessible PL/SQL procedures. They call the same validations that are used by the standard MoDESTI process thereby guaranteeing that the data injected to TIM are of the same quality as those that follow the standard declaration path.

Having a well-structured and organized database is essential in providing the correct configuration for the monitoring systems. Another, equally important role of the reference database is to serve TIM users with complementary information related to their work. For this reason we provide applications such as Help Alarm and SMILE (Static Monitoring Information Lookup Engine). Since alarm and data tag signals are transmitted with minimal operational information, these web based applications allow an efficient interpretation of alarms as well a full understanding data tag meanings. Authorised CCC operators can also register further operational instructions or update a tag's detailed information contributing to the overall "operational" quality of the TIM data.

Class/Device/Property

The class/device/property data paradigm is very popular in the CERN control environment and in particular in the accelerator sector. For this reason, in 2014, the TIM data model was adapted to handle the class-device-property representation for TIM data [3]. This new data abstraction layer extends the definition of data tags allowing a mapping to the properties defined within devices. Organizing data in the class-device-property model enhances the further usability of the data sets which share common definitions and are logically classified within devices. This allows us to focus on the particular type of device being monitored, and "hides" the underlying tags to a large extent. On the client side, both when creating visualisation symbols or monitoring views, and writing client applications, the user can work exclusively on the device level without needing to know about the underlying tags used interpret the device. Having opened this new data dimension in TIM we foresee a rapidly growing interest in TIM in the coming years [4].

TOOLS AND TECHNOLOGIES

The current implementation of MoDESTI uses CERN's Engineering Data Management System (EDMS) to create data integration requests and then manage the workflow, and pre-formatted Excel data sheets to capture the data

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

which is then loaded onto the reference database prior to configuration in TIM.

Though this approach was quick to implement, it has several serious drawbacks. First, the batch processing nature of the database validations leads to a slow data integration process as users must wait until the cycle is triggered to receive the validation result, and must wait until the next cycle to reattempt their validation. Second, managing different versions of Excel work sheets when changes to TIM or CSAM require the specification of new data point parameters is also time consuming and requires cooperation with users to migrate to the new Excel template. Third, the workflow cannot be easily altered to adapt to new requirements, moreover the workflow cannot be made dependant on the data of a request. Lastly, the current implementation could not take advantage of the improvements in TIM which allowed for the dynamic reconfiguration of its data acquisition modules.

MoDESTI 2

Having reviewed the drawbacks of the existing collection of tools, a new design to realise the MoDESTI principles and requirements was proposed in March 2015. MoDESTI 2 is a web-based collaborative configuration management tool which incorporates the entire workflow of the data integration process from initial data definition through to data retirement in a single web application.

Plugin System

MoDESTI 2 incorporates extensibility as a first-class principle from the beginning. It has a simple yet flexible plugin-based architecture, which allows a new monitoring domain to be easily integrated with the system. Each plugin provides two major components: a mapping like representation of its domain and constraints called a schema, and a specification of the workflow requirements of the domain called a workflow process. The following sections describe these two components.

The MoDESTI Schema

Each domain provides its own schema, which defines how data points of that particular domain must look. The schema specifies the properties that the points of a domain can have. A schema also provides flexible ways of defining constraints and relationships between properties of data points. Schemas are written in the simple, flexible and lightweight JSON format.

The MoDESTI Workflow Process

Workflow processes in MoDESTI are implemented as BPMN (Business Process Model and Notation) files [5]. This is an XML file that specifies the stages the workflow is composed of and the conditions that govern the transitions between those states. The file also includes a graphical representation of the workflow for ease of understanding by non-experts. The Activiti workflow engine [6] is used to provide reliable, robust, transactional workflows for MoDESTI requests. The default built-in workflow begins with initial data input and flows through validation, approval, configuration and testing stages. However, the default workflow is not prescriptive and individual plugins may supply their own custom workflows as required by the domain in question.

Architecture

The architecture of the new MoDESTI system consists of two distinct components: the backend and the frontend. The backend is a standalone application server which exposes a REST API for creating, updating, reading and deleting "requests", and persists them to a database. The frontend is a web-based user interface which allows a user to graphically interact with the backend to craft a MoDESTI request. The following diagram illustrates the high-level structure of the system.

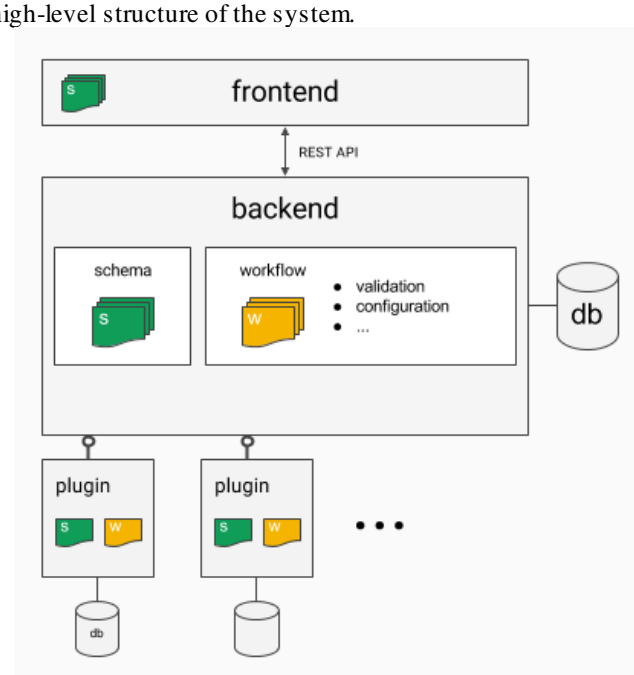


Figure 1: MoDESTI 2 architecture.

The diagram illustrates the plugin-based nature of the system, whereby each plugin provides a schema and a workflow. The backend collects these schemas and workflows, and provides them to the frontend. the frontend shows dynamically generated domain-specific data manipulation controls based on the schema in use.

This design was chosen for a number of reasons, including separation of concerns, maintainability, loose coupling, testability, and the differing skill sets required by frontend and backend developers. Consequently, the frontend contains little to no business logic; it simply provides graphical manipulation capabilities.

Backend

The backend is a Java application which makes use of several libraries for ease of implementation. Most notably, the Spring framework is used for core dependency injection and bootstrapping functionality, as

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

well as several useful abstractions for implementing database persistence functionality.

Frontend

The frontend is written in JavaScript and makes extensive use of the AngularJS framework for structural organisation and coherent functional separation. AngularJS lends itself well to the REST API paradigm, and contains built-in mechanisms for asynchronous backend communication via JSON.

To ease the migration process for existing MoDESTI users, the need for an Excel-like spreadsheet-based interface component for defining a set of data points was chosen, providing a familiar user experience. The "handsontable" library was chosen as the base for this component. The spreadsheet is dynamically generated based on the domain-specific schema.

Advantages over Existing System

MoDESTI 2 is a significant improvement on the existing system for a number of reasons. First, it provides a centralised platform with which users can define new data and search for existing data, without having to switch between several different programs and web pages, and without having to manually manage different versions of files stored locally on the user's computer. The entire user experience is integrated into a single web application; the interface provides instantaneous validation feedback to the user, vastly reducing the time required to identify data entry errors from potentially days to a matter of seconds, and the target systems (e.g. TIM or CSAM) can be configured instantly with new data.

The schema-based approach to modelling domains and their constraints is a powerful one that enables flexibility to changes. Changes to the parameters of a domain can be quickly and easily incorporated by modifying the schema.

Last, as target domains in MoDESTI 2 are integrated as plugins, the core system contains no domain-specific logic or functionality. Each target system specifies the structure of its data points (via schemas), its workflow, and its domain-specific validation and configuration logic. This means that new monitoring systems (or other systems that may benefit from the MoDESTI data integration/configuration management principles) can be quickly and easily integrated into a MoDESTI plugin.

Summary

The frontend world of JavaScript frameworks is fast-changing in contrast to a more slowly evolving Java-based backend world. Having a solid, stable backend positions us well for the future in the eventuality that the frontend might need adapting to a new framework. Separating the two is a good general approach, providing flexibility to change in the long term.

CONCLUSION

The use of a reference database for holding monitoring parameters has been a crucial factor in ensuring a smooth transition between major system upgrades. It facilitates the validation of newly declared data points as well as changes that are often made to monitoring parameters, and lastly provides a view to non specialists of what is monitored by the system. Users can have confidence in TIM insofar as whatever is defined in the system abides by established rules and standards, and has been validated by the agents involved in the monitoring process.

The implementation of the new interface is an important step in making the system more responsive and easier to use, as well as freeing support staff from tedious maintenance tasks. The extensible, plugin-based nature of the new system enables new monitoring systems to be integrated with the well-defined principles that MoDESTI enforces to ensure quality and validity of monitoring data.

REFERENCES

[1] M. Bräger, M. Brightwell, A. Lang, A. Suwalska, "A customizable platform for high-availability monitoring, control and data distribution at CERN", ICALEPCS'11, 2011

[2] MoDESTI home page https://edms.cern.ch/document/707081/1

[3] Devices: a new data structure in C2MON by Mark Brightwell (2013) https://edms.cern.ch/document/1308340/1

[4] Integration, processing, analysis methodologies and tools for ensuring high data quality and rapid data access in the TIM monitoring system (ICALLEPCS 2013) https://edms.cern.ch/document/1311918/2

[5] BPMN 2.0 Specification, Object Management Group (2011) http://www.omg.org/spec/BPMN/2.0/

[6] Activiti BPM Platform http://activiti.org/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ADVANCED MATLAB GUI DEVELOPMENT WITH THE DataGUI LIBRARY

Sascha Meykopff, DESY, Hamburg, Germany

Abstract

On the DESY campus Matlab is a widely used tool for creating complex user interfaces. Although the on-board GUI tools are easy to use and provide quick results, the generated low-level code lacks uniformity and advanced features like automatic verification and conversion of input and output data. These limitations are overcome by the newly developed DataGUI library. The library is based on the model-view-controller software pattern and supports enhanced data handling, undocumented Matlab GUI elements, and configurable resizing of the user interface. An outlook on features of the upcoming release is also presented.

INTRODUCTION

At the European XFEL and the FLASH facility, both located on the DESY campus, most of the operation handling is based on the client-server model with JDDD as the front-end client and DOOCS servers as the back end software [1]. If an operation task needs a more complex graphical user interface (GUI) it's recommended to use Matlab. The GUIDE tool is an interactive method to create a GUI with Matlab. The use of GUIDE provides quick results but extensive code work is necessary to handle the interaction between the GUI elements and the program logic. The effort to implement a GUI complete programmatically without the help of GUIDE is small. The DataGUI library was developed to reduce the coding amount, and to improve the stability of Matlab applications.

ACCESS AND VERIFY PROPERTIES

Matlab GUI elements are designed as objects. The GUI element behavior is defined by object properties. To access element properties the developer has to call a get or set function. Starting with Matlab version R2014b one can use the dot notation to query or set properties [2]. The property names and type format depend on the GUI element type and Matlab version. To write an application which works on different Matlab releases these circumstances must be considered.

```
function edit_field_callback(handle, event)
   edit_input = get(handle, 'STRING');
   edit_input = str2double(edit_input);
   if ~isnan(edit_input)
      display(['input is ' num2str(edit_input)])
   else
      display('input invalid')
   end
```

Figure 1: How to verify and convert text field input in a Matlab callback function.

In a good software design every input variable should be verified. To neglect parameter verification results in unstable software. Currently a Matlab developer is not encouraged to verify the GUI input. Figure 1 shows an example how to verify and convert an edit text field for integer input. This code has to be implemented for every input element.

SHARE DATA AMONG CALLBACKS

The interaction between GUI elements and user code is done by callback functions. If one clicks on a GUI button, Matlab calls an assigned function. Most of the Matlab callbacks carry two input and no output parameters. The first parameter is the handle of the triggered GUI element. The second parameter is defined as 'eventdata' without any value. The developer can add additional input parameters. No return parameters are defined in Matlab callbacks. If the code needs to share data among callbacks the function parameters are not suitable. The Matlab documentation describes four different solutions [3]. These solutions have in common to use a function to get and store the shared data. These functions needs to be called at the start and the end of every callback. This design causes a lot of code duplication and potential errors. A lot of real world software use global variables to store data among callbacks. This is one of the worst solutions because global variables yields in a bad code design.

MODEL-VIEW-CONTROLLER

The Model-view-controller (MVC) is a software pattern which is followed by the most modern GUI toolkits. The pattern is an abstract idea of separating the code into three parts. One part describes and handles the data model of an application. The second one creates the view of the data model. Multiple views of the same data model are allowed for example to view a data table and a plot. The last part covers the program logic. This part updates the data model and modifies the different views. The MVC pattern gives a rough software structure and helps the developer to create a good software design. The software design usually suffers because developing a Matlab GUI is time-consuming. The DataGUI library guides the developer to separate his code which results in a better design.

GLOBAL DATA DESCRIPTION

Matlab software who uses the DataGUI library has unified callbacks. Every callback has a 'data' variable as input and output. The type of the 'data' variable is a structure. One can add, remove, and store fields inside the structure. The storage among callbacks is handled by the

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DataGUI library. The library defines special fields in the data structure called 'registered variables'. These registered variables have additional properties. The library stores the type of the variable. An automatic data conversion ensures the variable type if this variable is modified by a GUI input element. Additionally the library supports constraints. For example a numeric variable has a minimal and a maximal value. The constraints is checked if an input element changes the registered variable. If the input value violates the constraints no modification of the data structure is happen and the invalid input element is marked red. There is no invalid data values in the registered variables. Another additional parameter of a registered variable is a callback definition. If a GUI element has modified a registered variable this function is called. In DataGUI code the callbacks are bind to registered variables not to GUI elements. If different GUI elements modifies the same registered variable the same callback is executed. Figure 2 shows an example for a variable definitions. The definition of the registered variables conform to the view part of the MVC pattern.

```
i = cell( {
   {'data.input_string' 'Test string'              }
   {'data.input_num'    1           [0 10] @new_num}
   {'data.turnoff'      'on'                       }
});
data = data.CALL.addData(data, i);
```

Figure 2: Example code of data definition. Registered variable 'input_string' with value 'Test string'. Variable 'input_num' as numeric input in the range from 0 to 10 and a callback definition 'new_num'. The addData function creates uses the cell array as input and creates all variables.

GUI ELEMENT DEFINITION

The DataGUI library handles the creation of new GUI elements. A new GUI element is defined by the type string of the new element, the new tag name, and the parent tag name. Depending on the type of the new element some parameters are necessary. For example a new label element needs a string as parameter. This string defines the displayed output. If this parameter value is the name of a registered variable the behavior is different. The output of this GUI element is now linked to the registered variable and the element displays it's current value. If the value of the variable is modified inside callback function the GUI element will be changed direct after the callback is finished. If the new GUI element is an edit field the mandatory parameter should be linked to a registered variable. If a user changes the value of the edit field the library read the current value of the GUI element. In the next step the value is converted to expected type format and the constraints is verified. If this verification is successful the new value is stored in the registered variable and the associated callback function is executed. The developer modifies GUI elements by assigning new values to the registered variables. No exception handling because of invalid input parameter is necessary. This will decrease the code size and improves the software quality. In most cases the proper definition of new GUI elements require more parameters. Additional parameters can be defined by property/value pairs. For example the 'visible' property of a GUI element can be set to the value 'off'. But this value also can be the name of a registered variable. In this case the property status depends on the value of the registered variable. A developer can change the visibility status by writing 'on' or 'off' into the linked variable. A small example about the GUI element definition with DataGUI is shown in Figure 3 and the resulting GUI in Figure 4.

```
g = cell( {
    {'figure'   'fig1'   'Name' 'Constant title string'                 }
    {'edittext' 'edit1'  'fig1' 'data.input_string'                      }
    {'edittext' 'edit2'  'fig1' 'data.input_num'                         }
    {'edittext' 'edit3'  'fig1' 'data.input_num' 'visible' 'data.turnoff'}
});
data = data.CALL.addGui(data, g);
```

Figure 3: Defining a GUI with 4 elements. 'Name' and 'visible' are additional parameters.

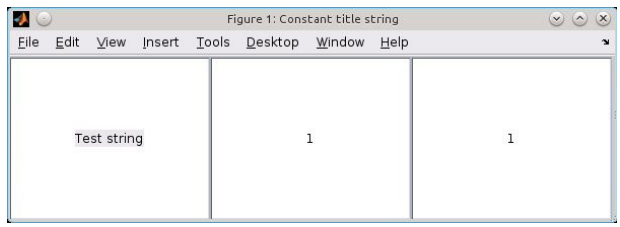


Figure 4: Example GUI with code of Figures 2,3,6.

ADDITIONAL FEATURES OF DATAGUI

The DataGUI library supports all standard Matlab GUI elements. Additionally support for undocumented GUI elements is implemented.

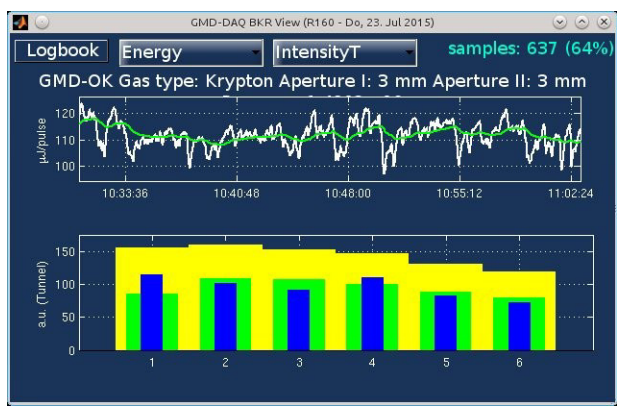


Figure 5: Example of advanced event support. The upper plot can be scale by mouse drag and drop.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

This covers tabulator support for older Matlab versions with a consistent interface. Tree elements with a convenient data interface are included. Dynamic trees obtain their data by a callback function. Also timers are handled by the library. Matlab timers are difficult to handle. The callback function of a timer is able to interrupt a GUI callback. If the timer interacts with the GUI or other data this behavior result in unstable conditions. The timer of the DataGUI library will not run a callback if another callback is executed. A special timer is implemented which repeatedly executes a callback function while a defined condition is true. For example this could be used to emulate multi-threading functionality. A unified callback interface for undocumented callbacks supports the developer to implement a drag and drop interface or to catch other advanced mouse events. An application which uses mouse events to implement a drag and drop control of a plot is shown in Figure 5.

RESIZING OF A MATLAB GUI

The Matlab GUI supports the resizing of elements. A GUI created by GUIDE resizes every element relative to the figure size or not at all. On modern computer systems with huge visible monitor sizes this behavior is unsuitable. For example axes should be resized in a different way than edit fields. To handle the resize of a figure a developer has to set the 'ResizeFcn' or in newer Matlab versions the 'SizeChangedFcn' property. Inside the referred callback function the developer has to calculate and set the complete layout of all visible elements. The DataGUI library shorten this work.

```
layout = {{0 0 0} {'edit1' 'edit2' 'edit3'}};
data = data.CALL.addLayout(data, 'fig1', layout);
```

Figure 6: Example layout definition.

The layout of a GUI is defined by a nested cell array. Each cell describes a horizontal or vertical layout. The cell contains pairs of targets and size definitions. The target is the tag names of GUI element or another nested cell array. The size definition is the size in pixels, a fraction, or the remaining space of the layout. It's possible to use registered variables as a size parameter. This allows the developer to change the layout dynamically. An example layout definition is in Figure 6. Figures 7,8 show a more complex example for a nested structure.

```
h_lay = {{ 0 0 } {'edit1' 'edit2'}};
v_lay = {{ 30 30 };{'edit3' h_lay}};
data = data.CALL.addLayout(data, 'fig1', v_lay);
```

Figure 7: Nested cell array example.

OUTLOOK

While the development of control room applications we improved the library a lot (see Figure 9). But some topics have to be done.

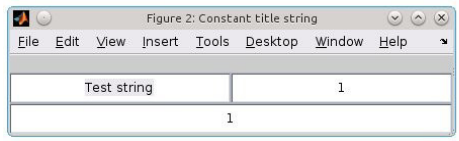


Figure 8: Result of code shown in Figure 7.

One topic is a new object oriented design of the layout to improve the legibility of the code. With the release 2014b of Matlab some fundamental changes happened. This results in a different timing behavior of the applications and the support for special mouse events must be revised. In discussions with different developers a general object oriented approach of the DataGUI library can be considered.

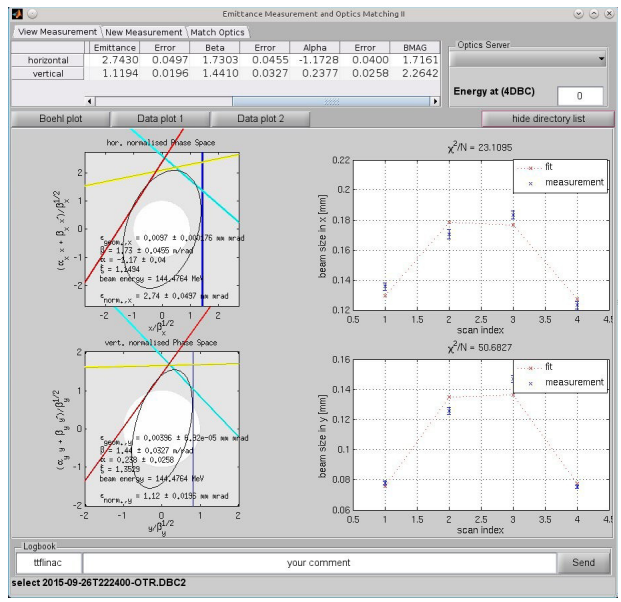


Figure 9: The new emittance measurement and match tool uses the DataGUI library.

CONCLUSION

The DataGUI library enables to build full featured graphical user interfaces with Matlab. The design of the API separates data handing and GUI element handling which will result in a much cleaner code structure. To reduce code duplications and to improve code stability the library takes care about converting parameters and observes constraints. The library supports a wide array of modern GUI elements. The different Matlab API over a long range of Matlab releases is considerd. The DataGUI library supports extensive possibilities to resize the layout. As demonstrated in the DESY control-room the library offers powerful, stable, and easy to handle Matlab software suitable for the daily use by the machine operators.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REFERENCES

[1] R. Bacher et al., "The large scale European XFEL control system: Overview and status of the commissioning", MOA3O02, Proc. ICALEPCS'15, Melbourne, Australia, (2015)

[2] Matlab release notes over a range of versions: https://de.mathworks.com/help/matlab/release-notes.html/.

[3] Matlab manual (share data among callbacks): https://de.mathworks.com/help/matlab/creating_guis/share-data-among-callbacks.html/.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A STRUCTURED APPROACH TO CONTROL SYSTEM GUI DESIGN FOR THE SOLARIS LIGHT SOURCE

V. Juvan, I. Dolinšek, T. Humar, M.Pavleski, Cosylab, Ljubljana, Slovenia
P. P. Goryl, Solaris, Krakow, Poland

Abstract

In the framework of delivering control system services to the Solaris synchrotron light source, Krakow, Poland, Cosylab realized a comprehensive set of controls GUIs, using a structured approach. The goals of using this architecture are threefold. The first is to achieve reliable, predictable and consistent behaviour of the controls software. The second is that it is easy to deploy and maintain through scripting. The third is that it is future-proof by providing extensibility, using dedicated templates. The system is based on a configuration database, populated with devices, device specifics and device groups (clusters of devices performing specific operations). The GUIs are dynamically generated from this configuration. For the synoptic views, TANGO-standard JDraw and its configuration are integrated into the framework. Existing GUIs, written in PyTango can be easily adapted to function as part of this system. The compelling user benefits are high usability and life-time management through controlled upgrade and extension. For new big physics projects this GUI control program offers a customizable solution for any TANGO based control system.

CHALLANGES

When realizing the controls GUIs, we were presented with a series of challenges that we had to overcome.

Firstly, we had to develop a full machine control and state overview program. It was designed as a single entry point program for performing all operations, from every day operator to expert use. Moreover, a broad set of controls GUIs was required for specific operations.

Multiple instances of the control program were foreseen, each intended for a separate control system at the facility.

During the development, we expected to adjust the functionality and features to user feedback, which would be acquired in steps.

Existing tools were not sufficient since they did not provide the desired functionality. With a limited budget, we undertook a development of the Solaris Synchrotron Control Program.

GOALS

The Solaris Synchrotron Control Program had to meet a variety of goals.

We wanted to achieve a predictable and consistent behaviour of the control room software. The control program had to provide a required set of functionalities and features for operation. A transparent and convenient use of the software was requested.

A very important aspect of the control program was that it had to be configuration driven. The goal was to maintain one source of information from which the control program would gather all required information. We would evade hardcoding at all costs. Moreover, we made use of the dynamic GUI generation and generic panels to enable the control program to be adjustable by only modifying its configuration source.

A significant goal was to guarantee easy extensibility. This was achieved by providing dedicated templates and support for integrating external applications in the control program itself.

Last but not least, we wanted to provide easy deployment and maintenance through scripting.

SOFTWARE ELEMENTS

The final deliverable consists of four components, namely:

- Configuration database:
  - Device configuration
  - Device group configuration.
- Control Program.
- Custom GUI library
- GUIrunner

Configuration

The configuration was split into two parts, the device configuration and the device group configuration. The first holds all the information regarding the Tango devices that were to be managed by the control program. The latter holds the information in respect to the groups of devices, to which they may belong to. It heavily relates to the custom GUI library.

The device information incorporates the following parameters:

- Element name
- Type
- L[m], S[m], X[m], Y[m], Z[m],
- Section
- Subsystem
- Managed in CS
- Device Server name
- Device Server instance
- Device class
- Full TANGO device name
- Alias
- Triggered by TTL
- Custom GUI
- Aggregate GUI

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- Description
- Comment

The device group information incorporates the following parameters:

- Group GUI Name
- Previous Group GUIs
- Next Group GUIs
- Additional argument

For more information about the configuration and its parameters please refer to the Solaris Synchrotron Control Program documentation [1].

We should mention here that the same device configuration file was used for the Solaris Import Program (a utility for populating the Tango database). Note that the Solaris Import Program also used a second source of information, regarded as device property configuration. The latter however exceeds the scope of the paper.

ControlProgram

The Control Program is the heart of the controls GUIs. It serves as a main application for accessing all the features and GUIs for operation purposes.

The Control Program parses the configuration and populates the content accordingly. It exposes the devices and device groups and provides access to the control GUIs.

The devices are presented in two different views:

- Device Tree
- Device List

The devices groups are exposed in a separate view:

- Device Groups

Moreover, the following features are implemented:

- Standard/custom device panels
- Custom panel generation
- Device filtering
- Device display management tools
- Device group panels
- Device state monitoring
- Profile management

Device overview

The main functionality of the Control Program is to present all the devices in the system. For every device, it provides a default device panel, exposing all the Tango attributes and command of the device to the user. Additionally, it is possible to use a custom implementation of the device screen for every particular type of the device instead.

The first two views of the Control Program (Device Tree and Device List) expose all the devices in the system. In the first view, user can access devices in a tree, structured in three layers. In the first layer, the devices are split according to the section to which they belong. In the second layer, the devices are split by their subsystems. In the third layer, actual devices are displayed, along with their descriptions. The name, section, subsystem and the description for every device are all read from the device configuration. In the Device List view, the devices are displayed in a list. The order of the devices in the list is read from the configuration, and is foreseen to match the actual order of the hardware in the facility.

For every device, a default device panel or a custom device panel can be opened to access the attributes and commands of the device and to execute required operations.

When working with the first two views, user is presented with additional functions for easier operation:

- Device filtering
- Tools for managing the displayed devices
  - Expand, collapse the tree
  - Select all devices

Attribute overview

Generating a custom panel was supported. Using standard selection techniques, user can select a desired subset of devices in any of the devices views. When requesting to generate a custom, the Control Program will open an additional popup window, prompting the user to select any subset of attributes that the selected devices expose. By confirming the attribute selection, a custom generated panel will appear, presenting all the selected devices with the selected attributes in a structured and flexible form.

Device groups

Devices can be organized into any number of groups. Each device group has a name, a type, any number of preceding device groups, any number of succeeding device groups and a desired additional parameter.

Affiliation of the devices to the device groups, the device group names and device group type are read from the device configuration. The preceding/succeeding instances and the additional parameter are read from the device group configuration.

For every type of the device group, a dedicated GUI must be developed. They are generally developed from a template that provides the layout and the form of the GUI, a background thread for heavy operations, the basic functionality for communication with the Control Program and the functionality for opening the preceding and succeeding instances. The actual content is independent from the template and is developed separately for each device group type. Moreover, the interpretation of the additional parameter for each instance of the device group is also independent from the template.

All device group instances are presented to the user in the third, Device Group view.

When opening a device group GUI, a Control Program looks for the implementation of the GUI for the device

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

group type of the selected device group instance, and retrieves a GUI panel. Doing so, the Control Program passes all the information, required by the GUI implementation to generate the panel, as an input argument. The input argument consist of a device group name, a list of all the devices belonging to the selected device group instance along with their ID and description, and an additional parameter of the selected device group instance. The GUI implementation for the device group type can upon this information build the panel for the selected instance.

Note that the implementation of the device group GUI differs for every device group type. The input to the device group GUI however differs for every device group instance.

A variety of device group panels were developed, intended for different operations:

- Beam diagnostic panel
- Modulator conditioning panel
- Power supply ramping panel
- Vacuum section overview panel
- Synoptic panel for GUN section of linac
- …

State monitoring

The Control Program provides full or partial monitoring of the device states. With this, a full control system state overview is provided. A user can also take advantage of this to monitor a certain subsystem, or to monitor the states of any device subset. The states of the devices are displayed on the left side of the device name, in both device views, and are represented with standard Tango colours. Additionally, a colour legend is displayed when this feature is requested.

Profile management

The Control Program supports the basic functionality for saving and loading a profile. With a profile, we refer to a set of opened GUIs, consisting of Device Panels, Custom Generated Panels and Device Group Panels. When monitoring a commonly used set of panels within the Control Program, it is useful to save a profile, to spare time when trying to open the same set of panels. When a profile is loaded, all the panels, with the same configuration and content are displayed, on the same position on the screen, as they were at the time when a profile was saved.

Custom GUI library

The Custom GUI library is a library that provides the custom device panels and device group panels that can be used within the Control Program. The structure, organization and naming conventions must be insured. For more information refer to the Solaris Synchrotron Control Program documentation [1].

GUIrunner

GUIrunner is an instance manager, an application for running the Control Program. Multiple instances of the control systems were foreseen, therefore multiple configuration sets and multiple custom GUI libraries were developed, each for one control system. The GUIrunner was developed to resolve the problems of feeding the correct configuration set and GUI sources to the Control Program. Moreover, GUIrunner also serves as an update manager. Before opening the Control Program, it checks for any possible updates and applies them beforehand.

ADAPTATION

The adaptation possibilities of the software were crucial in order to be able to adapt it to multiple control systems at Solaris. A configuration based design enabled such a degree of adaptation. Moreover, extendibility of the control program had to be guaranteed. Templates for developing new device group panels were provided to ensure easier future development. Later, a support for integrating external application into the Control Program was introduced.

The software is generic, future-proof and can be adapted to any Tango based facility.

CONCLUSION

Single entry point software for all control room operations was developed. Providing controlled content, a transparent usability was achieved.

The software is easily maintained; any updates or transitions are easily handled.

The required features and functionalities for the operation were developed; nonetheless the software additionally guarantees a great deal of extensibility of features.

A configuration based design was realized, and a required level of extensibility has been reached.

Future development of the software was also foreseen. A new layer of data aggregation was envisioned. Archiving support could be integrated, etc.

REFERENCES

[1] Solaris Synchrotron Control Program: https://github.com/synchrotron-solaris/app-cosylab-controlprogram

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

GUI STYLE GUIDE FOR CONTROL SYSTEM APPLICATIONS AT ESS

Frank Amand, Miroslav Pavleski, Mark Plesko, Cosylab, Ljubljana, Slovenia
Leandro Fernandez, ESS, Lund, Sweden

Abstract

To help developers create consistent-looking control system application GUIs, the European Spallation Source Integrated Control Systems group asked Cosylab to develop a Style Guide (SG) document.

Its purpose is to avoid that GUIs needlessly diverge and make the end-result of all screens combined look harmonious, even if GUIs have been developed over several years by many contributors.

Also it will speed up development, by letting developers start from design patterns, rather than starting "from a blank page".

The document defines a set of basic panel sizes, containing a 960px-style grid for consistent organization of content. It also defines a color scheme and font usage, in-line with the overall ESS corporate communications manual, with the addition of signal colors.

In addition it shows example screens to serve as GUI design patterns for typical screen types such as engineering screens, control applications and synoptic screens.

It concludes by setting rules and recommendations for the usage of automation symbols and display of engineering and physical units.

The document is further complemented by a separate document with Usability Guidelines for Human-Machine interfaces.

CONTROL SCREENS MATCHING ESS BRAND

The GUI colors and fonts that the SG defines were chosen to match he ESS color scheme as defined in the ESS corporate communications manual.

The ESS corporate style has a blue hue as the main color. This is a very suitable color for styling GUI elements. The choice to be in line with the ESS corporate style can enable the control GUIs to look nice and consistent by themselves.

An additional benefit is that this way the GUIs will also be harmonious when used in conjunction with web-based applications that are made in style of (use cascading stylesheet templates similir to) the ESS website itself.

DEFAULT PANEL SIZES AND A GRID

A grid is a proven technique for creating consistent looking GUIs e.g. in web-design.

The style guide defines a "960 pixel grid" in 12 colums. From this basis, 2, 3, 4 and 6 column layouts can be derived and combined.

The style guide also provides patterns for smaller screens/dialogs. For full screen applications it recommends combining smaller grid-based elements.

SIGNALS STAND OUT FROM BACKGROUND

Alarms and signal changes should be very noticable. Their full saturated colors clearly stand out from the *by-design* subdued backgrounds.

The included design patterns illustrate how to use moderate-sized LEDs and other indicators. This keeps a balance of bright colored signals versus the more sober backgrounds and it maintains the overall easthetics. (A Control GUI should by no means a Christmas Tree)

TITILLIUM + OPEN SANS FONTS

The SG defines Titillium [1] for titles (but for titles only). This Open Source font is the ESS corporate font and has nice distinctive features.

But to maximise readability for the body of the GUIs, also at smaller sizes, the more standard-styled Open Sans is chosen. This Google font is a modern, humanist sans-serif. It is designed for good readability at various font sizes. It is NOT just another Arial. It has more font weights (10 in total) then the old normal + italic + bold + bold-italic set that OS standard true type fonts used to provide.

DESIGN PATTERNS AND OTHER REUSE

Various design patterns reuse best practices from other projects on the design of

- Control screens: task oriented screens, either for monitoring and control of a single device or a contral application employing a combination of physical equipment / devices.
- Diagnostics screens, such as a beam current monitor.
- Synoptic views, showing machine overview in a graphical way.

The style guide also defines the way automation symbols are used. In this project we decided to reuase the efforts done on incorporating a symbols database in the EPICS CS Studio tool by the ITER CODAC initiative.

Some effort was spent on making recommendations for implementing efficient and user friendly machine browsing and naviation. This is often a painpoint in control systems of large machines.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TYPE OF GUIDELINES

The document's guidelines are organized in three categories or levels:

- RULE: these must be followed. Exceptions must be treated on a case-by-case basis
- RECOMMENDATION: It is strongly advised to follow these. Less strong then RULE, the application developer can deviate on his own discretion.
- TIP: Design advice that can improve the visual appeal of the GUI.

It is advisable to make this distinction. Stating all guidelines as must-follow rules does not give the software developer the freedom to make good designs for the specific job at hand. At the same time we must make sure important guidelines can be enforced by the quality control team. So they should not be stated in a vague fashion.

SUMMARY AND CONCLUSION

A (20 or so page) style guide is the right first step towards designing for high usability. It is a must to achieve consistent looking GUIs.

A logical next step is a Usability Guidelines Document to also cover dynamic behavior of GUIs. Page count can range for 30-100 pages depending on the project. Cosylab also delivered this at the request of ESS.

The final step in high usability design is applying the principles of goal directed interaction design consistently throughout the software development project.

REFERENCES

Note: for GUI screenshots, see the poster.
[1] http://www.campivisivi.net/titillium/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ALICE MONITORING IN 3-D

O. Pinazza, INFN Sezione di Bologna, Bologna, Italy; A. Augustinus, P. M. Bond, P. Chochula, L. M. Lechman, CERN, Geneva, Switzerland; A. N. Kurepin, INR RAS – Institute for Nuclear Research of the Russian Academy of Sciences, Moscow, Russia;
Jeremi Niedziela, Warsaw University of Technology, Poland, and CERN

Abstract

The ALICE experiment is a complex hardware and software device, monitored and operated with a control system based on WinCC OA. ALICE is composed of 19 detectors and installed in a cavern along the LHC at CERN; each detector is a logical set of modular elements, assembled in a hierarchical model called Finite State Machine. A 3-D model of the ALICE detector has been realized, where all elements of the FSM are represented in their relative location, giving an immediate overview of the status of the detector. For its simplicity, it can be a useful tool for the training of operators. The development is done using WinCC OA integrated with the JCOP fw3DViewer, based on the AliRoot geometry settings. Extraction and conversion of geometry data from AliRoot requires the usage of conversion libraries, which are currently being implemented. A preliminary version of ALICE 3-D is now deployed on the operator panel in the ALICE Run Control Centre. In the next future, the 3-D panel will be available on a big touch screen in the ALICE Visits Centre, providing visitors with the unique experience of navigating the experiment from both inside and out.

INTRODUCTION

With a length of about 26 m, width and height of 16 m, ALICE [1] is one of the biggest and most complex experimental apparatus in the world. It's installed at CERN in Geneva, at access Point2 of the Large Hadron Collider, in an underground hall at a depth of 52 m.

The experimental apparatus is extremely complex, being composed of 19 sub-detectors, each with its own specific technology choices. Monitoring, operating and managing ALICE is accomplished in collaboration by more than 1000 physicists and engineers, from 105 Institutes in 30 different countries.

The ALICE Detector Control System (DCS) [2] is a composite hardware and software structure, connecting and coordinating controls and operations on the sub-detectors and the infrastructure. All software controls are developed and integrated through the SCADA software framework WinCC OA [3], selected by CERN as a standard for all experiment control systems.

Monitoring and controlling ALICE through the DCS is performed by an on-site shifter on a 24/7 basis, whose main duties are to guarantee the safety and the integrity of the detectors and the infrastructure, and smooth running of the experiment. Several on-call experts collaborate with the shifter for the most delicate operations.

The interfaces available for the operators are mainly table based, and all states and actions are encoded using a Finite State Machine (FSM) concept [4]. In this way, complexity is hidden and operators are able to evaluate the status in real time, based on simple colours and keywords.

3-D GRAPHICS FOR WINCC-OA BASED CONTROL SYSTEMS

In spite of the evident simplicity of the tabular panels, a more realistic view of the system with its geometrical characteristics would allow a better understanding and a more faithful evaluation of the overall status of the experiment. For this reason, we decided to develop a new panel and display the sub-detectors and their actual FSM state in an intuitive way (see Figure 1).

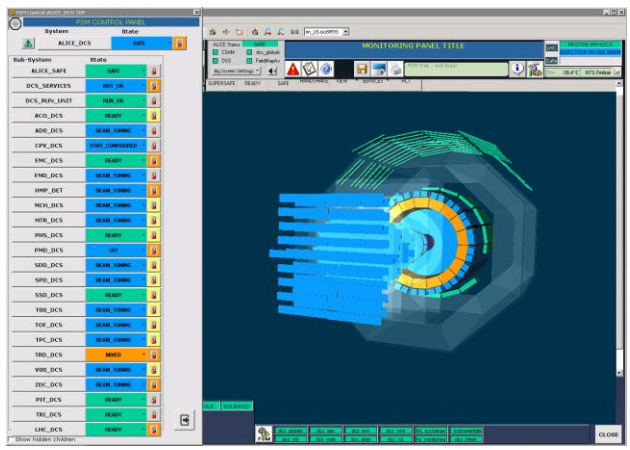


Figure 1: Tabular and 3-D representation of ALICE in one of the operator panels.

All DCS operator panels are integrated in WinCC OA. On top of WinCC, CERN has delivered a set of software tools called JCOP Framework, where common solutions and developments typical to HEP needs are made available. Among these tools, a 3-D viewer widget is available as an extension of the WinCC OA User Interface.

The fw3DViewer Framework tool [5] is based on the GEANT logic [6], and permits building several different shapes using the standard scripting language available in WinCC. The shapes can be simple "Boxes", "Spheres", "Cylinders", "Tubes", "Cones", up to more general "Trapezoids" and "Polyhedra". Each shape is represented by a typical set of geometrical data: a sphere is defined by the position of its centre and the radius; drawing a box

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

requires position, half-lengths in x,y,z, and a 3x3 rotation matrix; a generic trapezoid is represented by at least 13 parameters. Describing the ALICE geometry with this logic requires a combination of several different shapes, and a good knowledge of different detectors' details.

ALICE GEOMETRY AND DATA AVAILABILITY

The ALICE offline group maintains a highly detailed representation of all parts of the ALICE detectors. The parameters are stored in ALICE OCDB; geometry encoding was done by the various experts of all detector groups, and is available in raw code or through the analysis tools based on Root/AliRoot [7].

Since it's maintained centrally, always up-to-date, and is going to include information about detectors upgrade foreseen during LS2, this set of data is of course the official reference for all geometrical based applications. Among these, the ALICE Event Visualisation Environment (AliEve), is a Root based software, able to superimpose the online reconstructed tracks to the geometrical schema (see Figure 2.).

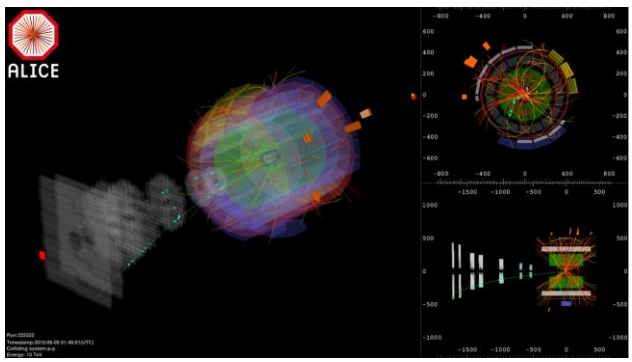


Figure 2: AliEve, the ALICE Event viewer based on ROOT, showing a semi-transparent ALICE frame as the spatial reference for online reconstructed tracks.

Experts and developers from offline, DAQ and DCS would benefit of a joint effort in extracting geometry data from OCDB into flat text files. Using standard Root tools, the conversion of Root objects consists of dumping all fields from the geometry class into an XML file. From here, different applications need suitable libraries to convert XML into their specific formats and standards.

The exported geometry is actually an XML file, describing all shapes and parameters available in the .root file, from the biggest containers (the "BoundingBoxes") down to the smallest internal components, including the description of materials.

During the export phase from Root, the GDML format can be specified. GDML (Geometry Description Markup Language) is an application-independent geometry description format based on XML, and is mainly used to interchange geometry between Root and GEANT4. Since the WinCC 3-D library used by DCS applications is derived from the same GEANT4 geometry description, GDML is the preferred format for the Root geometry extraction. Nonetheless, a conversion library is necessary whose development is currently ongoing.

Other formats were tested, like Collada, which is an open standard XML schema with extension *dae* (Digital Asset Exchange), widely used to interchange data between 3-D applications. Being based on meshes, triangles and vertices, Collada appeared versatile and easy to understand from within WinCC. Unfortunately, every attempt to load Collada geometry in fw3DViewer provoked the crash of the panel, and we focused then back to the XML/GDML format.

FULL AND SIMPLIFIED GEOMETRY

The ALICE detector geometry encoded in the offline OCDB is highly detailed, both in the geometrical description but also in the materials composition. This level of detail is too high for the DCS aim, and for the logical monitoring performed through the FSM. In ALICE case the first level of the FSM hierarchy is normally a geometrical representation of the detector, so that a part (a supermodule, a chamber, a sector, …) can be easily included or excluded from data taking. Operational experience confirms that simple and lightweight panels, showing only the first level of the FSM hierarchy with its included/excluded parts, can guarantee efficient operations, provided that the FSM logics is taking care of masking or propagating relevant events on the children units.

Where available in OCDB, a simplified representation is then extracted (see Figure 3.), as made available directly from detector experts for the sole scope of a geometrical representation, which is called "gentle geometry". In all other cases, the conversion of Root objects is limited to the first level Bounding Boxes, introduced as simple-shaped virtual containers for the real hardware parts.

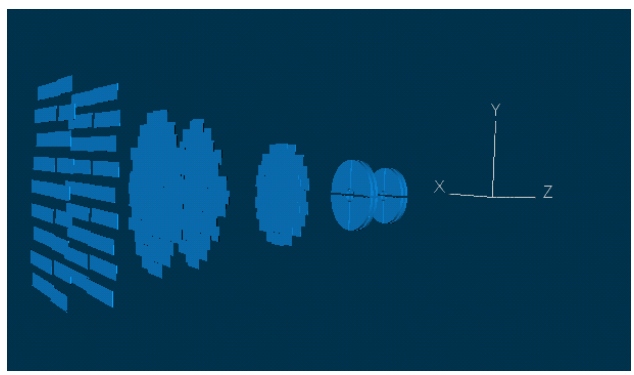


Figure 3: Detail of the ALICE muon arm, using a simplified description of the MCH and MTR detectors' geometry.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

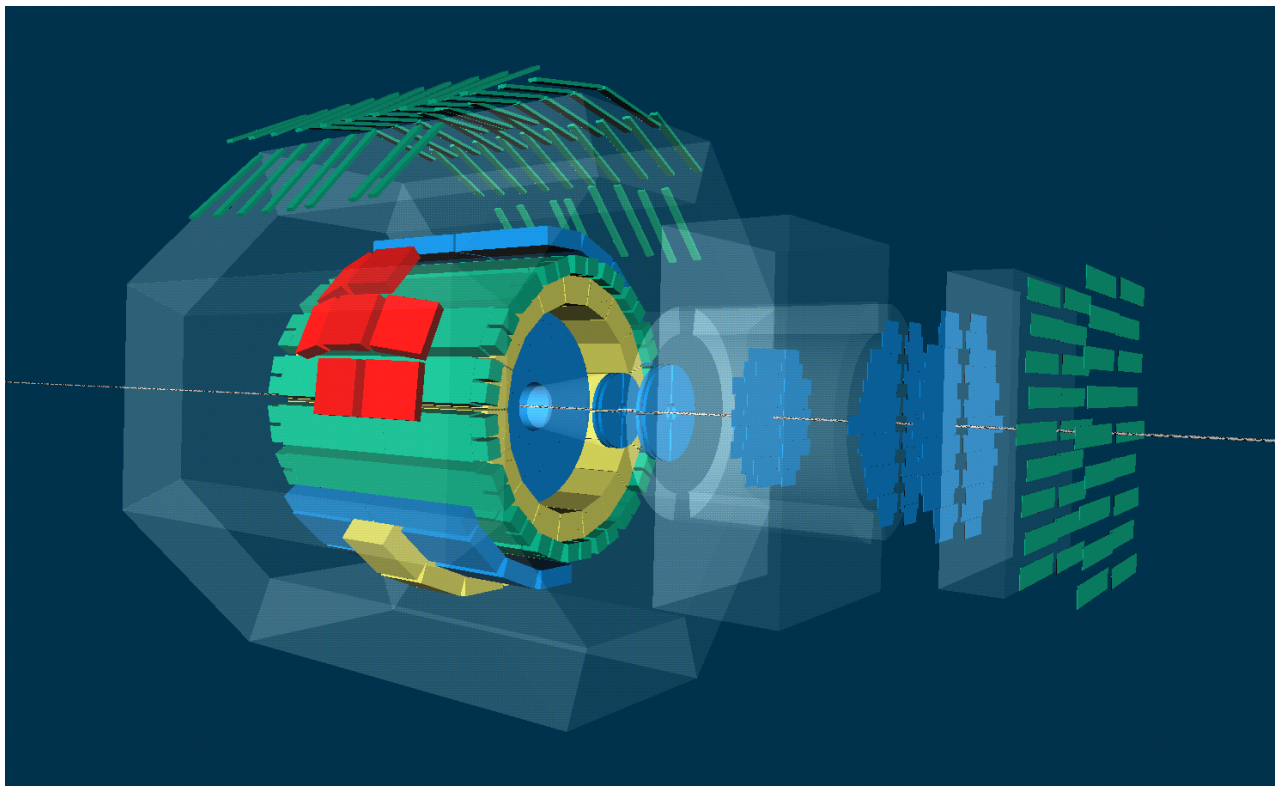


Figure 4: ALICE and its magnets, represented as semi-transparent shapes to allow a better view of the internal detectors. The colours represent the actual FSM state of the detectors.

CONCLUSIONS, THE EXPERIENCE OF MONITORING ALICE IN 3-D

One of the purposes of this work is to develop a general library to convert data extracted from Root objects in simple logical shapes, recognized by WinCC+fw3DViewer.

While proceeding with the realization of the conversion libraries, the ALICE DCS group has realized a first working version of the 3-D DCS panel in native WinCC+fw3DViewer format, in collaboration with detector experts. Even if Root extracted data was not ready to be used, the resulting panel is very accurate in the general shapes. At present, the panel is part of the principal User Interface for DCS shifters, in the ALICE Run Control Centre (see Figure 4).

It is interactive, in the sense that single parts of ALICE (detectors and magnets) can be removed or added, transparency can be set and the full shape can be moved and rotated using the computer mouse. FSM actions are available but not yet activated for the operators.

The panel shall be integrated into the ALICE simulator available in the ALICE DCS lab and used for the training of operators. The aim is to provide a realistic operational view to junior scientists and technologists who haven't had the opportunity for a closer experience with the hardware and the infrastructure.

A preliminary instance of the 3-D model was installed on a large touch screen, that was made available for the public during the OpenDays at CERN in September 2013. We plan to deploy an up-to-date and highly detailed copy on the active multimedia "Magic Windows" which will be installed at P2, in the area devoted to visitors.

REFERENCES

[1] The ALICE Collaboration et al., "The ALICE experiment at the CERN LHC", JINST 3, S08002, 2008.

[2] ALICE Collaboration, Technical Design Report of the Trigger, Data Acquisition, High-Level Trigger and Control System, CERN/LHCC/2003-062.

[3] Simatic WinCC Open Architecture, developed by ETM GmbH, http://www.etm.at/index_e.asp?id=2

[4] FSM: http://indico.cern.ch/event/416600/contribution/s1t4/attachments/857649/1198769/FSM_Presentation.pdf

[5] JCOP Fw3DViewer https://wikis.web.cern.ch/wikis/display/EN/JCOP+Framework+3DViewer

[6] GEANT4 http://geant4.web.cern.ch/geant4/G4UsersDocuments/UsersGuides/ForApplicationDeveloper/html/Detector/geomSolids.html

[7] R. Brun and F. Rademakers, ROOT – An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch/

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

UNIFYING ALL TANGO CONTROL SERVICES IN A CUSTOMIZABLE GRAPHICAL USER INTERFACE

S. Rubio-Manrique, G. Cuní, D. Fernández-Carreiras,
C. Pascual-Izarra, D.Roldán, ALBA Synchrotron, Cerdanyola del Vallés, Spain
E. Al-Dmour, Max-IV, Lund, Sweden

Abstract

TANGO is a distributed Control System with an active community of developers. The community features multiple services like Archiving or Alarms with an heterogeneous mix of technologies and look-and-feels that must be integrated in the final user workflow. The Viewer and Commander Control Application (VACCA) was developed on top of Taurus to provide TANGO with the user experience of a commercial SCADA, keeping the advantages of open source. The Taurus GUI application enables scientists to design their own live applications using drag-and-drop from the widget catalog. The VACCA User Interface provides a template mechanism for synoptic-driven applications and extends the widget catalog to interact with all the components of the control system (Alarms, Archiving, Databases, Hosts Administration). The elements of VACCA are described in this paper, as well as its mechanisms to encapsulate all services in a GUI for an specific subsystem (e.g. Vacuum).

INTRODUCTION

Tango and Taurus

ALBA[1] Synchrotron is a third generation Synchrotron lightsource in Barcelona, Europe, providing synchrotron light since 2012 to users in its 7 beamlines, with 2 more under construction. ALBA institute has been an active member of the Tango Collaboration[2][3] since the very beginning of its design and construction phase. Tango is an open source object-oriented control system, done in collaboration between ESRF and a growing community of institutes and companies developing new device servers and tools using either C++, Python or Java.

Our Human Machine Interfaces to the Tango Control System are developed using Taurus[4][5], a framework for creating both GUIs and command-line tools to interact with scientific[6] and industrial control systems and related data sources. Originally developed in-house at ALBA, Taurus opened its development to the members of the Tango community, becoming popular among the new members of the Tango Collaboration. Some of the causes of this success are the technologies involved: Qt, Python[5] and its integration with SciPy[6], the popular stack of scientific libraries for python.

Tango as SCADA

The concept of a Supervisory Control and Data Acquisition system (SCADA) is largely used in both industrial and scientific worlds to define a control system that remotely controls large installations, using multiple communication channels and providing an application to control large processes distributed on many remote equipments or stations

Tango fits perfectly within SCADA definition, providing the communication channels and software tools needed to manage large particle accelerators and other facilities, along with additional control services such as Archiving, Alarms and User Access. These services are managed by a collection of applications developed either by the core team or by other members of the community: Jive for database configuration, Astor for control host management, JDraw for Synoptics, Mambo for Archiving, ATK/Taurus/QTango for GUI development, Sardana for experiment control as well as several web toolkits and multiple alternatives for Alarm systems[7] like PANIC[8].

Those applications provide a rich functionality and a full-featured control system, but also a diverse collection of look & feels and workflows that may be inconsistent and interfere with user interaction (Fig.1). This paper will address this issue providing a Taurus-based solution.

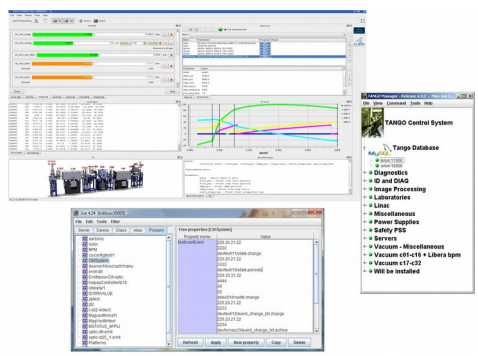


Figure 1: Tango look & feels: Taurus, Jive, Astor

State of the Art, Taurus and CSS

GUI consistency issues are not unique to Tango or to open source control systems, as they may apply to any collaborative project on which each member needs to tune the generic tools to its own context.

At ALBA, these problems were mostly solved once Taurus[5] became the default GUI framework for all our applications, thus creating the opportunity to unify the user interaction with the whole control system. Taurus does not enforce control-system specific conventions but is open instead to include multiple schemas and allows

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

the final user to configure what must be shown and how for all the catalog widgets: attribute and command forms, synoptics, data source trees, plots and trends, image viewers and the drag-and-drop application builder, the TaurusGUI.

But, although Taurus provides a consistent UI framework for all applications developed at ALBA, it still leaves uncovered those control services that are either completely Tango specific, not developed by ALBA or do not fit well in the current model URI's of Taurus.

The same problem has been approached before by the EPICS community, developing a common UI framework to manage all the aspects of the control system. The EPICS-based Control System Studio (CSS)[9][10], provides a consistent GUI that integrates Forms (BOY), Alarms (BEAST), Archiving (DataBrowser), Synoptics and other community tools to deliver a single control application that deals with all control system services.

Both TAURUS and CSS Studio are powerful GUI toolkits with strong communities and full-featured SCADA behind, Tango and Epics respectively. Being both leading projects on its field, this paper presents the implementation of a Taurus-based application as consistent as CSS but with a higher versatility.

IMPLEMENTATION OF VACCA

VACCA was originally developed by the ALBA Control Section (ACS) as the Vacuum Control Application for Accelerators. It was designed as a synoptic based application capable of summarizing the state of hundreds of devices and provide navigation tools to locate and plot any increase in pressure or temperature readings from the vacuum chambers (typical usage and target of commercial SCADA applications).

The development of VACCA has been an iterative process:

- SynopticTree (2007) Java/ATK Based tool based on JDraw and DeviceTree applications (ESRF) and Mambo Archiving browser (Soleil). Used during the commissioning of ALBA linac and transfer lines.
- VACCA/PySynopticTree (2009): First Python release, using the TAU library and integrating Astor/Jive functionality in a single tool as an Alarm toolbar. It did heavy use of composer devices [11] to deal with hundreds of devices.
- VACCA3.0 (2012): First Taurus release, developed in collaboration with the ESRF. Used only for beamlines and backwards incompatible with TAU. Lacking the Astor/Jive/Alarms functionality but providing the versatility of Taurus GUI.
- VACC4 (2015): First release to be deployed on both small and large control systems. All previous features are provided and all widgets become interconnected with all Tango services.

VACCA and Taurus GUI

The current implementation of VACCA (fig. 2) is based on the TaurusGUI framework[5]. This framework goes beyond WYSIWYG and embeds the application design within the application itself, empowering the user to create new panels on running GUI just using drag & drop from the Taurus catalog widget. Initial widgets may be setup in a python module, while additional panels are added on runtime and stored when saving the current layouts as Perspectives: user specific views of the application that record both look & feel and configuration.

The VACCA core is just a TaurusGUI configuration module, that builds a default Perspective with pre-loaded widgets for Synoptic, Archiving Trends, Searches, Properties and a Device Panel. VACCA supports widgets that can be part of the Taurus Catalog, loaded from Vacca or other PyTango modules (like PANIC[8]) or just added by the developer on his custom setup file.

Figure 2: VACCA default perspective

The files that store VACCA customization at facility / system / user levels are:

- vacca.config : the core of vacca python module, it loads and interconnects the most basic widgets.
- vacca.default : overrides certain specific variables of the configuration (e.g. institute logo, devices to be hidden in searches, add/disable alarms widgets) and expands the widget catalog with custom panels.
- [your_config.py] : optional python module passed as command argument, used to do specific setups of synoptics, panel models and filters for tree and attributes widgets (e.g., show only BL vacuum).
- [~/.config/VACCA/.ini] : where user Perspectives, Panels and Plot settings will be saved.

To a Consistent User Interface Catalog

As stated in the introduction, mixing tools coming from different institutes and developer contexts results in common inconsistencies like:

- Displaying different names for same information, e.g. showing or not alias or labels for Tango devices.
- Tools forcing to use the conventions of an specific institute (lower/upper case, spaces/dots in names).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- Not providing always the same behavior for user actions (draggable text, context menus, keyboard shortcuts, usage of user/expert levels).
- Too many solutions to the same problem (e.g., at least 5 different syntax exist to evaluate string formulas in Tango).

As Taurus is currently used for all user Interfaces at ALBA , it solved all these issues except for Tango control services: the Tango Database, Alarms, Archiving, Snapshoting and Starter services.

Those services were originally managed at ALBA using their own tools (Jive, Panic, Mambo, Bensikin, Astor), but gradually replaced by python tools and widgets during the commissioning of ALBA. This was possible thanks to the device server nature of all Tango services, that physically splits each service on client and server side and allows to replace each element independently.

The next step was to embed these service specific widgets in a generic Taurus application, a work done by the VACCA python module. Although these widgets already existed in Taurus (TaurusPropTable) or other ALBA packages (PANIC.AlarmGUI) they have been subclassed to interact between them in a consistent application (Table 1).

Table 1: VACCA Widget Catalog

Widget	Parent	Features
VaccaTree	TaurusDevTree	Provides embedded search bar, device info, Start/Stop of devices, attribute dragging and device selection
VaccaPropTable	TaurusPropTable	Provides device property edition and drag & drop
VaccaSynoptic	taurus.qt.graphic	Provides device selection, graphic element highlighting, context menus, SVG synoptics
VaccaTrend	TaurusTrend	Archiving enabled by default, draggable legends
VaccaPanel	TaurusDevicePanel	Drag & drop of device labels, custom icons for each device class
VaccaBrowser	ArchivingBrowser	Drag & drop to device panel and trends
AlarmGUI	panic.gui	New signals for highlighting synoptics, accept drag & drop on search bar an editor

Other advantages of creating subclasses are:

- To provide a fix interface to each class, so existing Perspectives can handle updates in the upper libraries.
- To keep backwards compatibility with Taurus, PANIC and other dependencies.
- To avoid polluting the upper libraries with ad-hoc features only used within VACCA.
- To enable system-wide configuration, using Class Properties in the Tango Database to customize the appearance of every widget for a given Device Class.

Widget Interaction

All the default widgets loaded at VACCA startup exploit an essential feature of TaurusGUI: the Shared Data Manager (SDM). This mechanism allows the creation of Reader / Writer channels within the application to send information between widgets, enabling translation elements to be inserted in the dialog chain. This feature enables any PyQt widget to be used as device / attribute selector as long as it provides a Qt signal or slot to connect with.

Every widget instantiated from the VACCA catalog (Table 1) is aware of the sources of information available on its own application SDM, subscribing to the available channels and sending its own signals to the crowd. Widgets like Synoptic, Tree or Alarm GUI become enabled as model selectors. VACCA widgets are not only aware of signals at GUI creation time, but also when later added from the catalog into a running application, becoming instantly connected and interactive with the rest of widgets (see Fig.3).

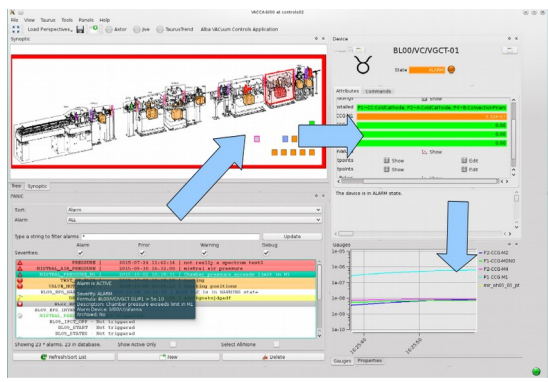


Figure 3: example of widget interactivity, selection of an active alarm will highlight in red the related devices in the Synoptic, where they can be selected for the Device panel or dragged into the Taurus trend.

Drag & Drop

Interaction between widgets that have multiple models have been implemented using drag & drop, a mechanism that is inherent to Taurus and used as the standard method to populate trends and attribute forms. VACCA extends the same approach to any text that matches Taurus models, enabling dragging from/to specific editors like the Properties or Alarms widgets.

When already existing, the drag & drop methods have been extended to use all available Mime types in Taurus: Device, Attribute, Model and PlainText for alias/label. It is used to reflect the multiple ways in which information may be referred-to in taurus, allowing the widgets to translate between the internal control naming

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

(Device/Attribute) and the user-defined naming (Alias/Label) in a way completely transparent to the user.

To provide a more generic solution when the full name of a Device is not known, the Archiving Browser have been added to VACCA to act as a universal search tool for Taurus. Developed to correlate user naming between databases it is both capable of searching on new data, archived data or even devices already deleted. Thanks to drag & drop capabilities it's possible to just drag the known alias into the browser, expand the matching attribute list and then drag the desired attribute to a trend, the alarm editor or wherever is needed.

VACCA and the Control System Studio

Following the classification of CSS Compatibility Levels wrote down by the CSS Studio team at DESY [12], the Widgets in VACCA are integrated at "Advanced" and "Integrated" levels within the framework. Taurus and VACCA cover all together both the “Common Use” and “Selected Use Components” of CSS, thus becoming a full-featured control system navigator.

Performance of a Full-Control System GUI

Performance issues were faced by a GUI like VACCA at the beginning of its design, as it had to be deployed to manage hundred of devices during installation and commissioning, having to deal with dead times and timeouts, as well as avalanches of events in case of sudden incidents that may saturate the whole application.

These issues have been solved using two different approaches. In the case of timeouts, PyStateComposer devices were used to summarize the information on relevant attributes for each accelerator sector (typically 30 to 50 devices). This cached summaries (updated every 3 seconds) allowed to reduce the number of proxies to be opened by the application, and allowed to be resilient against hardware timeout by adding a layer in between and a hook in SDM to translate composed attributes to its real devices.

In case of high event rates coming from the Tango Control System, the solution applied was the event filtering mechanism of Taurus. Event filters can be introduced both at Attribute and Widget level, and executed on either python or PyQt threads.

Both setups, composers and event filtering, allow to deal with most common performance issues. In both cases, however, performance improvements were only needed when managing very large systems and were done just at vacca.default level, with no need to modify Taurus library sources.

CONCLUSION

Open Source control systems may compete in the Scientific world at the same level than renowned commercial SCADAS; leveling or exceeding the performance and features of those private systems.

But, in comparison, the integration of the different subsystems sometimes lack of a unified look & feel due to the multiple teams working on different institutions. This situation is changing with the commitment of new communities of developers that are working on improving both user workflow and performance. VACCA, as a Taurus tool developed originally to manage only vacuum and protection systems, has evolved to get his own role as control system navigator and central hub to interact with all Tango services.

Future integration of additional control systems and features in VACCA will be based in Taurus Schemas and Plugins mechanisms still under development, trying to keep backwards compatibility with existing Perspectives as much as possible.

ACKNOWLEDGEMENT

Most of the early development was done in collaboration with former ALBA members Ramón Suñé and Tiago Coutinho. I must also thank Carlos Falcón, Daniel Roldán (ALBA) and Antonio Milán (MaxIV, Lund) for their work on testing and deploying VACCA on different platforms and institutes. I would like also to thank Andy Götz (ESRF) for its support to the project and Eshraq Al-Dmour (MaxIV) for its role in the evolution of the first releases of VACCA at ALBA.

REFERENCES

[1] ALBA website: http://www.albasynchrotron.es
[2] A.Götz, E.Taurel et al, “TANGO V8 – Another Turbo Charged Major Release”, ICALEPCS'13, San Francisco, USA (2013)
[3] Tango website: http://www.tango-controls.org
[4] Taurus website: http://www.taurus-scada.org
[5] C. Pascual-Izarra et al., “Effortless Creation of Control & Data Acquisition Graphical User Interfaces with Taurus”, ICALEPCS'15, Melbourne, Australia (2015).
[4] Z. Reszela et al. “Sardana – A Python Based Software Package for Building Scientific Scada Applications”, PcaPAC'14, Karlsruhe, Germany (2014)
[5] D. Fernandez-Carreiras et al., “ALBA, a Tango Based Control System in Python”, ICALEPCS'09, Kobe, Japan (2009)
[6] SciPy website, http://www.scipy.org
[7] S.Rubio et al., “Extending Alarm Handling in Tango”, ICALEPCS'11, Grenoble, France (2011)
[8] S. Rubio-Manrique et al., “PANIC, a suite for visualization, logging and notification of incidents”, PCaPAC'14, Karlsruhe, Germany (2014)
[9] J.Hatje et al., "Control System Studio (CSS)", ICALEPCS'07, Knoxville, USA, (2007)
[10] CSS website: http://controlsystemstudio.org/
[11] S.Rubio et al., “Dynamic Attributes and other functional flexibilities of PyTango”, ICALEPCS'09, Kobe, Japan (2009)
[12] CSS at Desy: http://css.desy.de/content/e760/e761

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

A HTML5 WEB INTERFACE FOR JAVA DOOCS DATA DISPLAY

E. Sombrowski, R. Kammering, K. Rehlich, DESY Hamburg, Germany

Abstract

JAVA DOOCS Data Display (JDDD) [1] is the standard tool for developing control system panels for the FLASH facility and European XFEL. The panels are mainly started on DESY campus. For remote monitoring and expert assistance a secure, fast and light-weight access method is required. One possible solution is using HTML5 as transport protocol, because it is available on many common platforms including mobile ones.

For this reason an HTML5 version of JDDD, running in a Tomcat application server, was developed. WebSocket technology is used to transfer the panel image to the browser. In the other direction, mouse events are sent back from the browser to the Tomcat server. Now thousands of existing JDDD panels can be accessed from remote using standard web technology. No special browser plugins are required.

This article discusses the general issues of the web-based interaction with the control system such as security, usability, network traffic and scalability, and presents the WebSocket approach.

INTRODUCTION

JDDD is a common tool for designing and running control system panels at DESY [2,3,4,5,6]. Around 3000 control panels are started each day from DESY offices and in the control room.

Experience shows that many of these panels are needed by experts for remote assistance. The most comfortable access way is using a web interface, which is available on any PC and mobile device all over the world without the necessity of installing any additional software. The requirement for this interface is that no special web version of the panels should be needed to avoid double work. The communication should be fast and light-weight and has to work also in limited bandwidth environments.

A common technology for modern responsive web communication is AJAX (Asynchronous JavaScript and XML). Using AJAX, the web browser polls the server for data on each update. For the transfer of big data packages at high frequency – as it is needed for the JDDD web interface - the creation of a new HTTP connection every time would be a bottleneck. A better solution is a persistent connection like WebSockets.

WEBSOCKET TECHNOLOGY

WebSockets provide a protocol between client and server, which runs over a persistent TCP connection. A visual representation of such a communication is displayed in Fig. 1:

Handshake

Any network communication that uses the WebSocket protocol starts with an opening handshake. The client web browser sends a request to the server to upgrade the HTTP to WebSocket protocol and if the server supports WebSockets, it sends back a response saying: Ok I am upgrading your HTTP to a single TCP socket connection.

Bidirectional Messages

Through this open connection, bi-directional, full-duplex messages can be exchanged at a high frequency (simultaneously or back and forth). In the case of a JDDD WebSocket connection, panel images, which are created on the server side, are sent to the client and mouse events, which are produced in the browser window, are sent back to the server.

Closing the Channel

If the client or server closes the connection, the panel image creation is stopped on the server and all monitors to control systems values are removed.

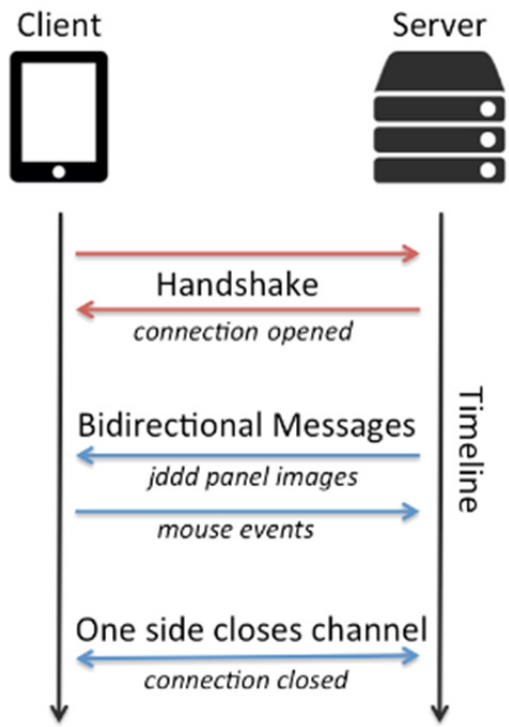


Figure 1: Using WebSockets for client / server communication.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

JDDD WEB INTERFACE ARCHITECTURE

Server Supporting WebSocket Protocol

The WebSocket protocol has been supported since Tomcat 7 and Glassfish 4. Both possibilities were tested and finally the choice was made for Tomcat, because it is more light-weight and the administration is easier.

On the server side one JDDD application is running in the Tomcat web server (see Fig. 2). All panels are started in headless mode in this single JDDD instance. A buffered image is created for each panel with an update rate of currently 0.5 Hz.

Server to Client Communication

To run the JDDD web interface on systems with low bandwidth, the network traffic has to be reduced to a minimum. On the server side this is done by cutting the panel image in 10 times 10 sub-images. Each sub-image is checked for modification and only the parts, which have changed are sent to the client.

Client to Server Communication

In the web browser (client side) all mouse move, click and drag actions are collected using JavaScript. Since the positions differ from browser to browser, they are corrected on the client side before sending them to the server. Now the JDDD web interface emulates the corresponding Java mouse events, so that the panel reacts like on "normal" mouse clicks.

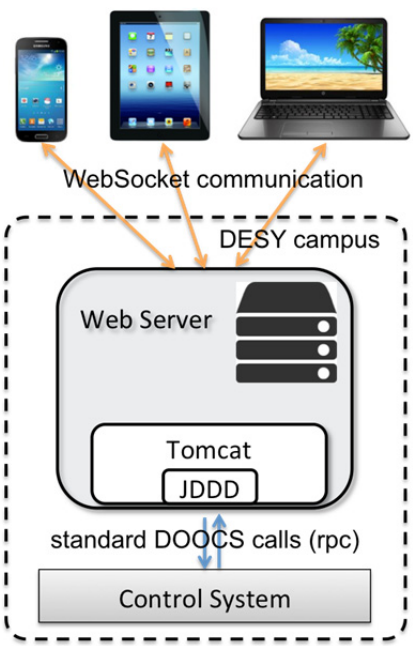


Figure 2: The JDDD web interface architecture using WebSockets for client / server communication.

SECURITY

Before starting the web service a JavaScript login dialog is displayed in the web browser. The user name and password is sent to the server and a DESY Kerberos authentication is executed. The panel is already started in the background and shows up on valid authentication.

The JDDD WebSocket project contains a session manager storing the username, session id and a time stamp for each session. When a new panel is opened on a button click, the session id is inherited from the parent panel and no further authentication is required. The session times out a certain time (currently 1 hour) after the last mouse click. Then all panels of the session are stopped and the session id is removed from the session manager.

During the first test phase, the JDDD web interface has been deployed in a "read only" mode. Changing control system values is not possible.

SCALABILITY

At the moment one Tomcat instance is running on the web server. In this Tomcat server JDDD is started in a single Java Virtual Machine (JVM). The number of panels, which can be operated in one JDDD application, depends on the CPU power, the maximum heap space and on panel complexity. With the existing hardware 15 to 20 standard panels can be started with a proper speed.

Assuming that each user starts approximately 5 control system panels at the same time in his web browser, 3 to 4 people can use the JDDD web interface simultaneously.

To provide the service for a larger number of users, the installation of a Tomcat Cluster is required. A load balancer distributes the incoming requests among the Tomcat servers in the cluster.

CONCLUSION

A proof of concept that the presented technology fulfils our requirements has been done. The web interface has a similar look and feel and functionality as native JDDD (see Fig. 3) and the panel update rate of 0.5 Hz is fast enough for most operational tasks (some restrictions exist: e.g. right mouse clicks cannot be handled, since these are reserved for browser context menus).

The fundamental advantages of the used architecture are:

- Panels have to be designed only once and can be used in standard JDDD as well as in the web interface.
- Changes and improvements in the JDDD source code are instantly available in the web interface.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

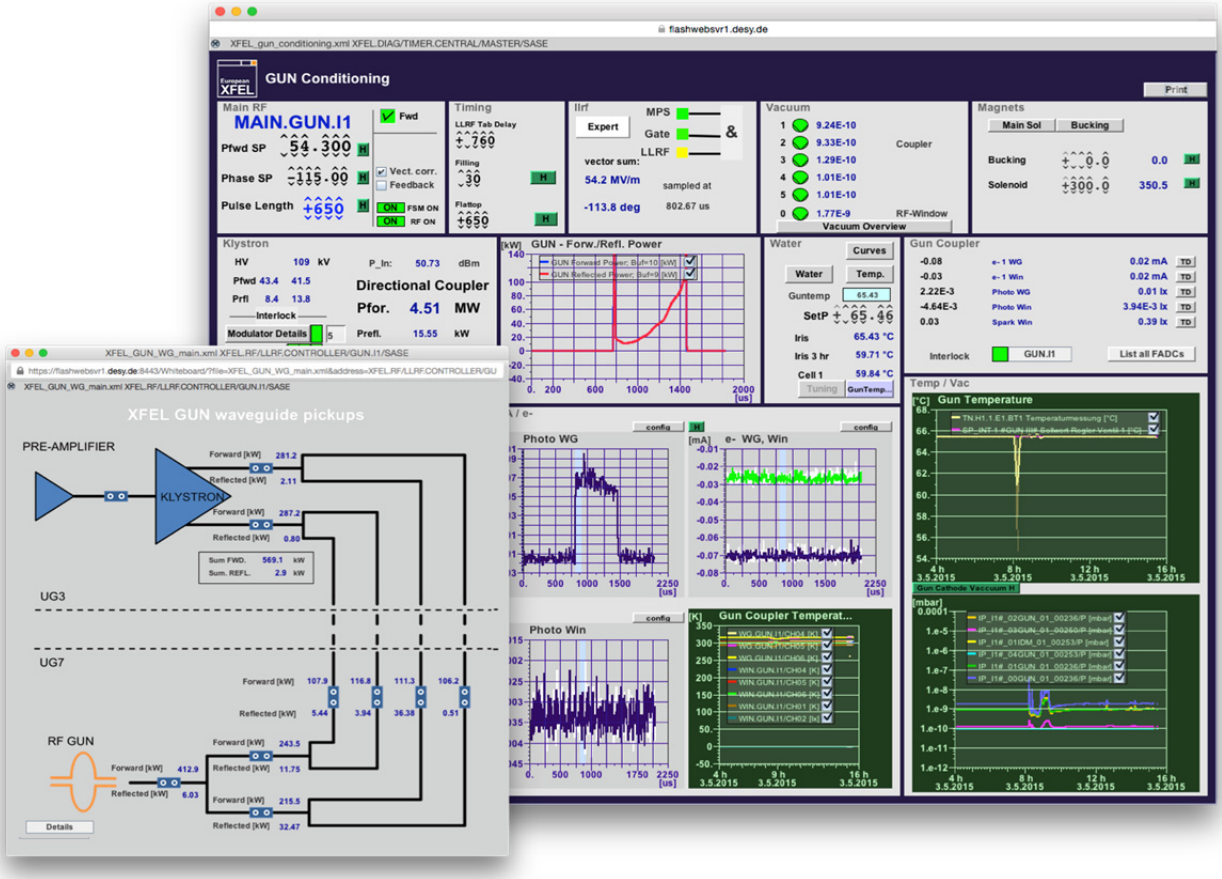


Figure 3: Screenshots of the JDDD web interface running in a Safari web browser (bigger panel) and in a Firefox (smaller panel). All useless browser toolbars are removed and each panel is opened in a separated window. Thus the look and feel and the operation of the panels is equivalent to native JDDD.

OUTLOOK

Some accelerator experts are already working with the current JDDD web interface. To provide the web interface to a larger group of users an improved server setup is required. To support the real operation the "read only" mode has to be changed to full read/write mode, which is planned to be done in the near future.

REFERENCES

[1] jddd website: http://jddd.desy.de

[2] E. Sombrowski. A. Petrosyan, K. Rehlich, W. Schütte, "jddd: a tool for operators and experts to design control system panels", ICALEPCS'13, San Francisco, USA, October 2013.

[3] E. Sombrowski. A. Petrosyan, K. Rehlich, W. Schütte, "jddd, a state-of-the-art solution for control panel development", ICALEPCS'11, Grenoble, France, October 2011.

[4] E. Sombrowski, P. Gessler, J. Meyer, A. Petrosyan, K. Rehlich, "jddd in action", ICALEPCS'09, Kobe, Japan, October 2009.

[5] E. Sombrowski, K. Rehlich, "First Experiences with jddd for Petra Vacuum Controls", PCAPAC'08, Ljubljana, Slovenia, October 2008.

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

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

TIME TRAVEL MADE POSSIBLE AT FERMI BY THE TIME-MACHINE APPLICATION

Giacomo Strangolino, Marco Lonza, Lorenzo Pivetta, Elettra, Trieste, Italy

Abstract

The TANGO archiving system HDB++ continuously stores data over time into the historical database. The new time-machine application, a specialization of the extensively used save/restore framework, allows bringing back sets of control system variables to their values at a precise date and time in the past. Given the desired time stamp t0 and a set of TANGO attributes, the values recorded at the most recent date and time preceding or equalling t0 are fetched from the historical database. The user can examine the list of variables with their values before performing a full or partial restoration of the set. The time-machine seamlessly integrates with the well known save/restore application, sharing many of its characteristics and functionalities, such as the matrix-based subset selection, the live difference view and the simple and effective user interface.

TERMINOLOGY

Context

The context is a set of TANGO attributes gathered together under a name; the context structure is exactly the same as the one beneath the classical save/restore database.

Snapshot

The snapshot is always associated to a specific context and stores the values of the attributes at the very moment the snapshot itself was taken. The relation between a context and the snapshot is represented by the context ID and a snapshot has a unique snapshot ID. Each context can have multiple snapshots taken at different instants, while each snapshot is linked to a unique context. Of course, there is no constraint in the definition of the contexts: distinct ones can have one or more attributes in common.

WORKING PRINCIPLE

The classic save/restore [1] control room application was designed to save a snapshot of a set of TANGO attributes on a database at a given point in time. The snapshots can be later revised and restored on the field. Only previously saved sets of values, identified by an ID, a comment, the date and time the snapshot was taken, can be put back on the equipments. The time machine takes a different approach. Relying on an event-based TANGO historical database archiver, named HDB++ [2], it is able to restore a set of attributes to their values at any date and time in the past. The HDB++ constantly saves TANGO attributes into the database according to a given threshold. Taking advantage of the event-based archiver and the specific extraction library, it is always possible to retrieve the value of a quantity at an arbitrary point in the past. Historical database snapshots can thus be created on the fly at a given date and time.

THE HISTORICAL DATABASE LIBRARY

The HdbExtractor++ framework allows fetching data from a historical database in a simple object oriented fashion. The library currently supports the HDB (the original version of the TANGO archiver) and HDB++ MySql database schemas. Scalar and vector data can be fetched from the database, given a start and stop date and time. A separate module provides a Qt interface to the extraction library and also a dedicated GUI, aimed at displaying scalar and vector data over time. Extraction in the Qt module operates in a separate thread. Vectors are displayed over time by a surface plot, based on the MathGL framework. Errors are notified by NULL values. They typically reflect a read error from the device at the moment of the save process. If the data of a given source is not recorded in a given period, it is possible to fetch values older than the start date. Moreover, data sets pertaining to different curves can be aligned in time, so that a point on the curve is associated to each time stamp in the x axis. Even if the HdbExtractor++ library is tailored to deal with TANGO data (as far as data type, write mode property and data format are concerned), it is not dependent from TANGO specific types and libraries. The Qt module is multi threaded and can be easily integrated in graphical user interfaces [1].

THE GRAPHICAL USER INTERFACE

The Time-Machine graphical user interface is written in Qt and is made up of two sections: the classical save/restore and the new time machine mode. When the application is launched, the classical save/restore mode is selected by default (see Fig. 1). A radio button allows to switch to the time machine view (see Fig. 2). The design of the user interface unifies the approach to the selection of the contexts and the restore process in both modalities. The top left view is a list of the available contexts. Note that the save/restore contexts and time machine ones are defined in distinct databases, so in principle they don't overlap. The right view displays the TANGO attribute names and properties. The information is updated with the read and write values of each quantity as soon as a

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

snapshot is selected or historical data is fetched from the database, according to the active modality. NULL values are highlighted by a red square in the set point column. In order to restore a set of attributes, you must select them

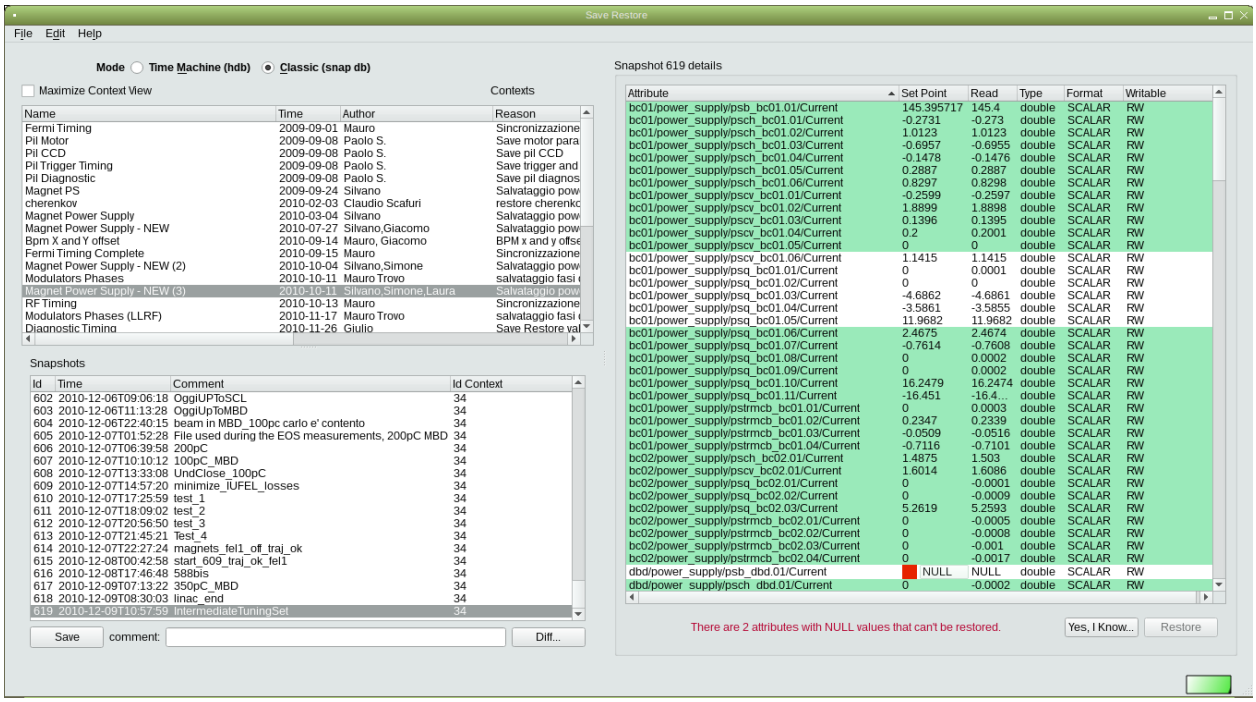


Figure 1: The save/restore classic view.

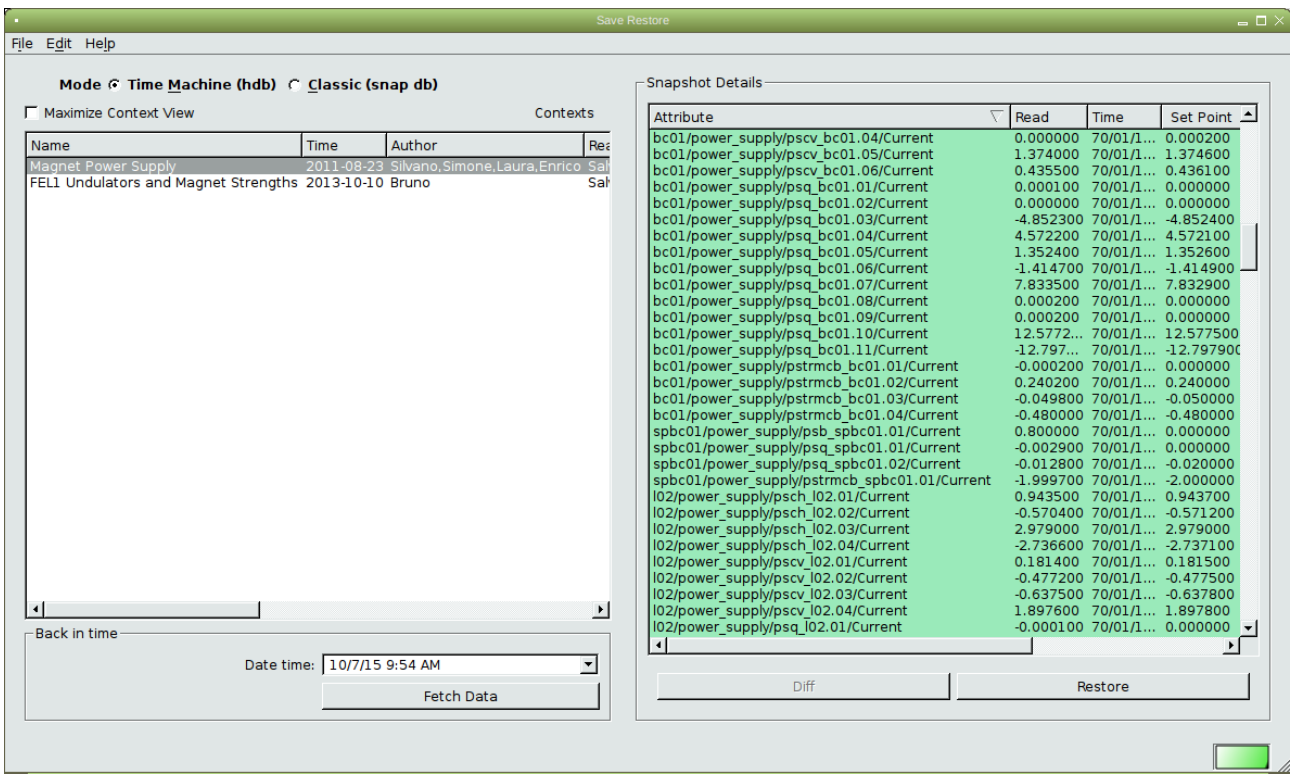


Figure 2: The save/restore application in time machine modality. Note that all the attributes have been selected for restoring and the presence of the *back in time* box.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

with the mouse. All items can be selected at once by right clicking on the view and activating the corresponding option from the menu. If NULL values are present, it is compulsory to click on the "*Yes I know*" push button before restoring (see Fig. 1). In time machine mode (see Fig. 2), the restore process requires the following steps:

- selection of the desired context;
- fetch data from the historical database, specifying a date and time in the past (see Fig. 2, bottom left box) and clicking on the "*Fetch Data*" button;
- select the attributes to restore and restore them.

The save (in the save/restore mode) and the restore processes are run on a separate thread so that the graphical interface remains responsive. A progress bar appears at the bottom of the panel to indicate the advancement of the process. For further information about the classic save/restore application, please refer to [1].

The "diff" View

When save/restore mode is enabled, selecting two or more snapshots of the same context makes it possible to view the differences between the snapshots and the current values read from the field (see Fig. 3).

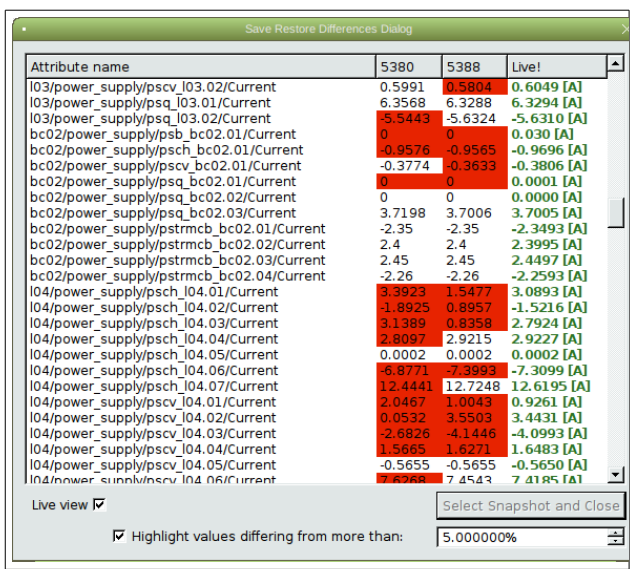


Figure 3: The *diff* view

The Matrix Selection

Alongside the classic item selection mode in the attribute selection list, a matrix selection is available. The selection groups and the different kinds of devices are defined into the database itself and reflect the sections in which the machine is organized. The actual matrix structure is based on the plant structure and the operational needs of FERMI. As shown in Fig.4, toggling an element of the matrix allows the selection (de-selection) of the items of the same type belonging to the corresponding section of the machine.

Helper Application

Right clicking on an item of the attribute list view gives the user the opportunity to launch a linked helper application. Helper applications are defined into the TANGO database and are fully equivalent to helper applications used by the QTango framework [3].

CONCLUSION

The combination of the classical snapshot-based save/restore procedure with the new historical database extraction opens new possibilities to restore the state of a complex machine, bringing it back to a previous condition. In particular, the time machine has been introduced in the FERMI control room to repair a situation after an accident or an unforeseen occurrence, such as a device failure, a thunderstorm and so forth, bringing back the system to a known healthy working state at a given time in the past.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

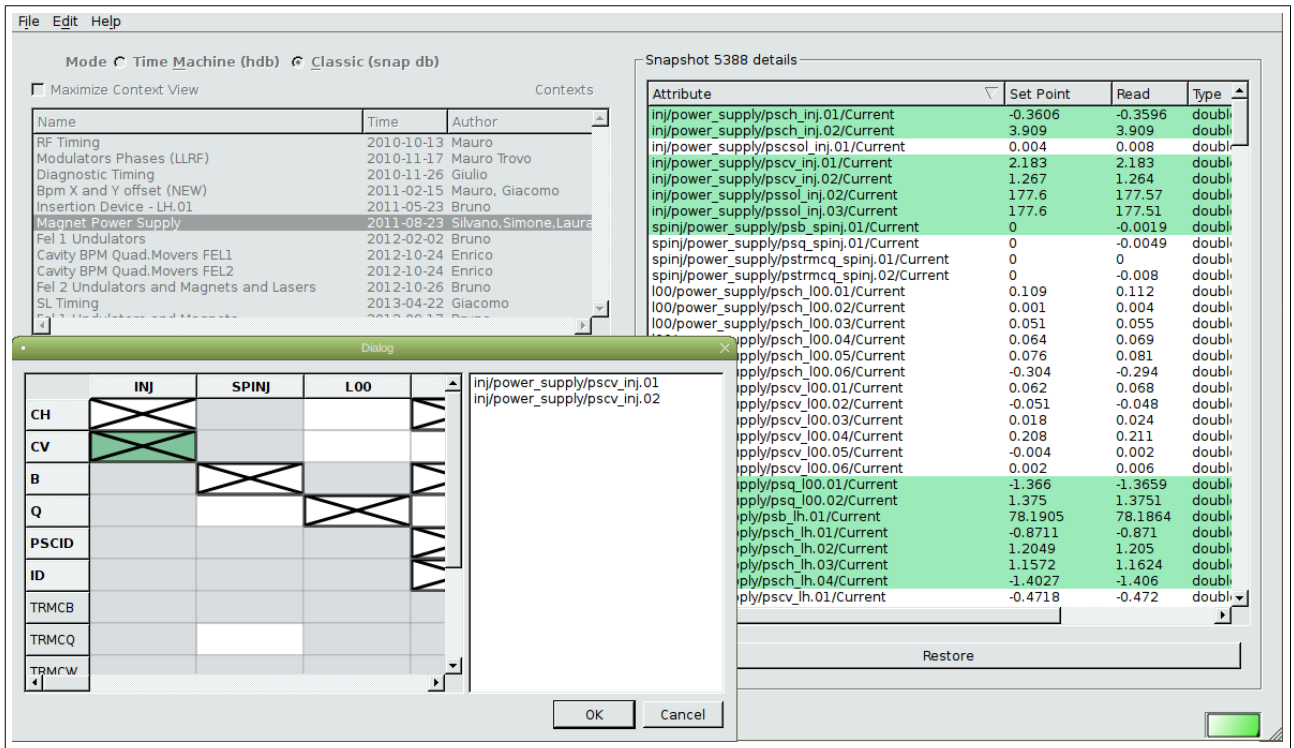


Figure 4: Selection matrix. Toggling an element of the table selects or deselects items on the attribute list.

REFERENCES

[1] G. Strangolino et al., "Control room graphical applications for the ELETTRA new injector", WEP015, PCaPAC08, Ljubljana, Slovenia, (2008)

[2] L. Pivetta et al., "HDB++: a new archiving system for TANGO", ICALEPCS'15, Melbourne, Australia, (2015)

[3] G. Strangolino et al., "QTango: a library for easy TANGO based GUIs development", THP096, ICALEPCS'09, Kobe, Japan, (2009)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Karabo-GUI: THE MULTI-PURPOSE GRAPHICAL FRONT-END FOR THE Karabo FRAMEWORK

Burkhard Heisen, Martin Teichmann, Kerstin Weger, John Wiggins,
European XFEL, Hamburg, Germany

Abstract

The Karabo GUI is a generic graphical user interface (GUI) which is currently developed at the European XFEL GmbH. It allows the complete management of the Karabo distributed control and data acquisition system. Remote applications (devices) can be instantiated, operated and terminated. Devices are listed in a live navigation view and from the self-description inherent to every device a default configuration panel is generated. The user may combine interrelated components into one project. Such a project includes persisted device configurations, custom control panels and macros. Expert panels can be built by intermixing static graphical elements with dynamic widgets connected to parameters of the distributed system. The same panel can also be used to graphically configure and execute data analysis workflows. Other features include an embedded IPython scripting console, logging, notification and alarm handling. The GUI is user-centric and will restrict display or editing capability according to the user's role and the current device state. The GUI is based on PyQt technology and acts as a thin network client to a central Karabo GUI-Server.

THE KARABO DISTRIBUTED CONTROL AND DATA ACQUISITION SYSTEM

Karabo is the control and data acquisition system which will be used on all beamlines of the European XFEL to control the equipment, acquire data and process it [1]. It is a centralized system where all communication is done via a central broker, except for high-bandwidth data streams which are transported via dedicated point-to-point connections.

A Karabo system is a collection of *devices*. Those devices can serve many purposes, they might be a driver for a particular hardware, a composite device that controls several other devices if some coordination between different hardware is necessary, a data processing device that may be just one of hundreds of equal ones in a server farm processing data, or a device saving raw or processed data to disk, not to mention many service devices which keep Karabo running.

There are many different approaches of a GUI for a control system. Some use an already existing development enviroment and extend them to the needs of developing GUIs (e.g. GDA [2], CSS [3]). Others developed stand-alone graphical editors (e.g. JDDD [4], Taurus [5]). Karabo has one fully integrated GUI, which is not only a graphical editor but can be used for all control and data acquisition tasks.

PROJECTS

A typical user of Karabo is not interested in the entirety of the system, rather a specific task to work on. Those tasks are often overlapping, as an example a vacuum technician may want to interact with the same components of an apparatus a scientist works on.

All information necessary for a task can be bundled into a *project*. These are all the devices needed and the configuration with which they should be started, or into which they should be reconfigured to perform the desired tasks. Graphical visualizations of tasks can be added as *scenes* to the project. They are used to show and edit the configuration of devices and the connections between them. Repetitive tasks reoccuring for users of a project may be programmed as macros. If taking some data is the purpose of a project, the data to be taken can be stored in the project as *monitors*. The project itself, however, does not contain the data, which is written to disk independently.

The projects are generally stored on a central server. This way everything needed for a task is persisted, and enables users to use their projects on different computers, and makes the administration of Karabo installations easier, as projects can be archived much simpler.

The project is the core concept of the Karabo GUI, and most of the rest of this paper is a description of its components. Figure 1 shows a screenshot of a running Karabo GUI with all components.

LIVE NAVIGATION AND CONFIGURATION

A Karabo device is run by a Karabo *server*, which is software running on a computer. The device code to be run is installed as plugins into those servers. The installation and running of those servers is beyond the scope of Karabo, but it is typically done with common server administration software.

In the GUI, such an installation is shown as a tree, for each computer the available servers are shown, and the available plugins for each server. A user can thus see the entire Karbo installation. From the GUI, the devices can be instantiated from their *device class*, the code to be run for a device.

Before a device is even instantiated, its parameters are already known to the device server and communicated to the GUI. Thus a user can configure a device interactively by filling out a form which is automatically generated from the description of the device class. Those configurations can be stored in the project. As the full live navigation tree of a Karabo installation can grow large and confusing, those stored configuration give a good overview of the devices needed in a particular project.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

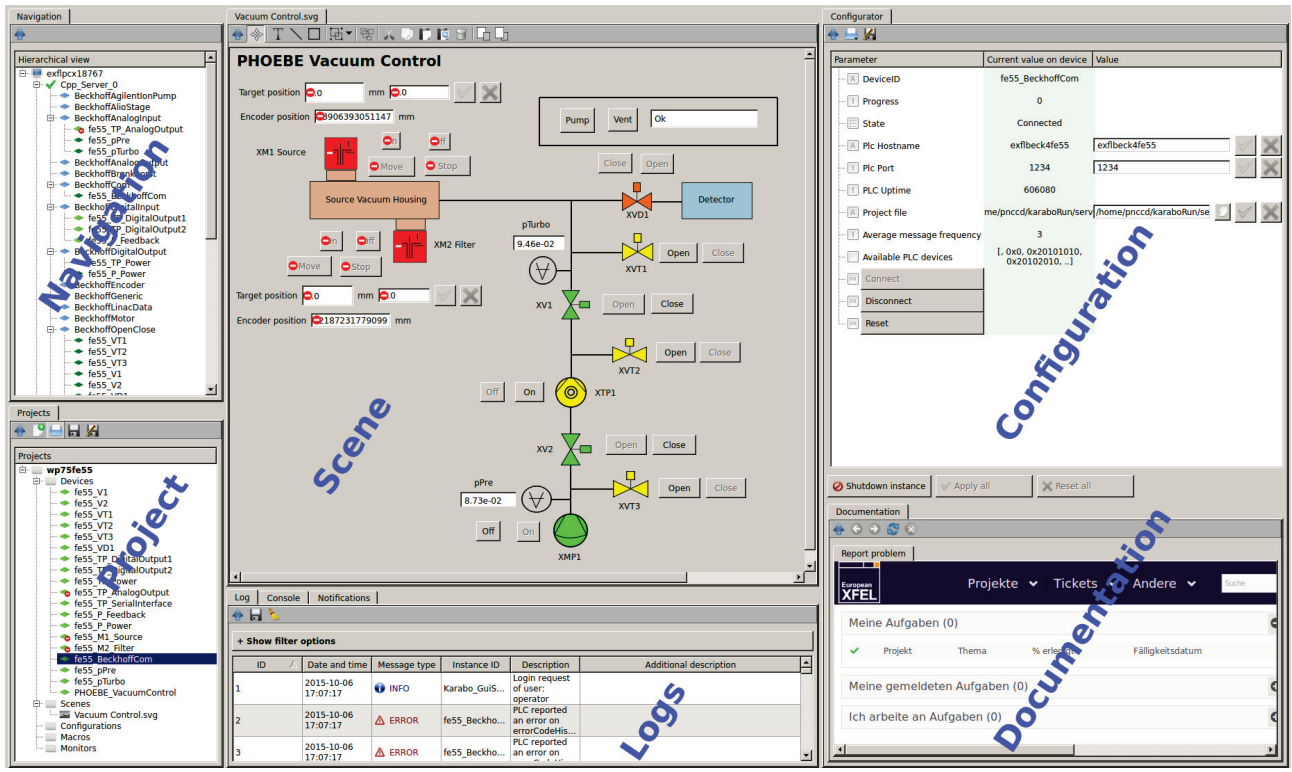


Figure 1: Screenshot of the GUI. (a) The live *Navigation*, with computers, servers, device classes and devices. (b) the *Configuration* of one device, automatically generated from its self-description, (c) a *Scene* with several device properties shown, (d) the *Project*, (e) the message *Logs* and (f) *Documentation*.

GRAPHICAL VISUALIZATION

The properties and commands of devices can be visualized in a graphical *scene*. This is done by simply dragging the property from the device's configuration view into the central panel of the GUI, where those scenes are shown. This is possible for both already running devices and device classes which are known to the system. The latter can be pre-configured and stored in the project, and their properties can be part of a scene.

Within the scene, the user can choose the way the data is shown. Besides variations of ways to show numbers on dials and other numerial indicators, there are also advanced widgets. As an example, numerical values can be shown in a trendline widget, showing the time evolution of a property. By simply moving the viewed time axis to the past, one can even retrieve historical data from Karabo's data logging service.

Commands to a device are represented as push buttons. They may show icons depending on the state of the device.

Once a scene has been designed, it is typically switched from design to production mode, such that the widgets actually react to user input. This can always be switched back, so that the design of the scene can be edited again. Scenes may also be detached from the rest of the GUI. This way a scene effectively looks like an independent application.

Scenes can also be used to design data processing workflows. Devices which produce or consume high-bandwidth data can be dragged into the scene and are shown as boxes with plugs representing the input or output data streams. Those plugs are then connected by "wires", as seen in Fig. 2. For high-throughput data applications, when the devices should run on many computers in parallel, one box in the scene may also represent a large number of computers each running the same device. Samples of the data can also be shown in the GUI for inspection.

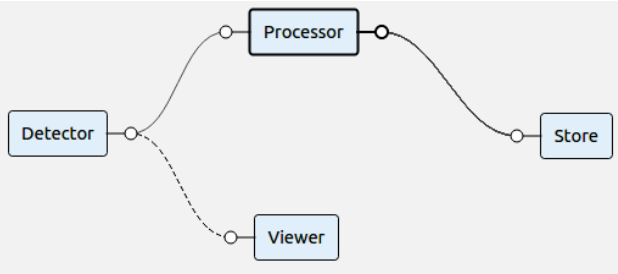


Figure 2: A simple workflow. In this workflow, a *Detector* device creates a data stream which is processed by the *Processor* device group, whose results are sent to the *Store* device. Samples of the data are copied to a *Viewer* device.

Technically, the file format of the scenes follows the SVG standard with some extensions. This allows the user to copy and paste graphical elements from an SVG editor into the scene. While the graphical editing capabilities of the Karabo

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

```
from karabo import *

class Scan(Macro):
    camera = RemoteDevice("camera1")
    start = Float(description="Begin")
    stop = Float(description="End")
    steps = Int()
    average_intensity = Float()

    @Slot()
    def execute(self):
        """Perform a camera scan"""
        with getDevice("motor1") as m:
            m.targetPosition = self.start
            m.move()
            waitUntil(lambda:
                m.status == "stopped")
            self.camera.takeImage()
```

Figure 3: A macro code example. An excerpt of a scan macro showing the syntax of the macro language. Macros are classes with executable slots that perform one task. They can have arbitrary parameters, which serve as input or output. Macros may continuously control devices with `RemoteDevice`, or temporarily with `getDevice`.

GUI are limited, this feature allows for the creation of visually appealing scenes using external SVG editors.

MACROS

While devices in Karabo are also used to automate repetitive tasks, they are often too cumbersome to be used, as they need to be installed on servers by administrators. This is where macros come into play: they are basically Karabo devices which can be entered directly into the GUI and started with a mouse click (they can also be started from the command line, but that is beyond the scope of this article).

The code is not executed in the GUI, as those are often running on user machines with an unreliable internet connection. Instead, they are sent to a central macro server and executed there.

Macros are much more than just a list of commands to be executed sequentially. They are written as a Python class, where the methods of the class can be executed by the user. They essentially behave like devices, so they can also be configured, and their properties can be made editable in a scene. A short example is shown in Fig. 3.

Finally, the only difference with a normal device is that macros should be used for specific tasks which are relevant in a particular context only. Everything generalizable should be made into an actual device and maintained by computer administrators.

LOGGING AND A CONSOLE

Karabo devices can broadcast messages that users may be interested in within the distributed system. Those are shown in a logging panel within the GUI, where those messages can be sorted, searched and filtered.

There are many tasks which can be more easily done on a command line than graphically. Therefore the GUI has a console panel in which an IPython session can be started to control Karabo. The same programmer's interface as for the macros is used, so that commands for a macro may be tested on the command line.

TECHNICAL DETAILS

The GUI is written in Python 3.4, using PyQt4 for the graphical output. Many of the widgets in the scene use PyQwt5 and guiqwt. The same code can be run under Windows, MacOS and Linux operating systems. Connection to a Karabo installation is established via TCP with a dedicated protocol. This way the GUI can also be used outside of the protected network of a Karabo installation.

Like the rest of Karabo, the GUI is completely event driven. The user cannot initiate any blocking operation. When the GUI sends a user's request to the network, instead of waiting for a response the GUI continues to work normally and will show results of the request when it arrives.

Projects are ZIP files, which contain the the data as XML files and in case of the macros, as Python source files. This allows users to inspect and change projects using other tools.

CONCLUSION

The Karabo distributed control system contains a single graphical user interface which integrates everything to control and use an installation. The user interacts directly with the live system. Via the self-description of the running device a user gets an immediate idea about its capabilities, allowing for an intuitive use of the system.

REFERENCES

[1] B. C. Heisen, D. Boukhelef, S. Esenov, S. Hauf, I. Kozlova, L. Maia, A. Parenti, J. Szuba, K. Weger, K. Wrona, C. Youngman, "Karabo: an Integrated Software Framework Combining Control, Data Management, and Scientific Computing Tasks", ICALEPCS2013, San Francisco, CA, USA 2013

[2] http://www.opengda.org

[3] Jan Hatje, M. Clausen, Ch. Gerke, M. Moeller, H. Rickens, "Control System Studio (CSS)", ICALEPCS07, Knoxville, TN, USA, 2007

[4] E. Sombrowski, A. Petrosyan, K. Rehlich, P. Tege, "JDDD: a Java DOOCS data display for the XFEL", ICALEPCS07, Knoxville, TN, USA 2007

[5] http://www.taurus-scada.org

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

VISUALIZATION OF INTERLOCKS WITH EPICS DATABASE AND EDM EMBEDDED WINDOWS

Evgeniy Tikhomolov, TRIUMF, Vancouver, Canada

Abstract

The control system for TRIUMF's upgraded secondary beam line M20 was implemented by using a PLC and one of many EPICS IOCs running on a multi-core Dell server. Running the IOC on a powerful machine rather than on a small dedicated computer has a number of advantages such as fast code execution and the availability of a large amount of memory. A large EPICS database can be loaded into the IOC and used for visualization of the interlocks implemented in the PLC. The information about interlock status registers, text messages, and the names of control and interlock panels are entered into a relational database by using a web browser. Top-level EPICS schematics are generated from the relational database. For visualization the embedded windows available in the Extensible Display Manager (EDM) are the EPICS clients, which retrieve interlock status information from the EPICS database. A set of interlock panels is the library, which can be used to show any chains of interlocks. If necessary, a new interlock panel can be created by using the visualization tools provided with EDM. This solution, in use for more than 3 years, has proven to be reliable and very flexible.

MOTIVATIONS

The new control systems at TRIUMF are implemented by using the Experimental Physics and Industrial Control System toolkit [1]. To control the devices in the upgraded M20 secondary beam line, Programmable Logic Controllers (PLCs) are used [2]. One of the tasks of the control system is to visualize the state of interlocks for various vacuum devices and power supplies. The present scheme for visualization of interlocks, which is used at TRIUMF was designed more than 15 years ago [3], when the old Display Manager with limited capabilities was used, and EPICS support was located in VME Input-Output Controllers (IOCs), which had only 16 Mb of memory, and 100 Megabit network connections.

At present EPICS support can be realized as an application running on a Linux server (we call it PLCIOC). This application communicates with the PLC by using the EPICS driver on one hand and with the EPICS clients on the other. Thus, PLCIOC plays the role of a transfer agent between EPICS clients and PLC: EPICS clients communicate via the network with this server, send/receive commands and set/get values from devices via PLCIOC (Fig. 1).

At present there is effectively no limitation for memory usage by PLCIOC. EPICS database can be of very large size and keep lots of information. At present we use a multi-core Dell PowerEdge 600 with 8 Gb of memory. Memory usage by M20 PLCIOC serving ~100 devices is only 1.5%. Processing information is very fast on such multi-core computers. CPU usage for M20 PLCIOC is ~12% for one core (out of 8). Transfer of data between EPICS clients and PLCIOC is also very fast for a Gigabit network connection.

Another favourable reason for new development was the addition of a new feature to the Extensible Display Manager (EDM) [4] in 2008, namely, possibility of assignment of macro value by a script or executable. Thus, it became possible to implement control device panels, in which interlocks can be represented by an embedded interlock panel, and all necessary information is retrieved from EPICS database (Fig. 2).

Figure 2: Interlock panel (on the left) is embedded in lower part of control device panel (on the right).

The goals of new implementation for interlock visualization are:

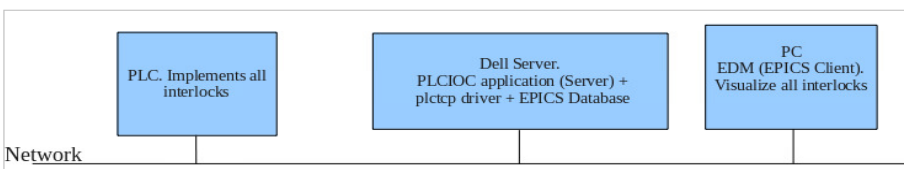


Figure1: Communication between PLC, PLCIOC, and EPICS clients.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Figure 3: Example of hierarchical interlock chains implemented for M20 Secondary Beam Line.

- Less development time for new Control Systems
- More flexible scheme for adding new types of devices and types of interlocks
- Simplification of maintenance (fixing errors in PLC code and relational database)
- Simple handling of very complicated interlocks
- Simple adding of new features (like variable trip points, local force/defeat of the interlocks, applying different colour rules, etc.)

BACKGROUND AND SOME DEFINITIONS

All interlocks are implemented in the PLC. The EPICS screens (device control panels + interlock panels) are used only for visualization of the interlocks. The strict correspondence between implementations of interlocks in the PLC and visualization in EPICS must be maintained during the development. Software support for M20 control system was implemented mostly by 2 people. One person developed PLC code and another EPICS support. The approved Interlock Specification (like shown in Table 1) was used by both developers to work on their tasks in parallel. At the commissioning stage the full implementation of control system (PLC+EPICS) was tested automatically by using developed Perl scripts.

Table 1: Interlock Specification

Device name	Device Type	OK to open/set on
M20:RV2	Roughing valve	A and B A: M20:CG1A< M20:CG2 or M20:CG1A < 50 mTorr B: M20:RV1 closed M20:RVC closed

The concept of 8-bit and 16-bit Status Registers was used both in PLC code and EPICS schematics. The

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

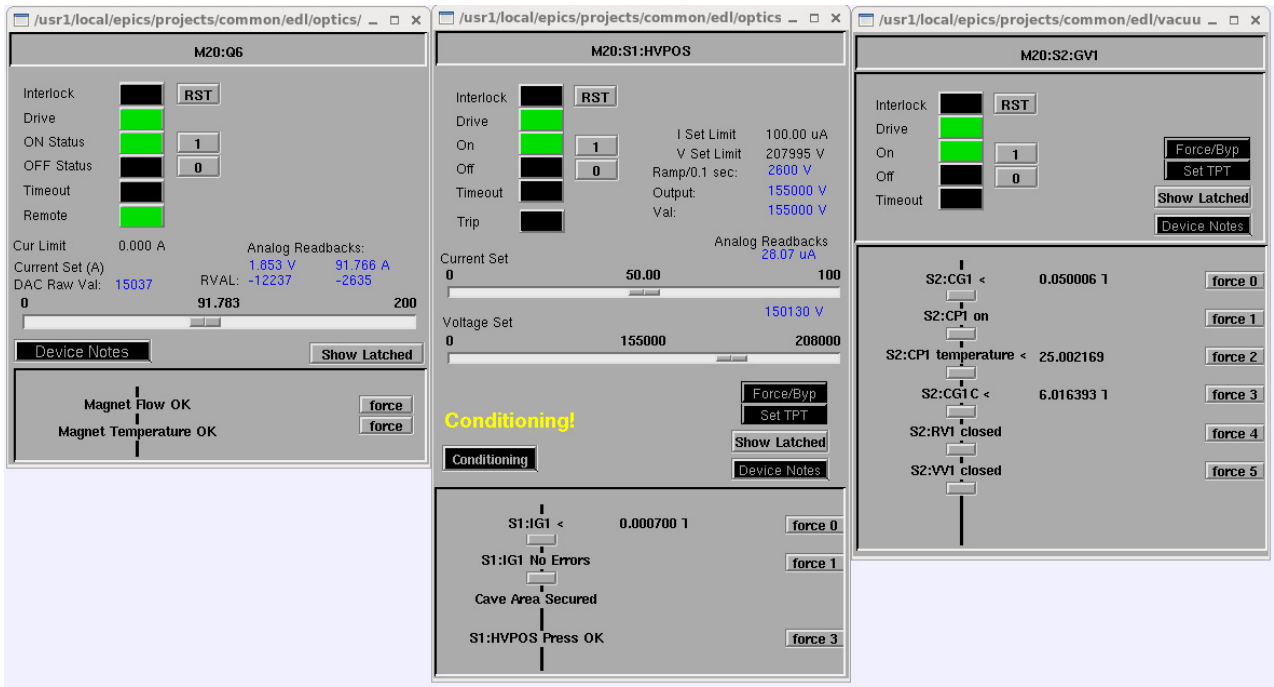


Figure 4: Visualization of interlocks for M20 optical and vacuum devices.

Status Registers contain bits which reflect status of interlock signals. Usually, "1" means interlock condition (i.e. NOT OK) and "0" means no interlock condition (i.e. OK). Some devices can have several status registers to incorporate more than 16 interlocks. Each signal used in registers is latched after getting NOT OK status by PLC code. Reset signal is sent from EPICS device Control Panel to PLC and clears latched interlocks. A single Web Form for entering all kinds of information into relational database was used for any interlock chains.

IMPLEMENTATION

Interlock Visualization in EPICS Control Panels

Device control panels were designed manually by using EDM. Each Control Panel has a container. The reason for using a container is a limitation of EDM: it doesn't allow a script in the File field for an Embedded Window. Otherwise, only a Control Panel with an embedded window for interlocks could be used.

All interlock panels used for visualization of interlocks were also created manually by using EDM. A library of interlock panels can be used for development of any control system. All necessary information for visualization is taken from EPICS Process Variables (PVs), which reside in PLCIOC by using small scripts called by the container. The size of the scripts is very small, ~20 lines.

Production versions of device panels with embedded interlock panels for rather complicated hierarchical vacuum interlocks are presented in Fig. 3. The visualization scheme is universal: for M20 it is used both for optical and vacuum devices in similar manner (see Fig. 4).

Implementation of Interlock Visualization in EPICS Database

A relational database was used for all necessary interlock information and to generate an EPICS database [5]. Special EPICS symbols with the names like plcdevint.sym were created by using TDCT [6]. Related schematic contains all interlock information. All device interlock symbols are aliased in relational database to the names like DEVINT, which are associated with symbol files (CGINT, CPINT, FGINT, etc.)

Using Multiple Status Registers

For each STATUS register the corresponding PV name is constructed as shown here:

<Device Name>INT<Device ID>_S<NSTATUS>,

where NSTATUS is the unique instance of the Status Register. For example, M20:RVINT2_S2.

Rules for Construction of Interlock Panel Names

The Interlock Specification provides logical equations for interlocks. Enumeration of bits starts with zero. For example:

(Bit_0 and Bit_1) and (Bit_2 or Bit_3 or Bit_4) and Bit_5 and Bit_6 and ... (some more bits)

Such equations correspond to the ladder logic of the PLC code. The name of interlock panel can be

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

constructed by using logical equations and applying the following simple rules:

- dash sign represents “and” for groups before and after the sign (corresponds to vertical lines on interlock panels)
- if several bits are used in “and' statements they are combined into the range of bits by using colon sign, for example, 0:1
- the word “or” is used every time to avoid confusion

Thus, the name of the interlock panel, which represents the above mentioned logical equation, looks like:

0:1-2or3or4-5:15.edl (see Fig. 5)

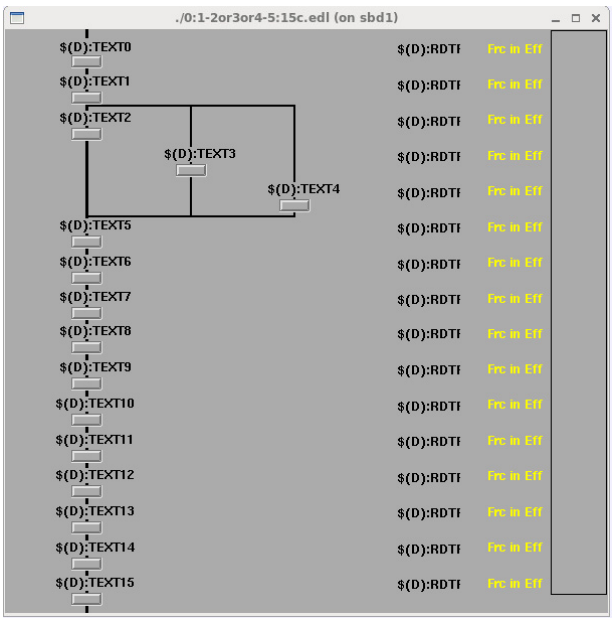


Figure 5: Construction of name for interlock panel from logical equation.

CONCLUSION

A complete scheme for interlock visualization was created. It was used for 3 years on the TRIUMF M20 Secondary Beam Line and proved to be easily maintainable and extensible. It can be recommended for development of any new control system, which uses the EPICS toolkit. The major steps for developing a new system are presented in the Appendix.

ACKNOWLEDGEMENT

The author is thankful to TRIUMF Controls groups and personally to Rolf Keitel and Mike Mouat for providing a very good creative environment and basis for development of the project presented.

APPENDIX

Major Steps to Implement Interlock Visualization with Embedded Windows

- Define PV name for each device. It should have a structure:

<Device Name><Device ID>,

where <Device Name> includes names for Subsystem, sub-subsystem, and device type, for example: M20:S1:RV2

- For each device construct a corresponding interlock device name:

<Device Name>INT<Device ID>, for example: M20:S1:RVINT2.

This name is used for creation of a device using the standard initial web form for relational database

- Enter all required information into relational database via web page form
- build EPICS database and load into PLCIOC
- if necessary, create new device control panel and interlock panel by using EDM

REFERENCES

[1] http://www.aps.anl.gov/epics/. Experimental Physics and Industrial Control System.
[2] http://www.schneider-electric.com/en/product-range/1468-modicon-m340/
[3] R. Keitel, R. Nussbaumer, “Automated Checking and Visualization of Interlocks in the ISAC Control System”, Proceedings of ICALEPCS 2001, Nov 27-30, 2001, http://arxiv.org/abs/physics/0111094
[4] http://ics-web.sns.ornl.gov/edm/
[5] Keitel, R., “Generating EPICS IOC databases from a relational database: A Different approach”, Proceedings of ICALEPCS 2001, Nov 27-30, 2001, http://arxiv.org/abs/physics/0111048
[6] Keitel, R., "TDCT – A Configuration Tool for EPICS Runtime Databases," Proceedings of ICALEPCS 2009, Oct 12-16, 2009

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

IMPROVING SOFTWARE SERVICES THROUGH DIAGNOSTIC AND MONITORING CAPABILITIES

P.Charrue, M.Buttner, F.Ehm, P.Jurcso, CERN, Geneva, Switzerland

Abstract

CERN's Accelerator Controls System is built upon a large set of software services which are vital for daily operations. It is important to instrument these services with sufficient diagnostic and monitoring capabilities to reduce the time to locate a problem and to enable pre-failure detection by surveillance of process internal information. The main challenges here are the diversity of programs (C/C++ and Java), real-time constraints, the distributed environment and diskless systems. This paper describes which building blocks have been developed to collect process metrics and logs, software deployment and release information and how equipment/software experts today have simple and time-saving access to them using the DIAMON console. This includes the possibility to remotely inspect the process (build-time, version, start time, counters,..) and change its log levels for more detailed information.

INTRODUCTION

Operating the CERN accelerators relies on a Controls Infrastructure in a perfect state. The CERN Beams Controls Infrastructure is deployed all around the CERN site, spanning from the Operator console in the control room to the device connected to a distant fieldbus close to the beam, via hundreds of servers, FrontEnds, networks and of course software applications.

The Beams Controls Group has studied, developed and deployed a complete diagnostic and monitoring infrastructure (DIAMON)[1] capable of detecting and repairing faults in the whole Controls infrastructure.

DIAMON

DIAMON is following the classic multi-layer system approach allowing high level of flexibility. As illustrated in Figure 1, specific DAQ modules, based on the C²MON[2] DAQ Core assure the communication between a variety of data providers and the DIAMON server. The communication can be uni- or bi-directional (support for settings and commands).

With a set of DAQ core plugins we establish communication to all required types of devices. The DIAMON server handles the messages received from the DAQ layer, computes the state of the rules (business logic) and provides the results to the client applications through C²MON Client API. The communication between components of the system layers is based on JMS messages and the scope of the monitored information is configured through a set of web applications operating on DIAMON configuration database. All configured points (further called *metrics*), alarms and rules (so-called *limits*) are handled inside the server's internal cache and the states are periodically persisted into the server's cache-persistency database. Updated metrics and limits are also periodically written into the Short-Term-Log (STL) database, which holds thirty days snapshots and provides the data on demand to the client applications. This allows our users to replay the chain of events and precisely analyse the problems retroactively. For all GUI clients and web-based client applications an instance of the Tomcat servlet container has been set-up, hosting a Spring MVC-based Java module which renders a set of web pages with the data received at runtime from the server as well as the offline data from the STL database.

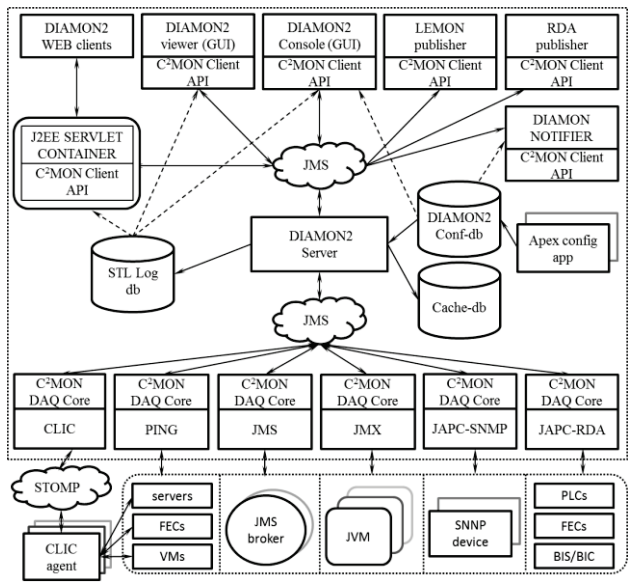


Figure 1: Overview of the DIAMON architecture.

THE DIAMON CONSOLE

The DIAMON console is the main tool that is used to retrieve and display monitoring data in a simple and homogenous way.

Specific views can be configured to target the family of devices one need to monitor, being all computers involved in the control of a specific accelerator, devices providing a specific service (such as Timing, Middleware, …), devices belonging to a specific group (such as Beam Instrumentation, RF, Power Converters, …) or the complete list of all monitored devices (~3000 at the time of writing).

Standard views are made available directly from the DIAMON console to cover the most used views, such as

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

one view per CERN accelerator (LINAC, BOOSTER, PS, SPS, LHC, …), one per most used equipment group (BI, RF, …), one per main framework (CMW, OASIS, …).

Figure 2 below gives a screenshot of the DIAMON console.

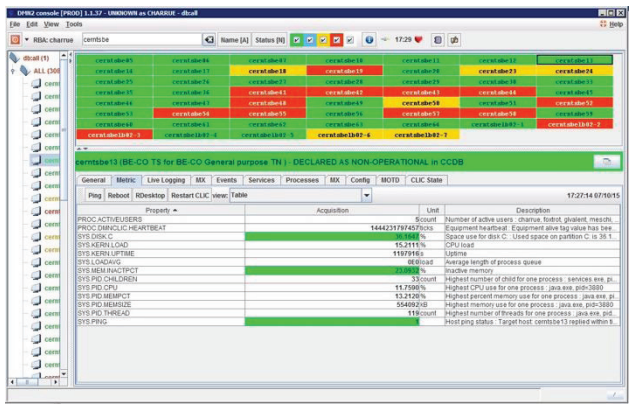


Figure 2: Diamon GUI.

The application window contains three separate panels:

- The tree on the left-hand side provides a hierarchical view on the monitored equipment, for instance with 3 levels: the accelerator, the controlling computer, the connected equipment. Users can configure the levels and content of the tree.
- The top right panel displays the full list of equipment corresponding to the element selected in the tree. The background color clearly displays the status of each item (green: OK; yellow: warning, red: error).
- The bottom right panel (the "Detail panel") displays all details for the element selected in the left hand-side panel (if the selected element is a leaf node) or in the top right panel.

DATA SOURCES

As shown in the bottom of Figure 1, the DIAMON infrastructure is fed with data from many different sources:

The first source of data is the CLIC agent running in every computer deployed in the controls infrastructure. It runs in fixed intervals of time and collects data about the system measurable such as CPU load, memory or disc usage, network throughput, presence of specific system and user software as well as data about connected hardware (timing boards, World FIP bus).

Then a few centralized agents running on a dedicated central server poll lists of devices to obtain the information to be monitored. This approach is used to monitor controls infrastructure equipment, where software can not easily be installed, like PLCs, video converters, power supplies etc.

In addition, DIAMON verifies in regular intervals network availability of all monitored computers using the PING DAQ.

DIAMON is also capable of exploiting information gathered by the Tracing initiative [3] from several SYSLOG and tracing files produced by the Controls computers. This information gives real time information from the running Operating System but also from Controls Frameworks such as FESA [4] (our FrontEnd realtime framework), CMW [5] (our Controls Middleware communication infrastructure) or our framework deployed to remotely restart specific application (wreboot)

GATHERING DATA FROM APPLICATIONS

The Controls Infrastructure is also composed of a huge number of JAVA and C++ applications used to control and drive the beams through the several machines in our accelerator chain. The presence and correct behaviour of these software applications are essential for the operation of our accelerators.

Therefore, DIAMON needs to be informed about any problems in the software applications deployed in the controls infrastructure. For this purpose, a specific framework for Java and C++ application has been deployed, JMX and CMX [6], allowing specific metrics to be published as shown in figure 3 below.

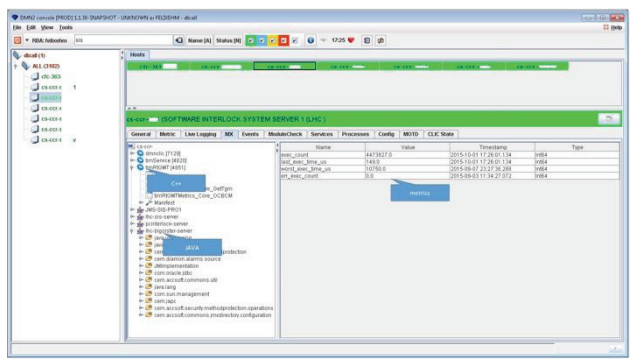


Figure 3: Metrics sent by C++ or JAVA controls application

Moreover, this framework can display the sometimes complex dependency graph allowing the detection of mismatch in the versions used to build specific application, as shown in Figure 4 below.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

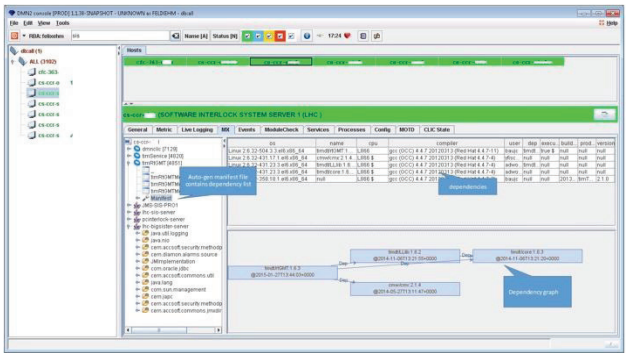


Figure 4: Display of a dependency graph for a C++ controls application

APPLICABILITY

This initiative to gather extremely different data coming from many different source into the same DIAMON tool has been very much appreciated both by the Controls experts and by the Beams Operators.

As shown on Figure 2, the Detail Panel offers a wide range of tools to access to diagnostic data, automatically configured according to the device being selected. Static data, such as the location of the device, the person responsible for its maintenance, the hardware and software configuration, etc, are available with one click of the mouse. Up to 30 days of history stored in the DIAMON server allow to easily visualise the last events on the device, such as reboot, restart of specific process, logins, …

Live logs, with on-the-fly tuning of the log level, can be visualised in real time, allowing a visualisation of the current behaviour of the selected device.

Users identified via our Role Based Access Control (RBAC) and having the correct authorisation rights can, still using the same standard DIAMON console, perform some operation on the selected device, such as triggering additional data acquisition, restarting selected processes or rebooting a machine.

The initial target of reducing the diagnostic time and of correlating data from heterogeneous sources has been reached today.

The CERN Controls Infrastructure contains more than 8'000 intelligent computers capable of producing more than 150'000 metrics. Thanks to the standard DIAMON console, to the JMX/CMX standardisation and to the concentration of the tracing initiative, it is now much easier to diagnose and make the correct action to solve a controls issue.

In addition, controls experts as well as equipment specialists are running local configuration of the DIAMON tool, with a specific target on the devices and infrastructure under their direct responsibility. By using all the DIAMON facilities (such as information mails or SMS), by correctly tuning the thresholds on the metrics under scrutiny, appropriate pro-active measures can be prepared and applied, thus avoiding possible problems to appear.

CONCLUSION

DIAMON infrastructure is a very powerful tool to gather diagnostic information on our distributed Controls Infrastructure. The investment into standardising and acquiring data from many different sources allows for a faster and better diagnostic of blocking situation for the Beams Operation, hence reducing our diagnostic time and therefore limiting to the maximum the time lost due to issues in the controls infrastructure.

In addition, this valuable collected data is also used by the controls experts to make pro-active actions that prevent an emerging issue to perturb Operation.

REFERENCES

[1] Diagnostic and Monitoring the CERN Controls Infrastructure: The DIAMON Project: First Deployment in Operation, ICALEPCS'09, Kobe, Japan, (2009)

[2] A Suwalska, M. Buttner, M. Brager, W. Buczak "C^{2}MON CERN control and monitoring platform", CERN, (2011)

[3] ACET – Accelerator Controls Exploitation Tools - https://wikis.cern.ch/display/ACET/Tracing

[4] FESA – Front End Software Architecture, https://wikis.cern.ch/display/FESA3

[5] CMW – Controls Middleware infrastructure, https://wikis.cern.ch/display/MW/

[6] F. Ehm et al. "CMX - A Generic In-Process Monitoring Solution for C and C++ Applications" Proceedings of ICALEPCS'13, San Francisco, USA, (2013)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DEVELOPING DISTRIBUTED HARD-REAL TIME SOFTWARE SYSTEMS USING FPGAS AND SOFT CORES

T. Włostowski, J. Serrano, CERN, Geneva, Switzerland
F. Vaga, University of Pavia, Italy

Abstract

Hard real-time systems guarantee by design that no deadline is ever missed. In a distributed environment such as particle accelerators, there is often the extra requirement of having diverse real-time systems synchronize to each other. Implementations on top of general purpose multitasking operating systems such as Linux generally suffer from lack of full control of the platform. On the other hand, solutions based on logic inside FPGAs can result in long development cycles. A mid-way approach is presented which allows fast software development yet guarantees full control of the timing of the execution. The solution involves using soft cores inside FPGAs, running single tasks without interrupts and without an operating system underneath. Two CERN developments are presented, both based on a unique free and open source HDL core comprising a parameterizable number of CPUs, logic to synchronize them and message queues to communicate with the local host and with remote systems. This development environment is being offered as a service to fill the gap between Linux-based solutions and full-hardware implementations.

BACKGROUND

Real-time controls in particle accelerators cover a broad range of applications that have different requirements for processing power, latency and determinism. For the purpose of this article, we divided these into the three following categories:

1. less than a microsecond of latency, such as beam injection and extraction, low level RF, interlocks and machine safety systems,
2. from dozens of microseconds to one millisecond, examples, power converter controls, orbit feedback or a number of timing and event distribution systems,
3. slow controls, such as cryogenics, vacuum, positioning or radiation monitoring.

Usually the parameter that has the highest impact on the architecture of the control system electronics is the worst case (never to be missed) processing latency. The sub-microsecond controls are the exclusive domain of FPGAs, ASICs and dedicated analog circuitry. On the contrary, the systems from point 3 are usually implemented on Programmable Logic Controllers (PLCs), or in many cases on industrial PCs running non-real time operating systems.

The widest variety of controls applications lies in between these two extremes. While the actions executed by these systems usually have strictly specified execution times, the data processing that triggers these actions can be done ahead in time, provided that the processing latency is guaranteed to never exceed a certain upper limit, as shown in Figure 1. For example, the receiver of the CERN General Machine Timing (CTR) can produce pulses with 40 nanosecond resolution, but the minimum time between reception of a timing event and the generation of a corresponding pulse is almost one millisecond.

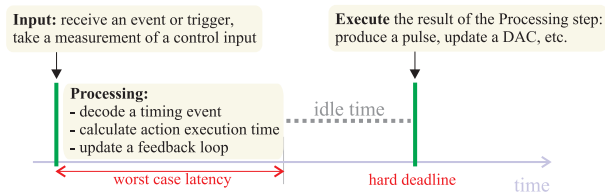


Figure 1: A typical hard-real time control system.

The traditional design approach in such systems, using an FPGA and developing custom HDL cores, brings several disadvantages:

- Only a small part of the HDL code implements the strictly timed part. In case of the CTR, these are the counters which drive the outputs and constitute around 20 % of the total HDL code. Event reception, filtering, logging and counter configuration logic has much slower execution time constraints.
- HDL development and testing take much more time compared to software development.
- Custom HDL cores need custom drivers, requiring additional development manpower.

More recent designs rely on soft processor cores, such as Xilinx MicroBlaze [1] or Altera Nios [2], which are responsible for the "slow" part, connected to custom logic implementing the "fast" part. This reduces the development time, but still leaves the following issues:

- No standard way of low-latency communication and synchronization between the devices (if they form a distributed system).
- Vendor lock-in - the cores provided by the FPGA vendors are not portable to other FPGAs,
- Issues with portability of drivers and low level host software, which are different for each FPGA manufacturer and each application.

In this article, we present the *Mock Turtle* (MT) - an HDL and software framework targeted at hard real-time distributed applications, offered as a ready to use service. *MT* provides:

- A deterministic multi-core CPU system that can be freely interfaced to user logic.
- A standardized way of communication with the host machine and (if needed) other devices in a distributed network.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- Optional built-in White Rabbit (WR) [3] synchronization.
- A generic Linux device driver, user space library and a set of GNU-based development tools.

The following chapters present the architecture of the HDL and software stack of the MT, the project status and future goals. We also describe the architecture of the *uRV* - an open source soft CPU core, being developed at CERN and targeted at hard-real time embedded applications.

HDL ARCHITECTURE

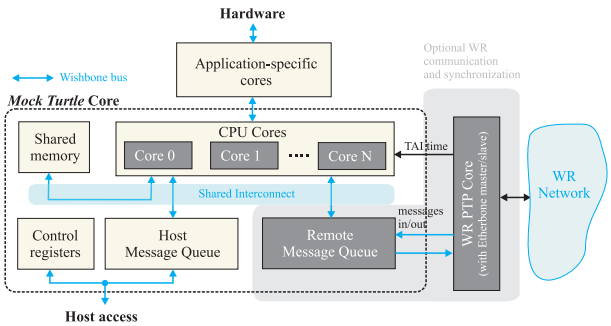


Figure 2: *Mock Turtle* Core Architecture.

Figure 2 shows a simplified block diagram of a *Mock Turtle*-based system. The key ingredients of the *MT* design, described in the following subsections, are:

1. Up to 8 32-bit CPU cores with a Shared Memory block.
2. Message queues, enabling communication with the host system and optionally, remote MT nodes.
3. User cores, controlled by the *MT* CPUs.
4. Control and debug logic.

All these modules are interconnected using Wishbone [4] buses, with a central, multiport crossbar SI (Shared Interconnect).

The CPUs and Shared Memory

MT provides a user-selectable number (1 to 8) of 32-bit RISC processor cores, organized into Core Blocks (CBs) as depicted in Figure 3. Each CPU has a private program and data space, located in the FPGA internal memory, whose contents can be uploaded at any time by the host software. This approach allowed to maximize the determinism of code execution by avoiding unpredictable wait states caused by arbitration of accesses to a shared memory block. For the same reason, the CPUs do not support interrupts - every asynchronous request must be serviced by means of polling.

The CPU core we used was initially a modified version of the *Lattice Mico32* (LM32) [5], due to its portability, speed and low footprint. Recent versions of the *MT* use the *uRV* RISC-V CPU core (described in a separate section of this article). We decided to switch to RISC-V as its architecture looks more viable in the long term than the proprietary architecture of LM32 (the LM32 license, despite being open source, contains export restrictions).

In order for the CPUs to communicate with each other, the *MT* core provides a Shared Memory (SMEM) block of user-configurable size, accessible by all the CPUs in an atomic way. The SMEM facilitates implementation of common inter-process communication primitives such as semaphores, mutexes, locks and queues, by providing a set of atomic operations: add, subtract, bit set, bit clear, bit flip and test-and-set. These operations are executed by reading or writing to the SMEM with the high address bits selecting the desired atomic operation (example in Listing 1). The SMEM can be also accessed by the host software, with the same atomic features as those available for the CPU cores.

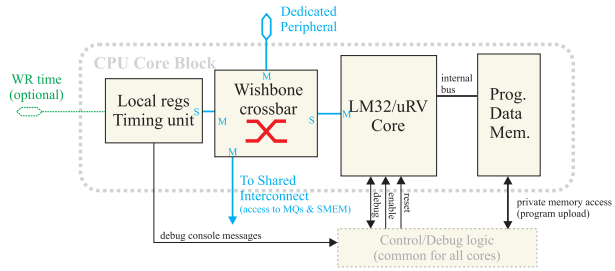


Figure 3: Design of the CPU Core Block (CB) in the *MT*.

Each CB also contains a set of Local Registers, accessible exclusively from the CB's own CPU. These registers let the CPU perform the following operations without disturbing other CPUs (i.e. without using shared resources):

- Get the current time (provided by WR or a local counter).
- Execute accurate delays (through self-decrementing delay generation registers).
- Poll the presence of received messages in the Message Queues.
- Send debugging messages to the host system.

Listing 1: SMEM access semantics example

```
#define OFFSET_ADD      0x00010000
#define OFFSET_TEST_SET 0x00020000

// atomically adds x to the *var
void atomic_add(uint32_t *var, uint32_t x)
{
  *(uint32_t *) (var + OFFSET_ADD) = x;
}

// atomically sets to *varto 1, returns
// the value of *var before set
uint32_T atomic_test_set(uint32_t *var)
{
  return *(uint32_t *) (var +
      OFFSET_TEST_SET);
}
```

The Communication System

The communication system consists of two message queues, shared by the CPUs. The Host Message Queue (HMQ) passes messages between the CPUs and the host system. The HMQ is the primary means of communication

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

between the node and the host software (e.g. FESA [6]). Presence of an outgoing message is indicated to the host by raising an interrupt.

The optional Remote Message Queue (RMQ) exchanges messages with remote nodes in the WR network. The Etherbone protocol [7], developed by GSI, is used as the transport layer. Writing a message in the RMQ automatically sends out a UDP packet containing the message.

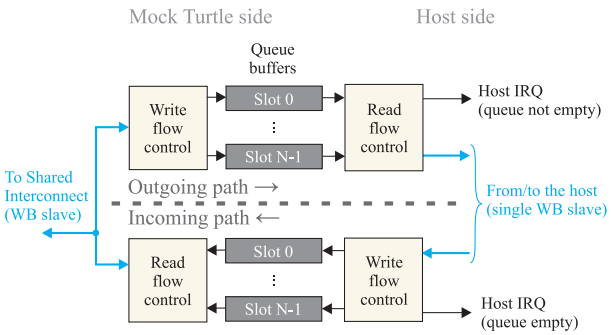


Figure 4: Design of the Host Message Queue.

The structure of the HMQ is depicted in Figure 4. The MQs are multi-word FIFOs, containing a number of unidirectional communication channels called *slots*. Outgoing slots transfer messages from *MT* CPUs to the Host or other MT nodes. Conversely, incoming slots let the CPUs receive messages sent by the host or the remote nodes. The number of slots is configured by the user. Each slot can buffer a user-configurable number of messages of configurable maximum size. Multiple slots allow handling independent message streams from different sources - for example, the Trigger Distribution application [8] uses a pair of incoming/outgoing slots for the control commands and a separate outgoing slot to stream the log of executed triggers.

In order to send a message, the transmitter writes a number of words to an MQ outgoing slot and marks it as ready to send. The receiver side gets an indication that its MQ incoming slot is not empty, reads out its contents and indicates to the MQ that it has processed the message. MQs ensure integrity of the messages: if the message is not received completely (i.e. the Etherbone core connected to the RMQ reported an error), it is not received at all.

Since *MT* is designed for hard real-time applications, the MQs do not have any flow control: if an MQ slot becomes full, it starts to drop the incoming messages. Users may implement flow control in software if needed, although in all *MT* applications we foresaw, any buffer contention is considered erroneous, as it may break the deterministic behavior of the system. We are developing a Forward Error Correction core to improve the robustness of message delivery without using retransmission.

Connectivity

The *MT* can interface with the user HDL design in two ways:

- Each CPU Block has a Dedicated Peripheral (DP) Wishbone master port that can be connected to the core's private peripherals. For example, in the WR Trigger Distribution [8] node, the CPU core responsible for reading out timestamps has a Time-To-Digital Converter Core connected to its DP port.
- The Shared Interconnect provides a Share Peripheral (SP) Wishbone master, that all cores can access concurrently. A typical use case of the SP is connecting large memories for storing auxiliary data or diagnostics info.

From the host side, all *MT* features are accessible through a Wishbone slave port and an interrupt line. The Self Describing Bus (SDB) [9] is used to automatically discover all the *MT* cores in the system.

THE *URV* SOFT PROCESSOR

The *uRV* core (expanded as micro RISC-V) is a small, robust and open source soft CPU core, targeted at deeply embedded FPGA applications, such as the *MT* or the WR PTP Core [10]. We took the following assumptions when designing the *uRV*:

- Fully open (no patent/trademark limitations) and standard ISA (Instruction Set Architecture).
- Optimize for modern FPGA resources and executing applications primarily from the FPGA's internal memory. Balance optimization effort between clock speed and FPGA area.
- Design with portability in mind. Separate the FPGA-specific features from the common CPU code.
- No high level operating system facilities, such as an MMU (Memory Management Unit). We intend the core to run low-level deterministic applications.

We decided to use the RISC-V [11] ISA with Multiply/-Division extension (RV32IM) [12], developed at Berkeley University. The primary reasons were simplicity (27 base integer instructions with clearly defined extensions), clear legal status of the ISA and availability of high quality development tools (recent GCC and Clang ports). Additional reason to support RISC-V was the vivid and diverse developer community, with ongoing high-performance silicon implementations and backing of several universities and companies.

Core Architecture

uRV employs a modified Harvard architecture: code and data reside in a shared 32-bit memory space, but are accessed through separate memory interfaces. Instructions are executed by a four-stage, single-issue pipeline, shown in Figure 5 and consisting of the following stages:

- Fetch (F), calculating the address of the next instruction and requesting it from the memory,
- Decode (D), which also computes the immediate values (sign-extensions and bit reordering), pre-computes the operand values for the Execute 1/Memory (X1/M)

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

stage and manages the hazards by inserting empty instructions into the Execute stage.
- Execute 1/Memory (X1/M), which executes most of the instructions, generates memory addresses and issues memory read/write requests.
- Execute 2/Writeback (X2/W), which completes execution of Load, Multiply and Shift instructions and writes the result back to the CPU registers.

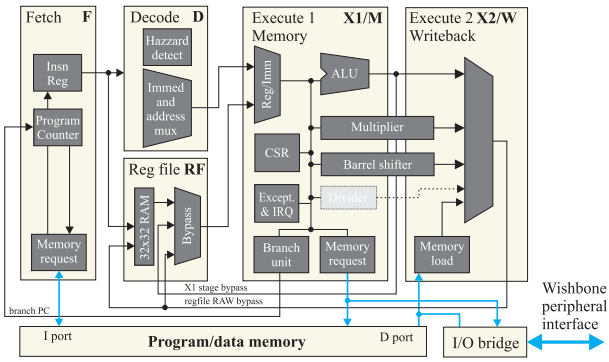


Figure 5: Design of the *uRV* pipeline.

In parallel to the D stage, there is a Register File (RF), hosting the 32 architectural registers (`x0 - x31`). The RF is built on top of two FPGA RAM blocks, providing two read ports and one write port, with a single clock cycle latency.

The core includes a platform-optimized barrel shifter and multiplier (both with 2 cycle latency) and an optional, multicycle division/modulo unit. Most of the instruction results are bypassed to achieve interlock-free execution of dependent instructions, either by the Read-After-Writer (RAW) bypass path in the RF or after the X2/W stage.

uRV supports the RISC-V Control and Status Registers (CSRs), interrupts and a limited set of exceptions, although the interrupt CSR layout has been simplified (with respect to the RISC-V specification) to minimize the occupied FPGA area. Exceptions are used to implement loads and stores at non-aligned addresses, as well as to emulate instructions not supported in hardware (division, modulus, upper 32 bits of multiplication).

Design Trade-offs

In order to achieve a relatively high clock frequency, combined with a reasonable area, we have applied several optimizations:
- The most timing-critical instructions, such as Multiply and Shift are distributed over two pipeline stages (X1/X2).
- Multiply, Shift and Load instructions are not bypassed to minimize the long combinatorial X2/W stage bypass path. Instead, the D stage raises an interlock in case of a RAW hazard. Surprisingly, with the instruction scheduling done by modern optimizing compilers, there are very few situations where the result of the current instruction is immediately needed by the next one.
- Multiplexing and bypassing of the ALU operands is carefully split between D and X1 stages.

Despite these trade-offs, with proper instruction scheduling, *uRV* can execute all instructions except division and jumps in a single clock cycle. Division takes 37 clock cycles, taken jumps have a constant 3-cycle penalty and missed jumps consume one cycle.

Connectivity

The core provides two simple buses for accessing the memory space, with support for memory wait states. These buses can be connected directly to an FPGA RAM block. The I/O peripherals are accessed through a dedicated Wishbone master, which is controlled by a bridge connected to the data memory bus. The internal memory buses can also connect the CPU pipeline to the instruction/data cache (currently under development).

Performance

An example implementation of an *uRV*-based system, incorporating a GPIO port, UART and 64 kilobytes of RAM takes 1210 LUTs, 954 FFs, 34 Block RAMs and 3 DSP cells on a Spartan-6 series FPGA, achieving a clock speed of 100 MHz (toolchain set up to minimize area).

The core successfully passes the official RV32IM test suite as well as the Coremark 1.0 [13] benchmark. The comparison of Coremark scores against other popular 32-bit architectures[1] is presented in Table 1. Despite the low footprint and several design trade-offs, *uRV* combined with a modern GCC tool chain (version 5.2) performs remarkably well.

Table 1: Coremark 1.0 Results Comparison for *uRV* Against Popular 32-bit CPU Cores

Core and platform	Compiler	Score/MHz
uRV (Spartan-6)	GCC 5.2.0 -O2	2.14
Nios II/f (Altera FPGA)	GCC 4.9.2 -O2	1.87
Cortex-M3 (STM32F103)	GCC 4.4.1 -O3	1.80
LM32 (*MT*)	GCC 4.5.3 -O3	1.78

Status and Outlook

The *uRV* is an ongoing project. In the nearest future, we are planning to:
- Add a Debugging Unit and JTAG interface,
- Implement caches, enabling access to large external memories through a Wishbone.B4 [4] or AXI4 Lite bus.
- Add floating point support for more demanding controls applications (e.g. power converter control).

[1] Scores of the competing implementations have been taken from the official Coremark result repository [13]. Note that Coremark measures both the performance of the hardware and the efficiency of the C compiler. Different compiler versions may give different scores.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

The core can be already used within the *MT*, and the WR PTP core is being upgraded to use *uRV*. We expect significant savings in the FPGA area due to the lower footprint and the more compact code size of the RISC-V ISA.

SOFTWARE

The *MT* software stack, presented in Figure 6 is made of three components: a Linux driver, a Linux library and a dedicated library for real time development. All these components together make a framework ready to be used for the final application development. This framework helps the developers by hiding all the MT details so they can immediately start coding the user-space and the real-time applications running on the MT soft cores.

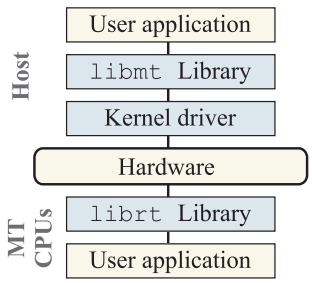


Figure 6: Architecture of *MT* software.

The driver exports all the *MT* features to user space:

- Cores management: load/dump the binary containing the real-time applications to/from each soft core's memory. Pause execution on selected cores, completely enable/disable or reset the core.
- MQ access: send and receive messages to/from the running real time applications. Both synchronous and asynchronous messaging APIs are available.
- SMEM access: access any location on the shared memory in concurrency with the running real time applications.
- Debug interface: read and dump all the debug messages coming from the running real time applications.

Except for the SMEM access, which uses `ioctl()`, all other features are usable directly from the shell through `sysfs` attributes or character devices.

To optimize driver usage the framework provides a library and a Python wrapper, with the same functionality. The library allows the developer to access directly all driver features, listed above. On top of this, developers can build application-specific libraries or directly write their applications. The Python wrapper can be also used for implementing tests or to easily decouple the development of the host application and the real time one.

The real time application library provides a set of tools to ease the communication with the host system. Mainly, it hides from the real time developer all the communication layer with the host system so that in the real time code it is reduced to what is really important for the real time tasks. The library also provides functions to allow the host application to read/write the memory locations explicitly exported by the real time application (variables, SMEM, hardware registers, structures).

A simple request-response communication protocol has been defined for the communication over the HMQ between the host and the real time application. To really take advantage of the full software stack it is suggested to use this protocol, but developers are free to implement alternative solutions.

PROJECT STATUS

The *MT* framework is the base platform for several projects at CERN, such as:

- The Trigger Distribution system for the LHC Beam Instability Diagnostics [8], already operational in the LHC.
- The proof-of-concept RF over Ethernet distribution system [8].
- The WorldFIP Master controller card [14].

MT is supported on the SVEC [15] and SPEC [16] FPGA Mezzanine Card (FMC) carrier boards. The HDL core, software and the documentation can be found at [17]. The *uRV* CPU source code and test software is available at [18].

CONCLUSIONS

Writing C is faster and less error-prone than writing VHDL or Verilog. A whole family of problems can benefit from a cast into software problems while still guaranteeing real time behavior. *Mock Turtle* aims at providing a modular open source solution which can be the seed of a collaborative community effort. We expect this platform to provide a robust basis for the development of distributed hard real time controls and data acquisition systems, reducing risks and development time.

REFERENCES

[1] Xilinx Corporation, Microblaze soft CPU IP Core: `http://www.xilinx.com/tools/microblaze.htm`

[2] Altera Corporation, Nios II soft CPU IP Core: `https://www.altera.com/products/processors/overview.html`

[3] J. Serrano, M. Cattin, E. Gousiou, E. van der Bij, T. Włostowski, G. Daniluk, M. Lipiński, "The White Rabbit Project", IBIC2013, Oxford, UK (2013).

[4] Wishbone bus specification, version B.4: `http://cdn.opencores.org/downloads/wbspec_b4.pdf`

[5] Lattice Corporation, Mico32 soft CPU IP Core: `http://www.latticesemi.com/`

[6] FESA 3 - the new Front-End software framework at CERN and the FAIR facility. A. Schwinn, S. Matthies, D. Pfeiffer, M. Arruat, L. Fernandez, F. Locci, D. Saavedra, PCAPAC2010, Saskatoon, Canada (2010).

[7] M. Kreider, R. Baer, D. Beck, W. Terpstra, J. Davies, V. Grout, J. Lewis, J. Serrano, T. Włostowski, "Open borders for system-on-a-chip buses: A wire format for connecting large physics controls", Phys. Rev. ST Accel. Beams, vol. 15 (2012).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

[8] T. Włostowski, D. Cobas, F. Vaga, J. Serrano, G. Daniluk, "Trigger and RF Distribution Using White Rabbit", ICALEPCS2015, Melbourne, Australia (2015).

[9] A. Rubini, W. Terpstra, M. Vanga, Self-Describing Bus (SDB) Specification for Logic Cores (version 1.1), http://www.ohwr.org/documents/428

[10] G. Daniluk, "White Rabbit PTP Core: the sub-nanosecond time synchronization over Ethernet", *M.Sc. thesis*, Warsaw University of Technology (2012), http://www.ohwr.org/documents/174

[11] University of Berkeley, The RISC-V project, http://riscv.org

[12] A. Waterman, Y. Lee, D. Patterson, K. Asanovic, RISC-V ISA Specification (version 2.0), http://riscv.org/spec/riscv-spec-v2.0.pdf

[13] Coremark CPU benchmark homepage, https://www.eembc.org/

[14] *MasterFIP* project homepage:' http://www.ohwr.org/projects/masterfip/

[15] Simple VME64x FMC Carrier project homepage: http://www.ohwr.org/projects/svec/

[16] Simple PCI Express FMC Carrier project homepage: http://www.ohwr.org/projects/spec/

[17] *Mock Turtle* project homepage: http://www.ohwr.org/projects/wr-node-core/

[18] *uRV* RISC-V CPU Core: https://github.com/twlostow/urv-core

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE LANSCE FPGA EMBEDDED SIGNAL PROCESSING FRAMEWORK*

J. Hill, LANL, Los Alamos, NM 87544, USA

Abstract

During our replacement of some LANSCE LINAC instrumentation systems we have developed a common architecture for timing system synchronized embedded signal processing systems. Our design follows trends of increasing levels of electronics system integration; a single commercial-off-the-shelf (COTS) board assumes the roles of analogue-to-digital conversion and advanced signal processing while also providing the LAN attached EPICS IOC functionality. These systems are based on agile FPGA-based COTS VITA VPX boards integrating a VITA-57 FMC mezzanine site. Our signal processing is primarily developed at a high level specifying numeric algorithms in software source code to be integrated together with COTS signal processing intellectual property components for synthesis of hardware implementations. We will discuss our requirements, our decision point selecting the VPX together with the FMC industry standards, the benefits along with costs of system integrating multi-vendor COTS components, the design of some our signal processing algorithms, and the benefits along with costs of embedding the EPICS IOC within an FPGA.

SCOPE

Legacy VAX backplane integrated RICE control and instrumentation hardware systems are in service now at LANSCE for forty years. Our maintenance costs only increase and legacy schemes applying a single type of multiplexed ADC for all input signals are now obsolete. We are therefore compelled to replace them.

This paper describes electronics upgrade for a subset of RICE signals requiring real-time synchronization with a timing-system maintained data structures determining the modal operation of the facility. For example, we must flavour-stamp the incoming data streams from the sensors for correlation with the beam species being produced. The system described herein will be initially deployed within replacement loss-monitor, current-monitor, and beam-position-monitor systems at LANSCE. The embedded processor and support software described herein are utilized also within the upgraded RF control systems.

OVERARCHING REQUIREMENTS

The design is based on a reusable framework of modular off-the-shelf components. This approach offers benefits related to a shared spares, chassis-space, source-code, developer-effort, and developer-expertise. The design prefers components based on modular industry-standard long-life-span interfaces, prefers industry-standard development languages, and multiple sources for each vendor purchased modular function.

Furthermore, in a long lifespan facility, we recognize that even newly replaced systems will eventually become obsolete. With electronics components the only certainty is change. However, with a reusable framework, based on industry standard interfaces, our design can accommodate a path forward. Likewise, when one project comes to the end, if we develop our skills within a framework then hopefully some of our efforts can be preserved for use in the next project.

REQUIREMENTS

Digital Signal-Processing

Within a framework, the algorithmic requirements of individual diagnostic systems must be realizable on a shared hardware platform. In a modern system, typically an FPGA (Field Programmable Gate Array) is performing most of the signal processing tasks. Within a framework, the FPGA capacity must be sufficient to meet the needs of a diverse set of systems. Likewise, with a framework we must accommodate the signal capture requirements of many different signal types.

At LANSCE machine protection functions are currently implemented in analogue circuitry responding to inhibit the beam within a maximum overall system response of 6 μS. The signal processing framework shall be capable of responding fast enough so as to not preclude implementation of advanced machine protection functions initially supplying redundant inputs within the pre-existing machine protection system.

Real-Time Software

At LANSCE, it is necessary to beam-species flavour-stamp the incoming data streams. Due to the complexities of the modal behaviours of LANSCE, this task can't be easily accomplished in hardware, and we must therefore deploy software, running on a real-time operating system, in close proximity to the digital signal processing stream. This software must receive time deterministic interrupt response from both the timing system's event receiver module and the digital signal processing system. Certainly, this is also possible sans operating system, but an operating system is specified to improve productivity and reduce maintenance costs.

HARDWARE DESIGN

We are aware of some trends. New products offer increased density and integration; what was once possible within the confines of 6U modules is increasingly possible within the confines of 3U modules. Increasingly, a processor and Ethernet interface are cost effective for integration at the module level. We can purchase off-the-

* Supported by US Depart of Energy contract DE-AC52-06NA25396.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

shelf hardware intellectual property modules for user integration into the FPGA. Increasingly, we can forgo register-level hardware languages for implementing signal processing algorithms in favour of numerical languages originally used only for software development.

VITA-57 FMC - FPGA Mezzanine Card

The VITA-57 FMC industry standard, for **F**PGA interfaced **m**ezzanine **c**ards, appears to be almost tailor made for a signal processing framework. With FMC a common FPGA development module, including an FMC site, can be leveraged over multiple application-specific analogue front-ends. We can identify a range of FMC products fulfilling a range of analogue requirements, offering previously unavailable signal density, all within a modular replaceable nine by seven cm form-factor. We are optimistic for an optimized cost to performance ratio leveraged over a reasonable lifespan, based on a healthy competitive vendor ecosystem; see Fig. 3.

Furthermore, the flexibility of FMC form factor is an enabling technology for increased integration; we can purchase off-the-shelf components for integrating a waveform digitizer, signal processing, EPICS IOC, outputs to the machine protection system, and the timing system's event receiver software all integrated into a single FMC carrier module.

Chassis Backplane

With modern backplanes we have options to choose between centrally switched (modules-to-switch) fabrics, and meshed (modules-to-modules) fabrics. Considering the natural tendency for design evolution inherent to experimental controls, perhaps centrally switched architectures are favoured over mesh fabrics for their flexibility. We can prefer slot agnostic signal processing modules, and also to accommodate evolving module throughput demands on the fabric. Also, centrally packet-switched backplanes offer reduced impact of communication topologies on software in contrast to mesh fabrics and legacy commander-slave backplanes. See Fig. 1.

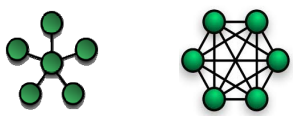


Figure 1: Central Switch (left) vs Mesh (right) Topology

The increased throughput available in modern backplane protocols is a secondary benefit; demand for higher levels of throughput occur when signal processing algorithms are pipelined together through the backplane, and when the software implementations are centralized. Also, with a real-time framework, additional contingency capacity is needed. Our timing Event Receiver Module, and industrial IO modules, require the PCI Express variant of packet switching. We observe also that PCI drivers typically require only small modifications before they are reused with PCI Express. Ethernet on the backplane is also desirable when there are increasing numbers of CPUs within a chassis, to simplify cabling. Proper use of backplanes with Rear Transition IO (RTIO) can result in simplified maintenance, improved cabling modularity, and reduced cable entropy.

In Fig. 2 is a brief summary of our backplane standard selection considerations. It can be observed that FMC carrier options are limited in VME, Compact PCI, and µTCA dot four. Both VPX and µTCA suffer from limited availability of generic IO modules; this issue is typically addressed via PMC or XMC mezzanine carrier modules, but closer inspection reveals that RTIO for such carriers isn't well standardized in µTCA. This was a particularly strong consideration due to our event receiver timing module only being available in these backplanes in PMC. RTIO *is* standardized under µTCA dot four, but vendor activity is scarce, particularly for PMC carriers that route signals to the back of the chassis.

	Stds Org	Generic IO	Activity	EVR Timing	FMC	Rear Transition IO	Packet Switching Backplane
VME	VITA	+	+	+	rare	+	VXS, rare
Compact PCI	PCMG	+	+	+	rare	+	Various, fragmented
µTCA	PCMG	-		PMC*	+	*	+
µTCA.4	PCMG		-	PMC*	-	+	+
VPX	VITA	--	+	PMC	+	+	+

Figure 2: Backplane Decision Summary.

We selected VPX; among its positives were high levels of signal processing vendor activity, well standardized high-density rear-transition IO, the VPX 3U form factor's match to our signal density requirements, and a modern backplane architecture appropriate for green-field signal processing systems. The primary negative with VPX is the expense of its proprietary high-density impedance-controlled module connectors. We also observe that, currently, there is diminished signal-processing vendor activity in 6U compared to 3U VPX.

Backplane and Chassis Architecture

We selected a centrally-switched 3U Open VPX backplane, see Fig. 4. This backplane supplies a single hybrid, PCI Express and Gb Ethernet, switch slot. Due to higher per-slot cost with this backplane, we opted also for a chassis design incorporating a 2nd auxiliary 6U Compact PCI backplane transparently integrated by the VPX PCIe switch, and also modular N+1 redundant 3U Compact PCI power supplies. A front-to-rear internal cable raceway at the top of the chassis was also specified.

Our VPX physical slot-keying design designates only two slot-profiles. Key position two prevents installation of switches into signal-processing slots, and the converse. Typically, module key-blocks for position one will be installed making no distinction among signal-processing slots, with selective keys required only by applications daisy-chaining multi-stage DSP algorithms between modules on the Open-VPX expansion-plane due to insufficient FPGA or FMC capacity; see Fig. 3.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

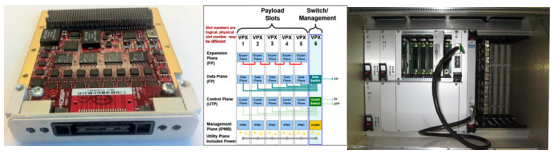


Figure 3: FMC module 125Msps, 14 bits, 16 Channels, 9 cm x 7 cm; Open VPX Backplane Choice; BKP3-CEN06-15.2.2-2, Chassis

Signal Processing Framework

Our design is based on the FPGA vendor's Avalon Streaming Interface allowing component-based modular stages of the signal processing algorithm to be connected together using the FPGA vendor's system integration tool. We have freedom to create algorithm components from high level languages such as MATLAB, create components from register level hardware description languages, or to acquire production ready components. We will also provide facilities to remotely upgrade the algorithms running in our production chassis, over the network. The analogue input is accessed through the same Avalon Streaming Interface. This facilitates algorithm reuse with more than one physical FMC board. See Fig. 4.

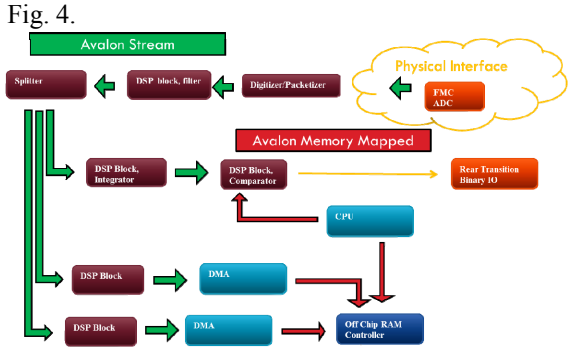


Figure 4: Component-Based Signal Processing

Generalized Filtering Algorithm

The dynamic range of our measurements in the presence of noise can be extended by averaging elements together in an oversampled input waveform; a mathematically optimal way to do this is to employ a Kalman Filter[1]. With a Kalman Filter we combine the model predicted state of the system together with a new measurement using a mathematically optimum weighted average. For example, with a current monitor toroid sensor we have at least two internal states for the system which can't be observed perfectly in the sensor data due to the presence of noise, and also due to imperfections in our estimates for component values of the sensor. These state variables are the voltage across the capacitance seen by the toroid transformer's primary, and the toroid's current. Mathematically optimum weights for these averages can be computed based on the statistical covariance of the system, where the covariance is a measure of the uncertainty of model predicted values of the system's state, sensor noise, and the input noise. Optimum behaviour of the algorithm is contingent on the noise, and other imperfections in our observation of the system varying with a Gaussian random-variable statistical distribution. Fortunately, Gaussian distributions are common with physical systems. The Kalman filter is a general approach, appropriate for a framework. Figure 5 shows some output from the current state of the MATLAB modelling for our filtering algorithm.

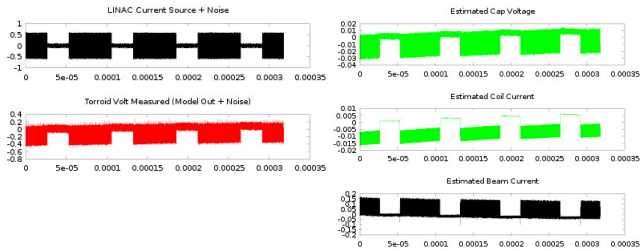


Figure 5: Current Monitor Estimator Simulation.

Furthermore, the framework is optimized to facilitate development of advanced machine protection signal–processing functionality. Currently, these functions are implemented within analogue systems, but this hardware is aging and in select situations becoming obsolete in terms of man-power maintenance costs. The legacy analogue-based radiation detector loss-monitor system compares sensor data against a fault threshold to determine the state of a fast-protect input, and the beam current transmission monitor system fast-protect input trips when an integrated beam current difference for two physically adjacent current sensors exceeds a threshold. We can observe that overall system reliability, in terms of detecting faults, might only improve when providing additional redundant fast-protect inputs from flexibly developed, and therefore potentially more advanced, digital-based algorithms adapting well to our needs, as the capabilities of LANSCE evolves.

Processor

Recently there are new options to integrate soft-core CPU intellectual property modules into the FPGA. Certainly, ASIC CPUs run at higher clock rates, but they are less agile. With a soft-core CPU we can directly connect our processor to the incoming signal-processing stream within the FPGA. Figure 6 shows our processor design including snap-in interfaces for FPGA boundary specific interface components. This approach allows for a reusable design which can be adapted to different boards and signal-processing applications. This type of memory-mapped master-slave integrated architecture is fairly easy to produce using the vendor's system builder tools. Impact on FPGA compile delays can be reduced when the CPU is isolated in an independent design partition from the off-chip interfacing FPGA intellectual property modules (IP), and or the signal-processing application.

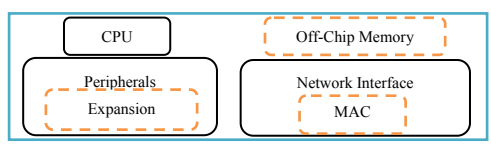


Figure 6: Processor Design.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

FPGA Build System

We have created a gnu-make-based build system for our FPGA development environment. This approach has some benefits compared to working solely in the vendor's graphical-user-interface based tools; we separate out the source code form the large number of generated files produced during a build, we can easily perform a clean rebuild from source so that pin assignment script detritus does not accumulate in the vendor's project database, and we can easily orchestrate an unattended rebuild including automation of any pin-assignment scripts that the vendor's intellectual property (IP) modules might require. A build system also results in a more consistent installation of the signal processing applications, and this hopefully facilitates developer portability between projects.

SOFTWARE DESIGN

Network Architecture

The LANSCE system is based on EPICS, and we can simplify our network architecture when embedding the EPICS IOC directly within our diagnostics systems. Our perspective is that systems based on protocol-converting-gateways are more difficult for on-call personnel to diagnose. This is especially true if they are closed software systems with no mechanisms to remotely observe their internal health within the control room. In contrast, when the IOC is embedded we have new opportunities to observe the internal state of our algorithms, and adjust parameters while the systems are running. Operational experience is proving that these capabilities may be essential for converging to the highest levels of quality. The maximum Ethernet throughput for the FMC carrier based IOC is currently at around 120 Mbps. Considering the number of this type of system deployed times this figure, we conclude that it is essential to govern the rate of EPICS subscription updates for each species beam, synchronized by the timing system.

Real-Time Software

We use the RTEMS real time operating system with some local enhancements to its support for the Altera Nios II Soft-Core Processor. From our perspective, a primary benefit of RTEMS is capabilities for a minimalist installation.

Board-Support Software

With a soft-core CPU we have options to develop a virtual board support package (BSP). With a conventional BSP custom code is required for each hardware design. In contrast, with a virtual BSP more of the BSP is in a reusable library of driver components consistent with the library of IP used to instantiate the hardware. If the processor is instantiated into a new board, the BSP software auto-configures itself off of the outputs from the vendor's system builder tools. For example, with RTEMS we first generate the vendor's BSP, extract the vendor's configuration information, and then auto-generate the necessary source files used to configure the RTEMS BSP. All of these steps are automated in the gnu-autotools-based build for the NIOS II RTEMS BSP. This approach has enable instantiation of the same RTEMS BSP into the processor design for the signal processing framework, and also a different processor design currently being deployed into the low level RF control boards. We have implemented an RTEMS task-level network-accessible GNU Debugger stub, which is perceived to be essential for properly converging to the highest levels of quality with this type of system.

COST VERSUS BENEFITS

When creating a new system there can be two opposite ends of the spectrum approaches; we can buy an integrated system from a single vendor, or at the other end of the spectrum we can assemble a system from parts, obtained from multiple vendors. The system assembled from parts certainly costs more in terms of in-house development time, but we are hopefully realizing some benefits when we are better protected from component obsolesce, and lack of design malleability. Hopefully our components are easily upgraded because they are based on industry standards, and healthy competitive vendor ecosystems. Since we have expertise in-house, then we can reprogram components that do not meet original design specification or that need flexibility to meet future requirements at LANSCE. However, for this type of system to be successful we must amortize its cost over multiple designs and installations. This requires a framework so that we can leverage our skills and components.

CONCLUSIONS

We are compelled to replace certain aging electronics systems at LANSCE. We have chosen the FMC and VPX industry standards as a solid technical basis for a healthy multi-vendor ecosystem, appropriate for a signal processing framework. We have implemented a component-based signal processing architecture, appropriate for a signal processing framework. We have embedded the EPICS IOC at the lowest level permitting the internals of our algorithms to be directly monitored and adjusted while they are in production use. It costs more, due to additional development time, to assemble this type of system from parts, but the benefits are hopefully visible when the system is malleable to our future needs while also reusable over a range of systems, projects, and facilities within a framework.

REFERENCES

[1] K. Shanmugan, A. Breipohl, Random Signals; Detection, Estimation, and Data Analysis, 1988

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MESSAGE SIGNALLED INTERRUPTS IN MIXED-MASTER CONTROL

W. W. Terpstra, GSI, Darmstadt, Germany
M. Kreider, GSI, Darmstadt, Germany

Abstract

Timing Receivers in the FAIR control system are a complex composition of multiple bus-connected components. The bus is composed of Wishbone crossbars which connect master devices to their controlled slaves. These crossbars are in turn connected in master-slave relationships forming a DAG where source nodes are masters, interior nodes are crossbars, and terminal nodes are slaves. In current designs, masters may be found at multiple levels in the composed bus. Bus masters range from embedded microcontrollers to bridges from PCIe, VME, USB, or the network.

In such a system, delivery of interrupts from controlled slaves to masters is non-trivial. The master may reside multiple levels up the hierarchy. In the case of network control, the master may be kilometres of fibre away. Our approach is to use message signalled interrupts (MSI). This is especially important as a particular slave may be controlled by different masters depending on the use-case. MSI allows the routing of interrupts via the same topology used in master-slave control. This paper explores the benefits, disadvantages, and challenges uncovered by our current implementation.

INTRODUCTION

To coordinate accelerator activities at GSI, we need a hard real-time control system. The hard real-time components of the control system are implemented using logic chips whose wiring can be reprogrammed in the field, called FPGAs. This makes it possible for our existing hardware to accommodate some of the changing requirements that arise as physicists revise their research goals. It also means that we have a great deal of flexibility in terms of how we build and connect the hard real-time components inside the FPGA.

For timing receivers, we have standardized on the pipelined Wishbone B.4 bus standard [1]. By using this open standard inside our FPGAs, we can interface components of our design more readily with components from CERN and the open hardware community at large. Our experience thus far, about six years, has shown that Wishbone strikes a good balance between simplicity and flexibility. Simple components can easily speak Wishbone, but the bus scales well enough that today our FPGA designs include nearly fifty distinct Wishbone-connected components.

In a complex system like ours, many components must communicate with each other. Most often, this communication flows from logic components, which direct the planned accelerator behaviour, to physical interfaces, which perform the external signalling to control magnets, measurement equipment, and displays. However, sometimes this communication flow must be reversed. If a magnet power supply encounters an error condition, it must be able to notify the components which control the system. The Wishbone standard does not include such a mechanism, and a traditional interrupt-based system is inadequate for our needs.

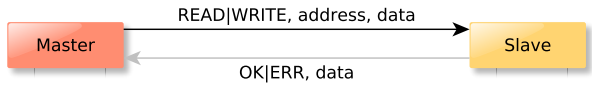


Figure 1: Wishbone connects one master to one slave. Masters initiate requests. Slaves respond with success or failure.

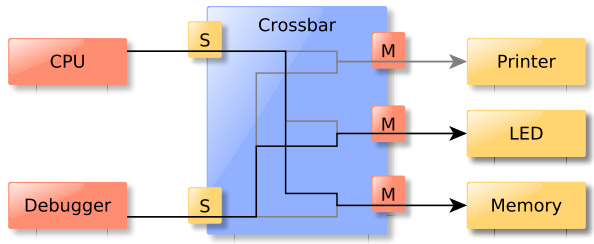


Figure 2: A crossbar providing two slave ports and three master ports. It internally connects each master to one slave.

This paper will explain how we have extended Wishbone to include a notification facility, based on message signalled interrupts (MSI). Our approach essentially mirrors the Wishbone bus to allow messages to flow bidirectionally throughout the design. While this approach carries an increased interconnect cost, it is very flexible and requires only a minor revision to the Self-Describing Bus standard [2].

WISHBONE BASICS

Wishbone is a master-slave point-to-point bus protocol. This means that it describes how to connect a single master to a single slave, as shown in Figure 1. All communication is initiated by the master, which sends bus operations (reads or writes) to the slave. The slave then executes the requested operation and reports any resulting data and the success or failure of the operation.

Bus operations include an address. Typically, the slave uses the address to determine what to do with the operation. For a slave that implements memory, the address is simply the location in its table where the value should be read or written. However, for most slaves, the address acts as an indication of the type of command. Writing to a particular address might cause the magnet to instantaneously change its field strength. Writing to a different address might cause the magnet to begin slowly ramping the field from one strength to another. Reading from one address might report the current field strength, while another address could report current or voltage. In any case, Wishbone slaves have a fixed number of addresses that a master can read or write.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

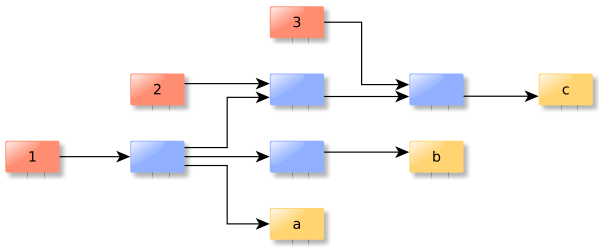


Figure 3: Real systems often include many nested crossbars.

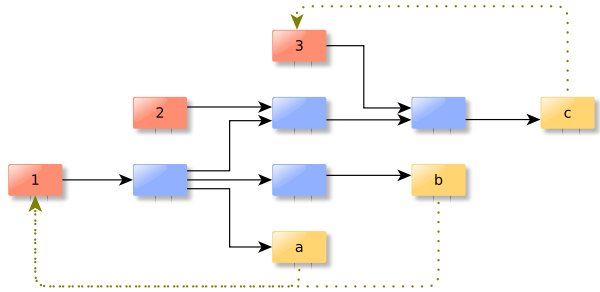


Figure 4: Interrupt lines connect a slave to one fixed master.

While a point-to-point protocol is already useful for combining two components, sophisticated systems generally need to connect more than two components together. This is the role of a Wishbone crossbar switch, as illustrated in Figure 2. The crossbar implements several slave ports, suitable for connecting Wishbone masters, and several master ports, suitable for connecting slaves. Inside the crossbar are wires which allow each master to send messages to each slave.

To decide which slave to connect to a master, the crossbar uses the Wishbone address. Recall that each slave has a fixed number of addresses. The crossbar essentially concatenates the slave addresses together to create one giant slave device. The address in the first bus operation from a master is used by the crossbar to decide which slave to connect to the master. Thereafter, the master stays connected to that slave until their communication is complete. If a different master wants to access the same slave, it must wait. However, as shown in Figure 2, two different masters can communicate with two different slaves, without waiting.

A single crossbar is already enough to build quite complicated systems. However, as the design grows in complexity, multiple crossbars start to appear. Consider, for example, a host motherboard with slots for inserting addon cards. The inserted cards might contain several slaves, connected together by a card crossbar. The motherboard hosting the slots likely also has some masters and slaves, connected by a host crossbar. When the card is inserted, the masters in the hosting motherboard would like to access the slaves in the addon card. This is achieved by connecting a master port of the host crossbar to a slave port of the card crossbar. Now the host masters have access to the host slaves and, via the crossbar-crossbar connection, the card slaves.

Beyond the physical need for nested crossbars in the previous example, nesting crossbars is also a useful organizational tool. For example, CERN provides a small White Rabbit (WR) crossbar with masters and slaves that cooperate to precisely synchronize clocks [3]. We use this WR design as a component in our larger system. Treating their entire WR crossbar conceptually as a single slave component on our larger bus allows us to cleanly separate our code from theirs.

In any case, once you have multiple nested crossbars, you end up with a Directed Acyclic Graph (DAG) like in Figure 3. As shown, we have masters 1, 2, and 3 connected to slaves *a*, *b*, and *c*. Of course, a real system would probably include far more masters and slaves on each crossbar, but we keep the example simple for illustration. Notice that masters 2 and 3 only have access to slave *c*, whereas master 1 has access to all three slaves. This is typical for a complex Wishbone bus. When well organized, the restricted access of masters to the bus helps us reason about the system as a whole.

THE INADEQUACY OF INTERRUPTS

As already explained, only masters initiate the exchange of messages in the Wishbone bus. If a slave has information for a master, it must wait for the master to read it. Unfortunately, this model is too simplistic. Consider a slave which rarely has anything to tell its master, but sometimes it has an urgent message that must be delivered immediately. In this case, the master must wastefully read the slave repeatedly, forever. This is not always feasible, so we need a way for slaves to signal to masters that they need service.

The classic approach is hardwired interrupt lines, as illustrated in Figure 4. In this model, the slave has a single wire (the interrupt line) with which it can signal its need for service. During normal operation, when it has nothing to be read out, it leaves the interrupt deasserted. Once it has something to read, it asserts the interrupt line. This informs the master that it must immediately read the slave. Once the data has been read out, the slave deasserts the interrupt.

The advantage of this approach is that it is very inexpensive. It only requires a single wire from slave to master. In an FPGA, this benefit is not really important, but on a PCB this can be a decisive factor. However, while cheap, this approach has serious shortcomings.

Firstly, an interrupt line typically connects one slave to one master. In Figure 4, slave *c* was connected to master 3. It might well be that master 3 is the only master which ever controls slave *c*, in which case this is fine. However, in our designs, it is often the case that a slave might be controlled via different masters depending on the deployment. For example, we might control a timing receiver via a PCIe slot in a PC. The same card might also be controlled over the network or via USB. Depending on how the card is connected, the interrupts need to go to different masters. This is not possible in a simple interrupt line scheme.

Secondly, there are usually only a fixed number of interrupt lines. In Figure 4, both slaves *a* and *b* are connected to the same interrupt line of master 1. When the master sees

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

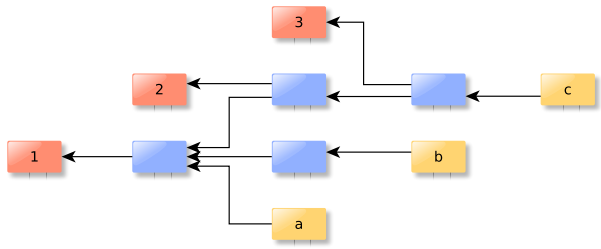


Figure 5: MSI reverses Wishbone, allowing each slave to send notifications to any of the masters which can control it.

an asserted interrupt line, he must check all slaves attached to that interrupt line for the source of the interrupt. If a master does not have support for the slave which asserted an interrupt, it is also unable to service the interrupt. Thus, the interrupt stays asserted and all other slaves sharing the interrupt line are now unable to signal the master.

Thirdly, a single slave might have several reasons it asserts an interrupt. As the interrupt line is only a single bit, it cannot indicate the reason. Different reasons might have different priorities. The master cannot know how urgent the interrupt was until after it has checked all possible reasons for the interrupt. In a bus with forty slaves, each with multiple reasons for an interrupt, it could take a long time to determine that the interrupt was not urgent after all.

Finally, interrupt lines break composability. We want to freely nest crossbars for code reuse and clarity. Interrupt lines cannot be plugged together in the same way we connect crossbars. We would like to plug two crossbars together and suddenly have the upper masters be granted access to the nested slaves. However, the interrupt lines of those slaves are probably already connected to the masters at their own crossbar. This means that the new outer masters may be unable to use the slaves properly.

MESSAGE SIGNALLED INTERRUPTS

One alternative to interrupt lines is Message Signalled Interrupts (MSI). In this scheme, we allow slaves to send messages towards masters. In a very real sense, this flips the role of masters and slaves; see Figure 5. Wherever there was a crossbar, it is now the slaves which initiate the message and the masters which receive them. In our system design, we implement this using a second Wishbone bus.

On the MSI Wishbone bus, masters are now essentially Wishbone slaves. For clarity, we will always denote master versus slave from the point-of-view of the regular Wishbone bus. Since masters now receive MSI Wishbone messages, they should provide a small address space. Each address in this space is analogous to a distinct interrupt line. However, 16 address bits suffice for 64K interrupt lines, enough for most uses. The MSI crossbars route messages from slaves to masters just like the normal crossbars from master to slave.

For each type of notification they will send, slaves should provide a configuration register to set the MSI target address. They should also provide an enable register that indicates if that type of notification should be sent at all. When a master wants to receive MSIs, he puts his own address into this target address register and enables the notification. When the slave wants to notify the master, if this notification is enabled, he sends a MSI out the MSI Wishbone bus to the configured MSI target address.

The data field of the MSI can be use for parameters within a notification. For example, consider a slave with a FIFO. It should provide three types of MSI notifications, each with its own configurable MSI target address: empty, full, and level. The empty notification has either a 0 or 1 in its data field and is sent when the FIFO changes between empty and not empty. Likewise, the full notification sends a 0 or 1. If enabled, the fill notification would be sent whenever the level changes, indicating the new level in the data field.

Compared to interrupt lines, this approach resolves all of our concerns. Firstly, any master which can control a slave can receive MSIs from that slave; the bus topology has all arrows between them reversed. Secondly, the masters have many thousands of MSI addresses, and can instruct distinct slaves to use distinct MSI addresses. Thus, a master immediately knows which slave sent the message based on the target address. Thirdly, if a master wants to distinguish messages by priority, he can provide different queues on different addresses. Then he instructs a slave to send urgent messages to one queue and unimportant messages to another queue. For example, he could connect a slave FIFO's full notification to an urgent MSI queue, disable the empty notification, and connect the fill notification to an unimportant queue. Finally, the MSI approach is just as composable as Wishbone, because it *is* Wishbone.

Clearly, this approach could double the interconnect cost in the worst-case. However, slaves which do not generate MSIs can be omitted, as can masters which do not receive MSIs. Furthermore, MSI bus logic tends to be simpler on both sides, because only writes are allowed. Nevertheless, there is an area price to be paid for this approach. We consider this an acceptable trade-off inside an FPGA.

As an optimization, when Wishbone leaves the FPGA, we sometimes use interrupt lines behind the scenes. In this case, a cheap interrupt-based external protocol is used to tunnel Wishbone and MSI messages. However, this use of interrupt lines is invisible to Wishbone masters and slaves. It is an implementation detail of the Wishbone bridge.

ADDRESSING AND SELF-DESCRIPTION

One detail we have ignored till now in Wishbone is how crossbars compose addresses recursively. The general idea is that a Wishbone slave should ignore the high bits it does not use itself. Thus, the GPIO in Figure 6 only pays attention to the low 5 address bits. The crossbar which connects it and the LED, inspects the 6th address bit to decide if an operation goes to the LED or the GPIO. This means that when accessed via this crossbar, the GPIO register 0x4 has address 0x24 outside the crossbar. Similarly, from the CPU's point-of-view, that GPIO register has address 0x124.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

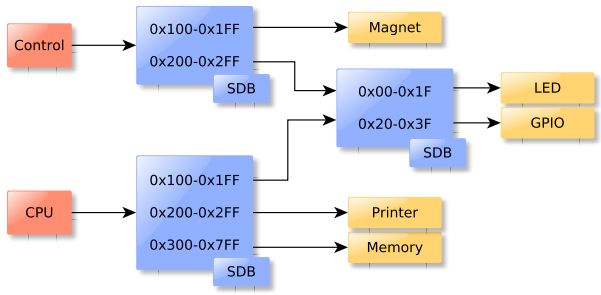


Figure 6: Wishbone routing is recursive. Slaves and crossbars ignore the high address bits matched earlier on the path.

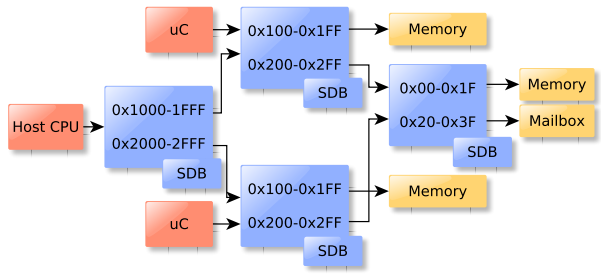


Figure 7: With shared bus resources, a master (Host CPU) sometimes sees a slave (Mailbox) twice under two addresses.

However, this compositional approach leads to something a bit surprising. That same GPIO register has address 0x224 from the point-of-view of the Control master. The correct way to understand this is that **an address describes a path**, not a destination. The address 0x224 tells you how to get from the Control master to register 0x4 of the GPIO.

If addresses do not describe destinations, how do we know what is on the bus? This answer is provided by Self-Describing Bus (SDB) records [2]. Each crossbar includes a small table that describes what is attached to it at each address range. In a sense, SDB describes the outgoing arrows on each crossbar. A Wishbone master recursively reads these SDB records, exploring all paths through the bus and composing their associated addresses.

When a master finds two identical devices via SDB exploration, it presumably wants to tell them apart. This is the job of the device driver. SDB will tell you that there are two paths to LED devices. The driver must query the LED slaves to discover that one LED slave should bink to indicate network activity and the other indicates power.

The final wrinkle, which also affects MSI, can be seen in Figure 7. Here, we have two microcontrollers (μC), each with their own private memory and a shared mailbox. However, there is a supervisor Host CPU which has access to both of their private memories. Diamond patterns like this become unavoidable in larger designs. In this example, it is clear that the Host has two paths to the mailbox. Armed with our refined understanding of addresses, it is obvious that this means that mailbox register 0x8 has two addresses from the point-of-view of the Host, 0x1228 and 0x2228. There are two paths, and so two addresses. As with identical LEDs, it is the mailbox driver's job to recognize that the two addresses are just different paths to the same device.

FINDING YOUR OWN NAME

Since the MSI bus is just the normal Wishbone bus inverted, it should be no surprise that masters have different addresses when viewed from different slaves. Again, addresses describe a path, not a destination.

When a master wants to receive an MSI from a slave, he must configure the slave's MSI target address register with his own address. However, the master does not have a unique address. He must use the address which describes the path from the slave back to himself.

We have seen that masters recursively scan SDB records to locate all paths through the bus. As the master explores paths, he constructs the corresponding path addresses top-down. However, on the MSI bus, the master is the target. For this reason, a master which scans SDB should simultaneously construct the backwards path address from that location on the bus to himself. This MSI path address is constructed bottom-up. For each path found during SDB bus enumeration, the master has a triple of information: the SDB meta-data describing the terminal slave, the address describing the path to that slave, and the address describing the path from that slave back to the master.

Unfortunately, to construct the bottom-up address path from slaves requires information about the masters in SDB. These new records have not yet been standardized.

CONCLUSION

Wishbone scales well from simple to very complicated bus systems. To support slave-to-master messaging in Wishbone, we take a MSI approach. This gives us great flexibility at the cost of increased interconnect area. In particular, we can combine crossbars together hierarchically while retaining the ability to deliver notifications from slaves to all the masters which might control them. To realize our design, it is necessary to revise SDB [2] so that masters may discover their own addresses when configuring slaves for MSI.

REFERENCES

[1] R. Herveille. WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores - Revision B.4, 2010. URL `http://opencores.org/opencores,wishbone`.

[2] A. Rubini, W. W. Terpstra, and M. Vanga. Self-Describing Bus (SDB) Specification for Logic Cores - Version 1.1, April 2013. URL `http://www.ohwr.org/projects/sdb`.

[3] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, and G. Gaderer. White Rabbit: Sub-Nanosecond Timing Distribution over Ethernet. In *International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 2009.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE EVOLUTION OF THE ALICE DETECTOR CONTROL SYSTEM

P. Chochula[1], A. Augustinus[1], P. M. Bond[1], A. Kurepin[1,2], M.Lechman[1,3], O. Pinazza[1,4]
[1]CERN, Geneva, Switzerland
[2]INR RAS, Moscow, Russia
[3]Institute of Physics, Slovak Academy of Sciences, Bratislava, Slovakia
[4]INFN Bologna, Italy

Abstract

The ALICE Detector Control System has provided its service since 2007. Its operation in the past years proved that the initial design of the system fulfilled all expectations and allowed the evolution of the detectors and operational requirements to follow. In order to minimize the impact of the human factor, many procedures have been optimized and new tools have been introduced in order to allow the operator to supervise about 1 000 000 parameters from a single console. In parallel with the preparation for new runs after the LHC shutdown a prototyping for system extensions which shall be ready in 2018 has started. New detectors will require new approaches to their control and configuration. The conditions data, currently collected after each run, will be provided continuously to a farm containing 100 000 CPU cores and tens of PB of storage. In this paper the DCS design, deployed technologies, and experience gained during the 7 years of operation will be described and the initial assumptions with the current setup will be compared. The current status of the developments for the upgraded system, which will be put into operation in less than 3 years from now, will also be described.

THE ALICE DCS DESIGN

The central ALICE detector consists of 19 subdetectors, built with different detection technologies and with largely different operational requirements, all supervised by a single operator. The architecture of ALICE Detector Control System (DCS) is based on standards adopted by the LHC experiments at CERN. The commercial SCADA system WINCC OA, extended by CERN JCOP and ALICE software frameworks, is configured as a large distributed system running on about 100 servers [1]. To guarantee the autonomous operation of each subdetector, the central distributed system is segmented into subdetector systems, each allowing for stable subdetector operation in isolation from the other subdetectors.

Wherever possible, the ALICE DCS is based on commercial hardware and software standards. OPC servers handle the communication between the WINCC OA and the controls devices. Nonstandard devices, like detector frontend electronics modules, are interfaced with WINCC OA using ALICE FED standard, a complex client-server mechanism based on the DIM communication protocol [2].

All controls tasks of the ALICE experiment could be fully satisfied with the pure WINCC OA system, however its operation would require a deep knowledge of many technical details. Using the SMI++ framework [3], installed on each WINCC OA system, the operation of controlled components is modelled as a finite state machine with well-defined behaviour. Figure 1 shows the schematics of main DCS components.

The various dependencies, like the need to configure a channel before it can be turned on, are encoded in the SMI++ logic. Using the mechanisms provided in the SMI++ framework,
ALICE is represented as a hierarchical tree, with the central DCS placed on top of the pyramid and individual channels on its bottom. Commands sent from the top objects are propagated to the child nodes: the subdetectors, subsystems (high voltage, cooling, frontend electronics), devices and their channels. Each object reports its current state to its parent object. Any part of the tree can be excluded from the hierarchy and operate independently.

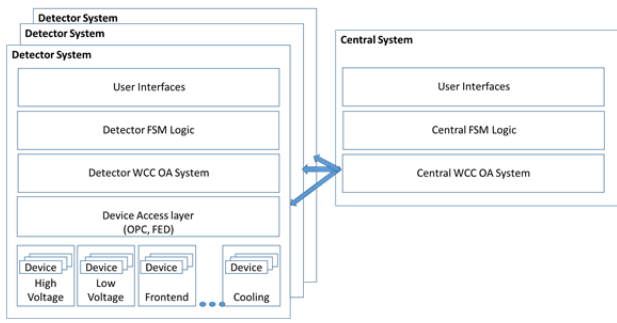


Figure 1: Architecture of the ALICE DCS.

OPERATIONAL EXPERIENCE

During eight years of ALICE operation, the initial architecture and system implementation proved their reliability and robustness. The DCS provided stable 24/7 services with small interruptions required mainly for the infrastructure modifications (cooling and ventilation upgrades, reorganization of the computer racks…). Even during the service breaks, the core systems related to safety remain operational.

With additions of new detector modules, the DCS is being continuously extended, profiting from scalability of the core architecture. In few cases new computers needed

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

to be added, to balance the load on busy subsystems. So far all detector requirements could be satisfied with the existing systems and tools, without the need for major changes in the original architecture. All system upgrades and patches followed the standard evolution of the hardware, software and operating systems.

Main modifications of the DCS are related to the experiment performance requirement. Procedures and operation tools are being continuously adjusted in order to implement data taking requirements and reflect operational experience. The original approach expected that the operator would interact directly with the state machines and move detectors to desired states as a response to operational conditions. This method, successfully implemented in the previous generations of high-energy physics experiments, soon reached its limits, due to the complexity of the controlled equipment and cross-dependencies between the subdetectors, infrastructure and external conditions.

For example, as a reaction to beam mode changes, the detectors need to adjust their settings. To protect gas detectors from damage, the high-voltage channels need to be ramped down to an intermediate state, assuring that an accidentally deposited charge will not damage the readout channels. Simple FSM mechanism does not assure, that all sensitive devices receive the command. Certain modules might be for example excluded from the central hierarchy for expert intervention and therefore ignore the commands sent by the central operator. A mechanism independent on FSM had to be implemented. Dedicated safety scripts regularly verify the status of physical hardware channels and calculate the safety condition of each detector based on readout values. Using this low-level bypass mechanism, the operator receives an overview of the experiment status based on the physical values instead of the logical state calculated by the FSM. The high-level tools allows to force the safety related commands even to modules, which are not under direct controls of the central operator.

The routine operations depend on a number of external conditions, which need to be verified before any command is sent to the subdetectors. For example a period of stable beam collisions is preceded by a complex procedure during which the particle bunches are injected into the LHC, accelerated, and adjusted. Each phase of this procedure represents a different set of risks for the detectors. While certain detectors can be set to nominal operational values during the whole procedure, some of them must remain at reduced voltage settings until the injection is completed. Operation of several subdetectors has to be postponed until the end of the particle acceleration phase and finally the operation of remaining detectors can be restored only after a phase of stable beam collisions has been reached. The status of LHC is not the only factor defining the actions to be executed by the operator. For example high radiation levels prevent the detector from nominal operations even if stable beam collisions were established.

All the operational rules are implemented in the high-level operational procedures. Instead of controlling the subdetectors directly, the operator only issues a command to reach a desired configuration. The role of the high level procedures is then to execute this request by controlling the FSM directly and taking all cross-dependencies and external conditions into account. Decoupling the operator from the low-level controls tasks significantly reduced the number of human errors as well as the time required for an execution of most complex actions.

The DCS operator has direct access to about 1 million of controlled and monitored parameters. The visualization and access to these values is established through graphical user panels, which are organized hierarchically on a single operator console. The user interface allows for easy navigation to any of the panels, using only single user interface.

The ALICE DCS is built in collaboration between the central coordination team and detector experts working at institutes remote to CERN. The information provided on the panels is largely biased by the deep expertise of the developers and is not necessary intuitive to operators, who have typically no prior experience with control systems. The central team has analysed the individual subdetector requirements and provided high-level tools and interfaces, which are presented to the operator. The tools invoke detector actions as needed.

The central team has issued a set of implementation rules and guidelines based on the experience with the previous generations of the control systems and supervised the developments carried on by detector groups. With growing experience as well as with the increasing complexity of the experiment requirements, the guidelines are being continuously refined. Big efforts are invested into implementing the new standards with a focus on uniformity across the whole DCS. For example, all popup messages aimed to bring the focus of the operator to a certain local problem were replaced with standardized alerts, displayed on a single screen. Each alert is accompanied by a set of instructions, which will guide the operator in the troubleshooting process.

The central DCS operator plays a key role in the operation. Originally, subdetector experts operated the systems and the central operator coordinated their actions. Currently the DCS operator is typically the only person in the ALICE run control centre linked to the controls tasks. To assure continuous 24/7 operation of the experiment, more than 150 operators were trained in 2015. The training procedure has evolved from a simple introductory presentation accompanied by hands-on experience to a formal process. It includes the lecture, hands on session using a simulator, compulsory training shifts during which the trainee operates the experiment under the supervision of an experienced shifter and then a final exam.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE ALICE O2 PROJECT

A major experiment update is foreseen for the third phase of the LHC operation, starting in 2019. New detectors will be installed in ALICE and readout of detectors will be modernized. The present data acquisition mechanism based on triggered readout will be replaced by continuous data taking. The detectors will be connected through Alice readout links to a farm of 250 servers, the First Line Processors (FLP), which will assemble the detector information into consistent data sets. The acquired data will flow to 1500 Event Processing Nodes (EPN), installed in ALICE, performing the full data processing. This new approach merges the online and offline roles into one system, named O2.

The farms need to digest 1.09 TB/s of detector data, producing about 40 PB of data for physics analysis each year.

The new detectors will be integrated into the existing DCS using the well-established standards and procedures. Thanks to the scalability of the DCS architecture, this task does not present a major challenge. There are, however, two areas requiring new approaches: the conditions data flow and frontend electronics control.

The Conditions Data Flow in O2

The present DCS archives all acquired values in a ORACLE database, independently from data acquisition. After each period of data taking, called run, the DCS collects all archived conditions parameters and sends them to offline processing.

Depending on the duration of a run, the delay between the actual readout of a parameter and its transfer to offline can be in the order of several hours. This situation dramatically changes with introduction of the O2. The data is sent to FLPs in 20 ms time frames. Each time frame must be accompanied by a full set of condition parameters. The current estimates suggest, that about 100 000 parameters will need to be inserted to each O2 data frame. This requirement is of course out of scope of the DCS, where parameter are updated at a rate typically lower than 1 Hz. The present mechanism expecting the data first to be archived in ORACLE would add another delay in the processing. A new mechanism, named ADAPOS (Alice DAtapoint Server) is being developed.

A specialized software module, the Data Finder and Publisher (DFP), will equip all WINCC OA systems. Its role is to collect all conditions parameters provided by its host system and publish them to subscribers.

The data collector is a client part of ADAPOS. It connects to all DFPs and reads published values. Alternatively, the data collector can retrieve also data from the DCS ORACLE database as shown in Figure 2. The received parameters are passed to the module, which is maintaining a control process image. This memory resident object keeps information about all monitored parameters. The data collectors refreshes the process image each time when a new value arrives. The whole process image is mirrored on a dedicated DCS FLP. Synchronization between the ADAPOS and FLP is handled using a dedicated data transfer channel. The FLP will then retransmit the process image to the EPN, synchronized with the 20 ms data frames.

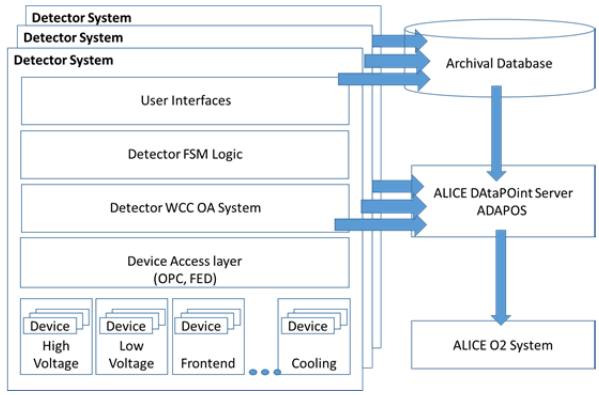


Figure 2: Architecture of the DCS-O2 interface.

The Frontend Control

Part of the DCS information is produced by the detector frontend modules. The DCS conditions data is inserted into dedicated packets, transmitted along with detector data to FLPs. The firmware of the receiver cards strips the DCS information off the data stream and publishes it to the DCS clients implemented in WINCC OA. Each client subscribes to a required subset of published values without the need to know the details on physical configuration of the frontends and FLPs. A common name service will handle the redirections of subscription requests. One of the already existing technologies supporting this mode of operation is DIM. First prototypes based on DIM are able to carry this task with sufficient contingency margins.

The configuration sent from DCS to the frontend modules is following a similar path – the WINCC OA based client sends a command to a server implemented on the FLP, which will insert it to a data frames sent to the frontend modules. This task requires synchronisation at many levels – the command may not interfere with regular data taking, conflicts sent by concurrent commands shall be resolved before the electronics is reconfigured, acknowledge signals must be sent after the commands are executed, etc. To handle this complexity, a concept of ALICE FED [2], known from the current operation, is being extended. Each detector will be equipped with one or more FRontEnd Device servers (FRED), which will listen to commands sent by WINCC systems and transmit them to the FLPs with the physical access to target devices. The responses will be then transmitted back to requestors either using the standard conditions data channels, or via the FRED servers.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONCLUSIONS

The ALICE DCS followed the evolution of the experiment and provided a stable service over a period of more than eight years. Major efforts were put into automation and unification of the operational procedures. The system extensions followed the evolution of the ALICE detectors and profited largely from the scalability of the ALICE DCS architecture.

ALICE upgrade for the LHC RUN3 period puts new challenges on the DCS. The access to the frontend modules has to be redesigned and existing standards will be modified to cope with the new electronics. Major architectural change is required for the conditions data flow. Current batch processing of the conditions data, based on the values archived in the ORACLE database, will be replaced with a publishing mechanism, allowing the transmission of measured valued to the O2 facility in quasi-real time.

REFERENCES

[1] P.Chochula et al., Operational experience with the ALICE Detector Control System, Proceedings ICALEPCS 2013, San Francisco (2013)

[2] P.Chochula, L.Jirdén, A.Augustinus, Control and monitoring of the front-end electronics in ALICE, 9th Workshop on Electronics for LHC Experiments, Amsterdam 2003

[3] Franek, B, Gaspar, C: “SMI++ object oriented framework used for automation and error recovery in the LHC experiments”, J. Phys.: Conf. Ser. 219 (2010) 022031

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

STATUS OF THE CONTINUOUS MODE SCAN FOR UNDULATOR BEAMLINES AT BESSY II

A. Balzer, E. Schierle, E. Suljoti, M. Witt, HZB, Berlin, Germany
R. Follath, PSI, Villigen, Switzerland

Abstract

At the synchrotron light source BESSY II monochromator (MONO) and insertion device (ID) scans can be done synchronized in two different modes. In step mode MONO and ID move independently to intermediate target positions of an energy scan. In continuous mode (CM) MONO and ID cover the whole range of the scan nonstop in a coupled motion. Data acquisition is done continuously at the speed provided by the CM scan and is available in regular user operation [1]. Currently CM is in operation at 11 undulator beamlines at BESSY II. 3 new beamlines requesting CM are under construction. During CM the MONO EPICS IOC acts as a controller forcing the MONO optics to follow the movement of the ID. A non-linear predictive control scheme is used to implement this dynamic coupling. The controller task utilizes polynomial regression to extrapolate the ID motion. Calculation of the trajectories for MONO grating and mirror is based on bijective gap to energy lookup tables and the grating equation. In this paper the technical implementation, limitations, recently developed diagnostic methods, and future plans for improvements are presented.

INTRODUCTION

The continuous mode is part of the monochromator control program (MCCP) and has been in operation since 2005 at a dipol beamline. The first employment at an undulator beamline was 2006 [2]. At the time of writing, 11 undulator beamlines (UE56/2 PGM1, UE56/2 PGM2, UE49/1 PGM1, UE52/1 SGM1, UE52/1 PGM1, UE46/1 PGM1, UE46/1 PGM2, UE56/1 PGM1, UE112 PGM1, U49/2 PGM1, U49/2 PGM2) and 6 dipol beamlines (PM1...4, HESGM, ISISS) support CM operation by the monochromator software. 3 beamlines (U125/2 PGM1, U411/1 PGM1 and EMIL-UE48/U17 PGM [3]) are under construction requiring CM. Dipol light sources provide a continuous spectra. Energy scanning can be done by solely moving the optical elements of the monochromator which makes the implementation of CM relatively easy. On undulator beamlines the motion has to be coupled with the undulator gap and shift axes that are controlled by the insertion device control program (IDCP). The mapping between energy and gap/shift is calculated using lookup tables and piecewise polynomial or splined interpolation. The tables are held by the MCCP, which sends the target positions for gap and shift via CAN bus to the IDCP and receives the gap/shift position feedback sent from the undulator (Fig. 1). Task synchronization for combined movements has to be done by the MCCP. As a result, the monochromator acts as slave and follows the movement of the undulator (master) .

Motivation

BESSY beamlines split up into branches with up to four experimental stations per undulator light source. Depending on the application, users can greatly benefit from the fact that CM scans are about five times faster than step scans [4]. In addition, exposure times of the samples are shorter and, in some cases, the measurements are even better [5].
A robust control loop with no maintenance effort, and without the need of manual interaction, is necessary in order to ensure accurate user operation with variable scan speeds. This requirement is even more challenging for the technical setups of the beamlines currently under construction [3]. Diagnostic methods are needed to locate possible disturbances from CAN/Ethernet-jitter, inaccurate lookup tables and vibrations.

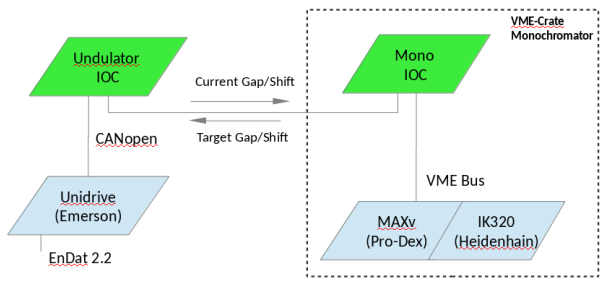


Figure 1: Interfaces used for CM from monochromator to motion controller (VME-Bus), encoder card (VME-Bus) and undulator IOC (CAN). VME based hardware for monochromator: MVME162 (Motorola), MAXv (Pro-Dex), or VME6, OMS58 (OMS), IK320 (Heidenhain).

TECHNICAL DESCRIPTION

The beamline control system is based on EPICS. Monochromator and undulator are controlled by EPICS IOCs (IDCP, MCCP). Both provide EPICS based operator interfaces. Additionally, the monochromator IOC provides connectivity over the BESSY extended monochromator control protocol (EMC) via RS232 or Ethernet. Users can control the beamline using EMC or EPICS. The communication via RS232 is slow but has proven to be reliable for CM operation when deterministic synchronization of data acquisition and monochromator energy feedback is critical. The CAN bus communication has been chosen between monochromator and undulator in order to reduce the jitter of the periodic position updates sent to the monochromator (Fig. 1).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Control Loop

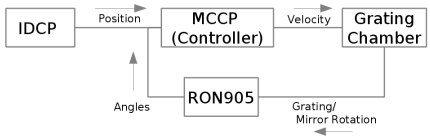


Figure 2: Feedback loop. Desired velocity profiles derived from the readback recieved from the undulator control program (IDCP). Monochromator position feedback from Heidenhain system (RON905).

A combination of the IK320 counter cards and a RON905 (Heidenhain) incremental angle encoder ensure an extremely high angular resolution of about 0.01 arcsec depending on noise and vibrations. The access time of the measured value of the IK320 is about 0.2 ms. This is relatively fast compared to the 10, 20 Hz position update from the undulator (Fig. 1). The MCCP receives the gap positions from the undulator and builds a list of undulator gaps and time stamps. The controller algorithm works at a variable rate from 2 to 4 Hz and forces the optical elements to follow the undulator by setting the proper velocities of grating and mirror. This is done by sending jog commands for linear blended moves to the motion controller (Fig. 1-3), causing the slow control loop update rate. Eventually, the result is a good approximation to the required non-linear velocity profile (Fig. 9).

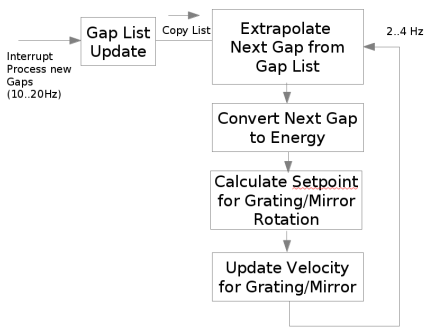


Figure 3: Feedback loop flow chart.

One-Step Ahead Prediction of Undulator Energy
Due to the slow update rate of the control loop (e.g. 4 Hz) and the update rate of the current gap position (e.g. 20 Hz) it is essential for the accuracy of the CM to extrapolate the gap to the time of next control loop update (t_{N+1}).
Based on the observation of the last N gaps, two different extrapolation modes are implemented. First, obtaining the slope m via linear regression, the one-step ahead predicted Gap(t_{N+1}) is calculated as

$$\begin{aligned} Gap(t_N) &= m \cdot t_N + Gap(t_0) \\ Gap(t_{N+1}) &= Gap(t_N) + \frac{m}{f} \end{aligned}$$

, where f is the update rate of the control loop. Gap(t_0) is the first gap in the list and Gap(t_N) is the last one.

The second method is quadratic interpolation fitting better to the non-linear relationship of gap to energy as well as to the trajectories of the gap motion. Obtaining a,b,c after translation of the coordinate system and quadratic polynomial regression, the Gap(t_{N+1}) is calculated according to

$$Gap(t_{N+1}) = (a \cdot t_{N+1} + b) \cdot t_{N+1} + c + Gap(t_0)$$

The inverse mapping of energy to gap can be determined by the help of 3rd order polynomial interpolation or cubic splines using the undulator lookup table. The grating equation, monochromator type (PGM/SGM) and conditions (fix focus, fix beta, fix theta) are used to estimate the velocity necessary to force the monochromator to the same energy at the time t_{N+1}. Figure 3 shows the flow chart of the control loop.

Technical Requirements

The following error (FE) is the difference between monochromator energy and undulator energy. This error must be within the limits defined by the bandwidth of the undulator. One result of the FE are intensity modulations during the scan which should not exceed 2% of the maximum intensity for a given harmonic. The maximum tolerable FE significantly decreases with the bandwidth of higher undulator harmonics, given by

$$\frac{\Delta E_h}{E} \approx \frac{1}{hN}$$

, where N is the number of undulator periods and h is the undulator harmonic. For example at 435 eV on a beamline at U49/2 we have a maximum FE of about 0.67 eV (1st harmonic), 0.25 eV (3st harmonic) and 0.18 eV (5th harmonic). Hence, the gap position should be accurate in the range from 3 μm (5.th harmonic) to 50 μm (1st harmonic). Inaccurate lookup tables for the mapping from gap to shift could have a major impact on the maximum value of FE. The demanding control specifications in particular for higher scan speeds (dE/dt) have led to the development of diagnostic software.

Diagnostics

The diagnostic of the CM is not an easy task. Critical are the CAN bus communication, position measurement, lookup tables, the control loop and the actuators needed to move the monochromator and undulator.
The feedback module is a library for the EPICS framework written for real time feedback and data acquisition used for beamline diagnostics and optimization [6]. A recent software development at BESSY II is the implementation of a feedback module plugin for the MCCP in order to improve diagnostics and user feedback of the existing implementation of the CM. The interrupt triggered scheduling of the data processing, utilizing the VxWorks auxiliary clock (auxClock), allows data acquisition at rates of up to 4 kHz. This is a theoretical limit for the encoder data and can be generated for grating and mirror position by triggering the encoder cards after data processing. The readout starts then with the next feedback loop (Fig. 5). Reasonable feedback rates for

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CM are in the range from 100 Hz to 1 kHz.
Monochromator energy, undulator energy and rotation of grating and mirror (Phi, Psi) are relevant parameters during CM. The gap position update rate via CAN bus can be analyzed and correlated to the higher frequency feedback of monochromator energy which is obtained via the grating equation and the encoder feedback for grating and mirror (Phi, Psi). Figure 4 shows the user interface of the plugin.

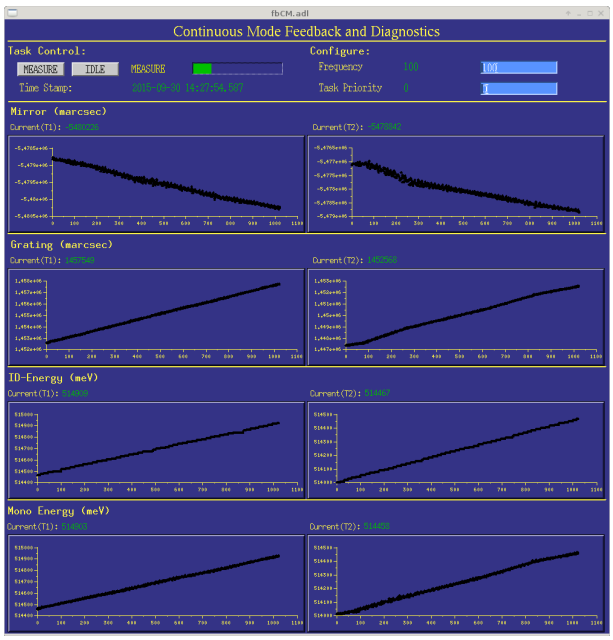


Figure 4: Continuous mode diagnostic panel for visualization during scan.

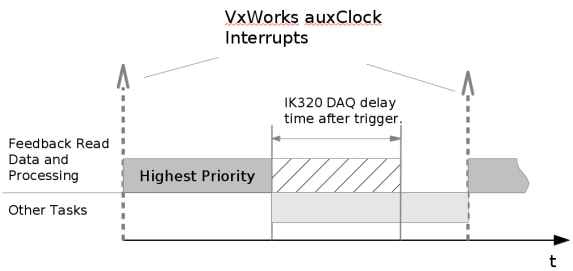
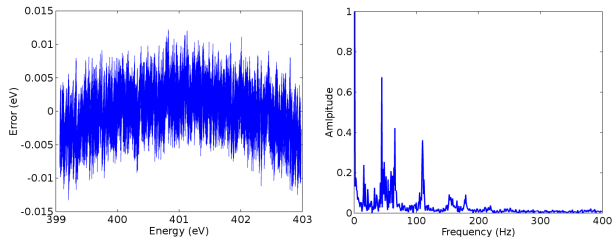


Figure 5: Real time position data acquisition and scheduling. The feedback task can be configured to run with highest priority 0. No polling or extra task switches are necessary to wait for encoder data to be ready.

Data Acquisition During data acquisition the user can read single data sets consisting of monochromator energy and data index. The waveform data can be received by monitoring two datasets (4 * 1024 values), which is undulator and monochromator energy, mirror and grating rotation. One set of arrays is filled by the feedback task while the other one is copied to the EPICS record layer (Fig. 4).
Figure 6 shows the divergence from constant energy velocity of a CM move at a high data rate. This could be used for post mortem correction of the energy scale by correlating the data acquisition of the experiment with the position data provided by the feedback plugin of the monochromator.



(a) (b)

Figure 6: Divergence of linear energy scale due to variance of dE/dt (a) and FFT of divergence signal (b).

Complex filters can be applied after the scan in order to remove disturbances like noise and vibrations from actuators.

Sample Frequency and Task Priority For such an application the feedback task has to be scheduled with highest priority so that the noise induced by the jitter of the data acquisition can be neglected (Fig. 5).
The waveform of Fig. 6a has no significant spectral contributions beyond 200 Hz. Hence, choosing a sample rate of 400 Hz, the waveform can be recovered with an error of the magnitude of the random noise level [7]. In the example of Fig. 6a it is in the range of meV and can, in principle be derived from the standard deviation of the position measurement of about $\sigma = 0.01$ arcsec and the grating equation.
Another application of the plugin would be during step mode. Closed loop positioning can by analyzed and any number of monochromator energy samples can be taken and precisely correlated to the measurement on the target position.

Position Update Jitter Distribution While gap position update jitter has no direct effect to the monochromator energy, it can have a big impact on the stability of the control loop causing intensity modulations. The jitter-distribution (Fig. 7) can be derived from the waveform data above.

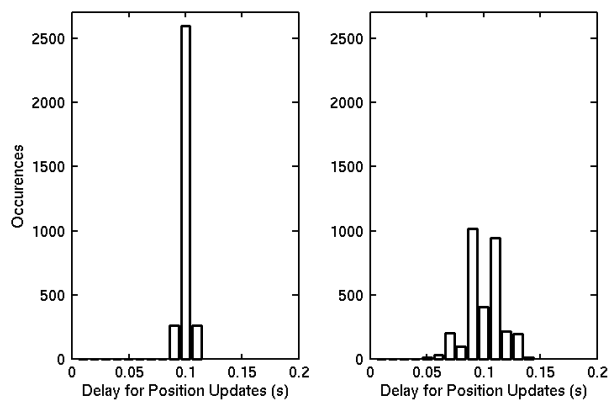


Figure 7: Jitter distribution for CAN bus communication diagnostics.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

EXPERIMENTAL VALIDATION

We verified the CM control loop and user feedback by taking a spectrum of molecular nitrogen. Figure 8 compares the conventional step mode with the continuous mode and the improved wavelength scale using data provided by the CM plugin of the feedback module. The scan time could be reduced by a factor of 5 improving the sampling rate at the same time.

A peculiar problem of incremental angular encoders, used for the positioning of the optical elements in monochroamtors, are interpolation errors that lead to a modulation of the wavelength scale of the monochromator. This is commonly encountered with the Heydemann correction [8] which is implemented in a fast and transparent way and also available in the continuous mode. The corrected data has been used to determine monochromator energy at a rate of 800 Hz. Disturbances like vibrations and noise can be filtered out of this energy trajectory profile for evaluation. This makes it in principle possible to improve the wavelength scale of the scan as demonstrated in Fig. 8.

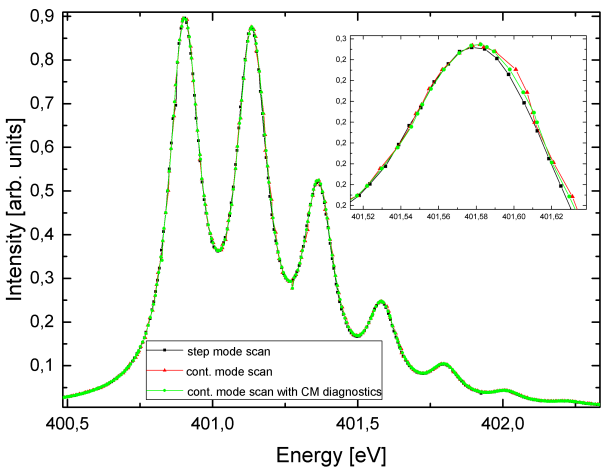


Figure 8: Nitrogen absorption spectra and comaparison of step mode (C_{ff} = 2.25), CM, and calibration of wavelength scale using the CM feedback plugin.

LIMITATIONS

At BESSY II there is a need for constant energy velocities. Unfortunately, the gap motion is limited to trapezoidal velocity profiles. Planar undulators would have to vary the velocity according to the non-linear mapping from gap to energy. An excessive example of this artifact is illustrated in Fig. 9. The energy rate dE/dt decreases significantly during the scan due to the non-linear mapping from energy to gap.

In order to obtain constant circular polarization, gap and shift would have to move on a complex path. This is very challenging with the existing motion control of the undulator (Unidrive, Emerson) and has not been implemented yet. Due to this limitation on the ID-controls, helical undulators cannot move the gap and shift synchronously. Energy scans have to be performed with fixed shift. As a result, scans

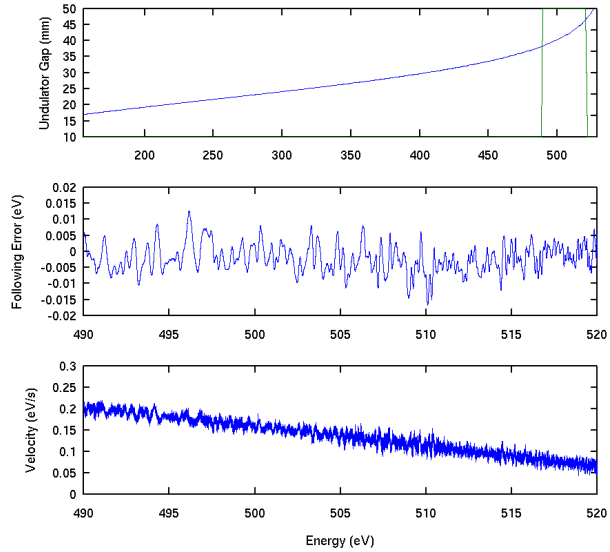


Figure 9: Energy vs. gap and the effect to dE/dt for moves with constant gap velocity.

are limited to only small ranges with a tolerable drop of elliptical polarization.

PROCEEDING DEVELOPMENTS AND FUTURE PLANS

Considering the exceeding costs and manpower to replace the existing motion controllers, we prefer the extensive solution to implement the complex path of gap and shift axes, which are required for elliptical polarization scans with constant energy velocities, on the existing motion control of the undulator (Unidrive, Emerson). This can hopefully be done in near future.

For the new beamlines being build for the Energy Materials In-Situ Laboratory Berlin (EMIL [3]) a new motion control scheme for the monochromators has been chosen. The axes involved in CM are controlled by Geobrick IMS 2 (Delta Tau). As an result, we have a non deterministic Ethernet connection between monochromator IOC and the motion control (Fig. 10). Our investigation showed that jitter introduced by the network would not be acceptable. A possible solution would be to split the encoder signal of the undulator gap axis (EnDat 2.2, Heidenhain) as already proposed in SUMS [9] or to use the incremental quadrature encoder signal output from the Unidrive for gap position feedback with very low jitter at the rate of the servo update clock of the Geobricks (e.g. 5 kHz). Therefore, an accessory of the Geobricks using a fiber MACRO ring connection (Delta Tau) is located in the cabinet of the undulator. Data arrays representing the motion profile are distributed to the motion controls using EPICS waveform records. Tests have shown a smooth output in a cubic spline motion profile on the Geobricks.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

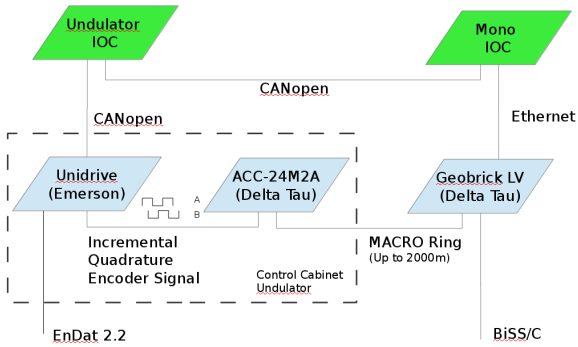


Figure 10: Interface used for CM from monochromator IOC to motion controller is non deterministic Ethernet. Direct encoder feedback from undulator gap to Geobrick.

CONCLUSIONS

An robust control scheme for CM is in operation at 11 undulator beamlines and 6 dipol beamlines. The monochromator successfully follows the undulator motion, intensity modulations can be avoided.

1 kHz real time energy feedback from the monochromator can be provided to the user by EPICS waveform records and time stamps. Diagnostic tools allow the detection of critical following errors, gap update jitter and other disturbances like vibrations and instabilities. Tests are ongoing for the undulator motion control to implement small segments of linear blended moves as approximation for the non-linear move profiles of gap and shift axes. For the new EMIL project the use of a new coordinated multi-axis motion solution has already been tested. A smooth motion control output for on the fly energy scans can be generated by programming splined moves in order to generate the complex path of grating and mirror.

ACKNOWLEDGMENTS

We would like to acknowledge the help from Stefan Gottschlich and Götz Pfeiffer for preliminary investigations of the Unidrive systems, and Winfried Frentrup for calculating and optimizing undulator lookup tables as well as estimating the undulator bandwidth. Gerd Reichardt assisted us by his management of the EMIL project. Thanks to Joachim Rahn for supervision and valuable feedback.

We would like to especially thank Roland Müller for assistance and steady encouragement.

REFERENCES

[1] V. Cuartero et al. *"X-ray magnetic circular dichroism study of the magnetic anisotropy on $TbMnO_3$"*, Phys. Rev. B 91, 165111 (2015).

[2] R. Follath, *"Stand des Continuous Mode bei BESSY II"*, Internal Report, 2007.

[3] R. Follath, M. Hävecker, G. Reichardt, K. Lips, J. Bahrdt, F. Schäfers, P. Schmid, *"The Energy Materials In-Situ Laboratory Berlin (EMIL) at BESSY II"*, 11th International Conference on Synchrotron Radiation Instrumentation (SRI). 2012.

[4] R. Follath, J.S. Schmidt, M. Weigand, K. Fauth, *"The X-ray microscopy beamline UE46-PGM2 at BESSY"*, 10th International Conference on Radiation Instrumentation (SRI). AIP Conference Proceedings, Volume 1234, Issue 1, p.323-326. 2009.

[5] J. Krempaský, U. Flechsig, T. Korhonen, D. Zimoch , Ch. Quitmann, F. Nolting, *"Synchronized monochromator and insertion device energy scans at SLS"*, AIP Conf. Proc. 1234, 705-708. 2010.

[6] A. Balzer, P. Bischoff, R. Follath, D. Herrendörfer, G. Reichardt, P. Stange, *"Diagnostics and Optimization Procedures for Beamline Contr AT BESSY"*, 10th ICALEPCS Int. Conf. on Accelerator & Large Expt. Physics Control Systems. Geneva, 2005.

[7] R. N. Bracewell, *The Fourier Transform and Its Applications*, 2nd Edition, Prentice Hall International, London. 1974.

[8] R. Follath, A. Balzer, *"Heydemann Algorithm for Energy Scale Linearisation"*, 10th International Conference on Synchrotron Radiation Instrumentation (SRI), Melbourne, 2009

[9] Manuel Izquierdo, Vincent Hardion, Guillaume Renaud, Lilian Chapuis, Raphael Millet, Florent Langlois, Fabrice Marteau, Christian Chauvet, *"SUMS: synchronous undulator–monochromator scans at Synchrotron Soleil "*, Journal of Synchrotron Radiation, 619-626. 2012.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MANAGING NEUTRON BEAM SCANS AT THE CANADIAN NEUTRON BEAM CENTRE

Mark Vigder, National Research Council, Canada
Martin Lee Cusick, Canadian Nuclear Laboratories, Canada
David Dean, National Research Council, Canada

Abstract

The Canadian Neutron Beam Centre (CNBC) of the Canadian Nuclear Laboratories (CNL) operate six beam lines for material research. A single beam line experiment requires scientists to acquire data as a sequence of scans that involves data acquisition at many points varying sample positions, samples, wavelength, sample environment, etc. The points at which measurements must be taken can number in the thousands with scans or their variations, having to be run multiple times. At the CNBC we have developed an approach to allow scientists to specify and manage their scans using a set of tools we have developed. Scans are specified using a set of constructors and a 'scan algebra' that allows scans to be combined using a set of scan operators. Using the operators of the algebra, long complex scan sequences can be constructed by combining shorter simpler scans. Based on the constructors and the algebra, tools are provided to scientists to build, organize and execute their scans. These tools can take the form of scripting languages, spreadsheets, or databases. This scanning technique is currently in use at CNL, and has been implemented in Python on an EPICS based control system..

INTRODUCTION

The Canadian Neutron Beam Centre (CNBC) conducts material science experiments using neutron scattering techniques. The neutron source is an experimental reactor that provides six beam lines for neutron scattering. Each beam line is specialized for a particular scanning purpose, such as triple-axis spectrometery, reflectometery, stress-scanning diffractometers, etc. In addition to the equipment controlling the spectrometer beam lines, a set of ancillary equipment is used to control the sample environment. This includes equipment such as furnaces, cryostats, stress rigs, etc.

When conducting a single experiment, scientists have a very large number of independent variables they must control. For the beam line this includes beam position and focus, energy levels on the main and diffract beams, and polarization of the main and diffract beams. Sample location and orientation must be controlled, as well as sample environment (temperature, stress, etc.) Setting each of these independent variables could involve setting multiple sub-variables, or sequencing through a number of steps to correctly set the variable.

An experiment involves setting the independent variables to the appropriate values and initiating a data acquisition phase for a specified duration. The independent variables are then reset to new values and the data acquisition repeated. A single experiment repeats this process performing a data acquisition at possibly thousands of different points within the n-dimensional space formed by the independent variables. Acquiring the data at a sequence of points in the experiment space is referred to as scanning.

Scanning is often a dynamic and interactive process. When conducting scans, scientists monitor the data as it is being acquired and will often stop, restart, modify, rerun, or resequence a scan. Creating and managing these scans and dynamically interacting with the scanning process presents a number of challenges to the scientist and to the system developer who must support the scientists work.

OBJECTIVE

The objective of this work is to develop a method that allows scientists to easily create, manage, and execute their scans. The work was undertaken during a major upgrade of the control and data acquisition system. The original spectrometer control system at NRU was developed incrementally over more than 30 years of work and ran on a VAX based system. In the course of replacing it with a distributed Linux based system, the opportunity arose to replace the outdated user interface with a modern interface, including improving the techniques for scientists to perform the scanning for experiments.

In redoing the scanning control and management, the following requirements were followed.

First, there was not to be a major paradigm shift in how the scanning process was managed by the scientists. The old system was dated, had no underlying model, and carried thirty years of baggage in its design and software. However, it also was designed by the users who knew what they needed, was very efficient and easy to learn, and carried thirty years of detailed knowledge of the scanning process. The new system was not to be a major shift in the process, but rather was to provide a better model of scanning, a refactoring of the design and implementation, and much more capability and flexibility that would be provided to users.

Second is the requirement that identical scanning control management software is to be used on all the beam lines. With minimal resources to maintain and manage the software, having different software on the different beam lines was not a viable option. Even software variants, no matter how similar, create software management problems. However because each beam line is unique this required that the system

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

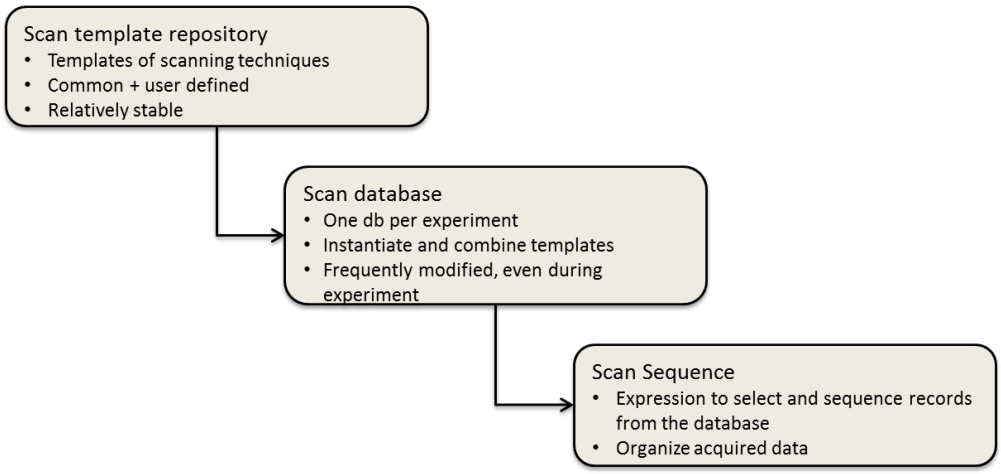


Figure 1: Managing scans.

be designed in such a way that it allowed for different configurations for different beam lines.

Finally, the system must be 'easy' to learn for incoming scientists. The CNBC is a user facility that hosts a number of researchers for days or weeks at a time. A local contact guides these users through the process of conducting an experiment. However, the visiting scientists should be able to work independently and not require the local contact for relatively routine operations performed during the experiment. A system in which the full functionality of the system can be learned by a naïve user within a few hours is clearly impossible. However, many of the operations performed by scientists during an experiment are simple in concept and repetitive in nature, and these should be available to visiting scientists with minimal training. More complex operations would still require intervention from the local contact.

MODELLING SCANS

The first step in upgrading the scan management and control system was to build a more formal model of scans. This was done using a simple algebra. Working within the n-dimensional space of the independent variables, the following can be defined.

An *independent variable* is a name and a type. The type is the set of valid values for the variable.

A *scan point* is a set of tuples, were each tuple consists of an independent variable, and an optional value for the variable.

A *scan* is an ordered sequence of scan points such that each point of the scan contains the same set of independent variables.

Scan Operators

As part of the scan algebra, a set of binary operators are defined. These operators are used to construct complex scans from simpler scans. The binary operators used are the following.

Sequencing (**+**). The scan sequence operator extends one scan with a second scan, **S1+S2** creates a single scan with the ordered scan points of **S1** followed by the ordered scan points of **S2**.

Looping (*). The scan looping operator allows one scan to be repeated for every point of another scan. The expression **S1*S2** is a scan of length **len(S1)*len(S2)** in which each point of **S1** is replaced by a scan consisting of that point extended with the entire scan **S2**. For example, if scan **Q1** steps the scattering angle **Q** through a sequence of points, and scan **T1** steps temperature through a sequence of values, then **Q1*T1** is a scan that scans the scattering angle **Q** at each temperature in **T1**.

Interleaving (|). The scan interleaving operator interleaves the scan points from two scans. For example, if **Q1** and **T1** are defined as in the previous paragraph, and scan **BG1** offsets an angle to do a background measurement, then **(Q1 | BG1) *T1** scans **Q** at each temperature of **T1** while also performing a background count for each point.

Dot product (^). The dot product operator joins two scans together point by point.

MANAGING SCANS

The method for managing scans is illustrated in Figure 1. There are three steps to the process.

Scan repository

The first step is the construction of a *scan repository*. The repository contains parameterized versions of standard scans. The most basic parameterized scans in the repository include the following:

- **step**. The step scan steps a set of independent variables through a linear set of equidistant steps. For example, step angle phi from 23 to 29 in steps of 0.5 degrees.
- **points**. A point scan assigns a sequence of values to a set of independent variables. For example, it can be used to set the main beam and diffract beam flipper coils in order to sequence through the different polarization options.
- **setup**. A setup scan is a single point scan. It is used to simply set a set of independent variables to

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

specific values. It is usually combined with a **step** scan to initialize a set of values and then step an independent variable.

In addition to these parameterized scan templates, a number of more complex scan templates are also stored in the repository. For example, the **texture** scan stores over a thousand scan points representing different sample orientations.

The scan repository is relatively stable and will generally contain no more than a few tens of scans at most. Additions to this repository are generally required only as new scanning techniques are developed.

Scan Database

A scan database is created by the scientist for each experiment being conducted. Each record of the database is a scan and is constructed by instantiating scan templates from the repository and combining the instantiated scans using the scan operators.

Unlike the scan repository, the scan database is highly dynamic. The scientist initially builds the database before conducting the experiment and is continuously modifying and updating the database during the course of the experiment.

Each record of the database is constructed by the user by first selecting the templates from the repository and instantiating them. For example, to do a texture scan across a range of temperatures, the user instantiates a texture scan and a step scan stepping temperature and combines them using the looping operator.

Sequencing

Once the user has constructed the scan database, the user executes the scans by creating a sequence expression. A sequence expression extracts records from the scan database and specifies the order in which the records will be executed. A few simple operators are available for representing repetition, sequencing, and selecting a range of records. For example, given that records within the database are referenced by record number, the expression: **(3-10), 25*10, (10-3)** instructs the system to execute scan records 3 to 10, followed by executing record 25 ten times, followed by executing records 10 to 3.

IMPLEMENTATION

The scanning control and management system has been implemented and is currently being deployed on the various beam lines at the CNBC. The main elements of the implementation include the following.

The scan algebra is implemented using the Python dynamic language. A scan class is defined that encapsulates the scan properties. Being a dynamic language, operators of the language can be overloaded allowing Python expressions to be used in a natural way to write the scan algebra expressions.

The repository is simply a set of parameterized Python functions. These functions are called *scan constructors*. Instantiating and calling the function returns a scan. Being relatively small in number, no special database is required to organize the scan constructors. Moreover, with a little Python knowledge and understanding of the scan operators, users can easily write their own scan constructors.

The database required a tool by which the scientist could construct scan records by calling scan constructors and combining the scans. It is possible for records to be constructed directly in Python and the resulting scans stored in a database. However it was felt that this would add a level of complexity that would be a barrier to many of the users. Most experiments are very regular and repetitive in how they scan, and a relatively easy solution is to represent the scan database as a spreadsheet, with the columns representing the scan templates and their parameters, and each row representing a scan record. This provides the user with the full editing capability of the spreadsheet for constructing the scans, while providing an intuitive representation understood by most scientists.

Finally, a GUI is provided through which users can view scans, construct sequences, execute and monitor scans.

DISCUSSION

The old scan management system developed over many years, and which has proven effective, has been used as a basis for the major upgrade that has been performed. This has resulted in a system which is familiar to users but which also has the power and capability to be further evolved. Since it was a complete rewrite of the code, we were able to start with a clean slate and hopefully will be as successful over the next thirty years with the new system.

The formalization of the concept of scanning using the simple algebraic techniques has proven to be a useful technique with both simplicity and power. Complex scans of thousands of points have been constructed using the algebra.

Python, always popular within the scientific community, provides an implementation of the algebra that is easily understood by both developers and scientists. Using spreadsheets for representing the scan database was a technique that scientists were very comfortable with. It is limited in its capability but does handle the majority of requirements in an intuitive manner.

The system is currently being deployed at three of the beam lines at the CNBC, with plans to deploy on the other three lines.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

PRELIMINARY DESIGN OF A REAL-TIME HARDWARE ARCHITECTURE FOR eRHIC*

R. Michnoff#, P. Cerniglia, M. Costanzo, R. Hulsart, J. Jamilkowski, W. Pekrul, Z. Sorrell, C. Theisen,
Brookhaven National Laboratory, Upton, NY 11973, USA

Abstract

The 3.8 km circumference Relativistic Heavy Ion Collider (RHIC) at BNL has been in operation since 2000. An electron-ion collider (eRHIC), which is in the design phase, plans to use one of the existing ion rings and new electron rings constructed in the existing tunnel to provide collisions of up to 21.2 GeV electrons with up to 100 GeV gold ions, 250 GeV polarized protons, as well as other species. Many new real-time systems will be required to satisfy the needs of eRHIC, including over 2000 beam position monitors, 1000 beam loss monitors, 18 current monitors, feedback systems, controls for about 10,000 power supplies, machine protection system, new beam timing systems, and more. The selected architecture must be flexible, expandable, cost-effective, reliable, and easy to maintain. Interface with existing and new accelerator timing systems is required, and compatibility with existing infrastructure and equipment must be maintained. Embedded modules based on the Xilinx Zynq gate array, with direct Ethernet connection and on-board Linux, housed in multi-slot chassis (VME, VXS, VPX, TCA, etc.) is under consideration. Preliminary design concepts for the architecture will be presented.

CONTROLS ARCHITECTURE HISTORY

Hardware architectures for accelerator controls have gone through many transformations as technology has evolved. In the 1970's, custom field-busses were designed to control remote devices in accelerators. Engineers at BNL's Alternating Gradient Synchrotron (AGS) designed a system called Datacon – a multidrop field bus capable of withstanding the electrically noisy accelerator environment [1,2]. A central computer typically controlled all of the devices. At AGS, the then state-of-the-art PDP-8 and PDP-10 were used.

As hardware evolved, real-time controls became more distributed. Intel's Multibus architecture became a standard in the 1980s for systems developed at the AGS. A typical Multibus system consisted of a custom enclosure with a backplane card cage housing processor, memory and I/O modules. A locally resident RMX deterministic real-time operating system was used. These units worked very well, but although the processor and some I/O modules were common, a major disadvantage was that each system required custom cable assemblies and connector panels to external devices. This made it very difficult to maintain spares because each system required a dedicated spare chassis.

*Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy.
#michnoff@bnl.gov

Then in the 1990's, VME-bus became a popular platform at AGS and for the RHIC project that was under construction at that time [3]. System maintenance greatly improved with the use of VME because modules could easily be replaced when failures occurred. Swapping VME boards is very simple compared to fully encased systems like Multibus and custom designed enclosures.

More recently, manufacturers and accelerator hardware engineers are commonly incorporating direct Ethernet connectivity into systems. This has provided tremendous advantages for system designers. However, custom chassis have again become popular, which can increase the complexity of maintenance and spare inventory.

THE eRHIC APPROACH

The present plan for eRHIC real-time hardware systems is to use the Xilinx Zynq family system-on-a-chip (SoC) devices (or similar newer components that become available), which combine large programmable gate arrays with embedded ARM processors all on the same integrated circuit. A suite of modules will be developed and housed in multi-slot backplane chassis such as VME, VXS, openVPX, or TCA, with each module containing a direct Ethernet connection for communication to the higher-level control system. An operating system such as Linux will reside locally on each module.

Gbit serial links will be used for communication between modules via either front panel connections or via Gbit serial lanes on the backplane. This high-speed communication across multiple modules is essential for providing the real-time operation required for many accelerator systems. Details for the communication links are not yet fully defined, but standard protocols will be used where feasible. A draft design is provided in fig. 1.

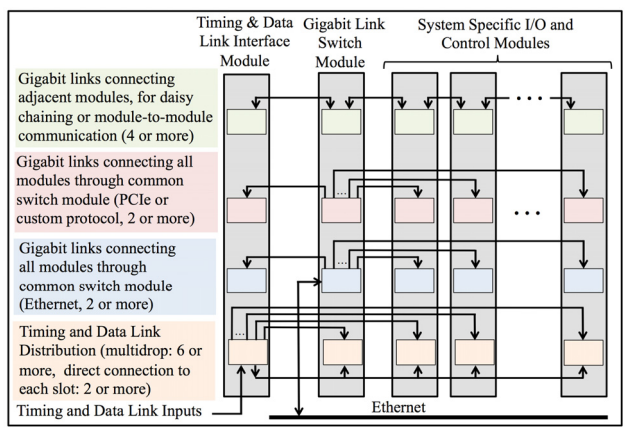


Figure 1: Draft design of inter-module communication. Communication via backplane is most desirable; some paths may be via front or rear panel connections.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

IS A DETERMINISTIC REAL-TIME OPERATING SYSTEM ESSENTIAL?

Deterministic real-time operating systems have been vital for many years in order to satisfy time-critical system functions. Processor modules in VME-bus systems currently installed in the BNL Collider-Accelerator complex use the real-time operating system vxWorks [4]. Task priorities are carefully planned to ensure that time critical functions are properly handled and interaction between hardware modules within the chassis is well orchestrated.

With newly available components like the Xilinx Zynq SoC, the majority of the real-time aspects are now performed within the gate array code, and the on-chip processor running Linux or similar OS acts like a window into the hardware that simply passes data to and from higher level workstations. Although Linux task priorities (or nice values as they are called) must still be carefully planned, true real-time functionality at the Linux level is not as important anymore. And with a low cost SoC installed on every hardware module, systems can be more distributed than ever; so each processor performs somewhat dedicated tasks. This is a major advancement!

THE PROTOTYPE

A first prototype using the Zynq XC7Z030 SoC with on-chip dual ARM Cortex A-9 CPUs has been developed (V301) for installation in a VME chassis (Fig. 2). The VME backplane use is limited to providing 5V power to the module and receiving machine timing and data link signals bussed over user-defined P2 pins from an in-house designed module (V208, Fig. 3). A VME interface is not provided; instead a direct front panel Ethernet connection is used for Control System communication. One of the on-chip CPUs is running Linux, and the second CPU, although not currently used is available if required.

The V301 is a fully contained beam position monitor system with flexibility for measuring several beam species including protons, ions, and electrons. This module is not only intended as a prototype for eRHIC, but will also be used as a next generation BPM system for the existing and other future machines within the BNL Collider-Accelerator (C-A) complex.

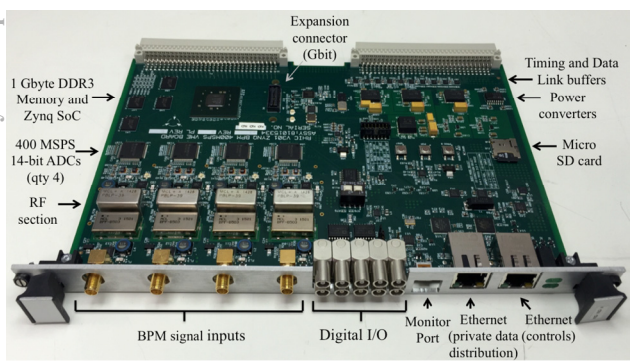


Figure 2: Photograph of V301 module.

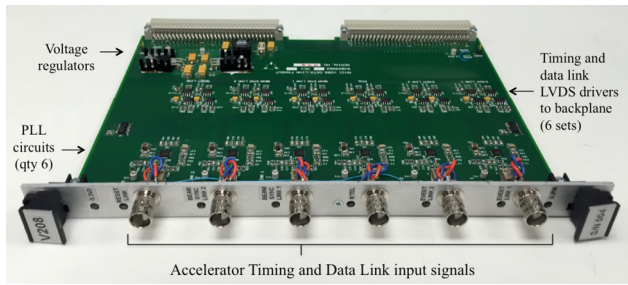


Figure 3: Photograph of V208 Timing and Data Link interface module.

Additional modules under consideration to be developed in the near future include:

- A multi-channel analog input module.
- A simple carrier module with connector to gate array pins to allow custom I/O boards to be easily developed into the Zynq architecture.

These new modules will include the digital section that has already been designed. The goal is to use a common footprint for the Zynq SoC, memory, and power converters so that new boards can rapidly be developed.

MAJOR DESIGN REQUIREMENTS

Some of the major design requirements for eRHIC real-time hardware systems include:

- Minimization of equipment rack space
- Easy replacement of operational modules
- High reliability
- Simple software configuration of modules
- Simplified interface to machine timing and data links
- High speed processing of I/O signals
- Two or more Ethernet connections on each module
- Embedded operating system on each module with tightly coupled interface to real-time hardware, and with control system software objects resident locally
- Gigabit communication between modules
- Ability to use commercially available modules
- Ability to share custom hardware modules between other facilities

Each of the above items is addressed below, including explanations of how the requirement will be satisfied.

The requirements in this list are not unique for eRHIC, but can be considered common for all modern accelerator control systems.

Minimization of Rack Space

Since the cost for buildings is high, minimizing the floor space required to house electronics seems obvious. While this requirement is not always at the forefront of our minds when engineering systems, higher and higher density electronic components with smaller footprints have allowed this requirement to be satisfied without tremendous thought.

Providing increased capability into each hardware module is part of satisfying this requirement, but system packaging details must also be carefully considered.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

With more and more embedded systems becoming available with direct Ethernet connections, many manufacturers and accelerator electronic design engineers have developed custom fully integrated enclosures for packaging of systems.

While this architecture works well when only a few units are needed, for systems like eRHIC that will require over 2000 BPMs, rack space requirements would be huge for so many individual units.

Therefore, the hardware architecture design for eRHIC will be based on a multi-slot chassis system like VME, VXS, openVPX, or TCA with many slots in order to minimize rack space requirements.

One might argue that using a multi-slot chassis can be very expensive if only a few slots are used. For these applications, chassis with fewer slots are an option.

The multi-slot chassis architecture will also satisfy other design requirements as will become evident in additional sections below.

Easy Replacement of Operational Modules

Minimizing the time required to replace failed hardware modules during operations is quite important in order to limit machine downtime.

For the present RHIC BPM system, which has been in operation since 2000, two hardware boards (one for each BPM plane) are installed in a custom 3U rack-mount chassis. Maintenance is quite difficult because replacing a failed hardware module requires removing the chassis from the equipment rack, removing many screws to access the interior of the enclosure, and then removing more screws and connections to access the board.

A multi-slot chassis system with modules like the V301 will provide easy replacement by simply disconnecting external cables and removing two front panel screws.

High Reliability

As with all accelerator systems, equipment must be robust and highly reliable. Radiation induced failures must be considered in the design.

Simple Software Configuration

For most front end computers and embedded systems with direct Ethernet connections, configuring the network settings usually requires a cumbersome method like local logon via a serial port or local monitor and keyboard to configure parameters in either flash or disk files.

With possibly thousands of modules at eRHIC having direct Ethernet connections, simplifying the configuration process is essential.

The prototype V301 module provides the required simple configuration. The network IP settings and host name are stored in files on a removable micro SD card, which also contains a boot file. When the module boots up, the hosts file and /etc/network/interfaces network configuration file are copied from the SD card to the local ram disk. Then a remote server is mounted, and the host name is used to point to a directory on the server that defines the software modules to be automatically loaded, including the gate array file. The technique of loading software modules from the server has been in use at RHIC for many years. The major difference here is the use of a removable micro SD card.

The beauty of using the micro SD card is that it can be set up externally, allowing a desktop computer to be used to easily pre-configure the micro SD cards. In addition, if an operational hardware module fails and requires replacement, the micro SD card can simply be removed from the failed module and installed in the replacement module, thereby preventing the need to reconfigure the new module using a direct connection.

Auto-configuration using DHCP is another option being considered.

Simplified Interface to Timing and Data Links

Several serial Manchester encoded 10 MHz and 14 MHz timing and data links exist in the Collider-Accelerator complex to distribute important machine data and provide synchronized machine timing triggers [5,6]. The eRHIC hardware architecture must provide a simple interface to these links.

Existing hardware in the C-A complex typically provides a direct hardware connection to the links, thus requiring that each module have several PLL circuits (one for each link), which although not complicated, requires a fairly large PC board footprint (Fig. 3).

For eRHIC, a goal is to further simplify the link distribution by connecting one copy of each link to a single module in the multi-slot chassis, and distributing the signals to all modules via the backplane.

A custom designed module (V208, Fig. 3) has already been developed with 6 phase lock loop (PLL) circuits to transmit the clock and data signals for each link over user defined P2 pins on the VME backplane using LVDS multi-drop pairs. An overlay module is installed on the rear of the backplane to bus each clock and data pair to all slots on the VME backplane. A simple interface to a gate array on each module is then possible.

High Speed Processing of I/O Signals

With 4 400 MSPS analog to digital converters resident on the V301 module, high speed processing of signals is essential. The Zynq family of SoCs is fully capable of providing the required processing speeds.

Ethernet Connections

Two or more Ethernet connections are required to be available on each module. One will be used for communication with the higher-level control system using standard Ethernet protocols and the other will be used to provide a private data distribution network.

Embedded Operating System with Resident Control System Software Objects

The latest low-cost SoC components like the Xilinx Zynq family have made it possible to easily and inexpensively design modules that incorporate an ARM processor running an embedded Linux operating system

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

(OS) that is tightly coupled on the same integrated circuit to high speed programmable logic. The eRHIC real-time hardware architecture plans to take full advantage of this capability.

Modules will be designed using the Zynq or similarly available SoC. The on-chip ARM processor will run a version of Linux that will be responsible for executing the low-level control system software objects (called ADOs in the RHIC control system, and IOCs in EPICS).

A dedicated Ethernet connection on each module will be used to communicate with higher-level software using the currently available software infrastructure.

The V301 prototype has been designed to perform as described here, and is currently operating exactly as intended.

Gigabit Communication Between Modules

In order to develop a system with building block modules that can be used in a variety of system applications, a method for high speed communication between modules is essential. Parallel busses like VME are no longer adequate for many systems, and it is unreasonable to think that a universal module can be designed to accommodate the I/O needs for every system.

The specific details for satisfying this requirement are a work in progress. Some options under consideration include:

- Custom designed Gbit link between modules using external board-to-board connections
- Ethernet hardware protocol with custom defined software packet structure to eliminate Ethernet layer overhead but allow commercially available Ethernet components and network switches to be used for private networks. This approach has successfully been used for the RHIC 10 Hz global orbit feedback system.
- Ethernet or PCIe communication via a backplane like openVPX through a custom designed switch module located in the center slots per the openVPX standard.

The V301 includes an on-board connector with direct connection to the Zynq MGT multi-Gbit pins (Fig. 2). This interface will be used to prototype high-speed serial communication between modules.

Using Commercially Available Modules

The present RHIC Control system utilizes many commercially available VME modules. The ability to use commercially available modules in eRHIC is highly desirable. This is another reason why a multi-slot chassis system architecture is planned.

Sharing of Modules

Similar to using commercially available modules, there is also a strong desire to be able to share modules between groups within the C-A complex as well as with other institutions. In order to make this viable, standards must be developed and adhered to.

CONCLUSION

A general architecture for eRHIC real-time hardware has been designed and a very successful prototype system using existing VME chassis has been developed.

The basic hardware architecture, which uses a SoC with embedded processor running an operating system, with the processor being tightly coupled to the on-chip gate array, is not unlike other recent developments in the C-A complex and other facilities.

Previous SoC system developments like the C-A low-level RF system [7] and the 10 Hz global orbit feedback system [8] have been constructed in self-contained chassis. The main difference with the proposed eRHIC architecture is the use of multi-slot chassis in order to house the large number of modules that are anticipated, to provide easy maintenance and board swapping, to allow the integration of commercially available modules, and to share hardware module designs with other institutions.

The present prototype system has been based on VME, mainly because VME chassis are abundant at the C-A complex. A more modern chassis and backplane design such as openVPX will likely be selected to provide Gbit serial link communication between modules via the backplane. However, of the chassis backplane configurations that have been researched to date, none contain an architecture that perfectly suits the needs for eRHIC and accelerator systems in general. The best option may be to develop a custom backplane, possibly using the VPX standard, which provides the capability to customize backplanes.

Since sharing modules with other institutions is highly desirable, this would be a perfect time for the accelerator hardware community to collaborate our efforts and together select a common backplane design.

REFERENCES

[1] R. Frankel, “Informal Datacon System Report,” BNL AGS Division Tech. Note No. 88, December 6, 1971

[2] V.J. Kovarik, “Signal Specifications and Transceiver Operation for DATACON II System,” BNL EP&S Division Tech. Note No. 58, March 14, 1973

[3] J. Morris, T. D’Ottavio, “Status Report for the RHIC Control System,” ICALEPCS 2001, San Jose, CA (2001)

[4] L.T. Hoff, “Real-Time Scheduling of Software Tasks,” ICALEPCS 1995, Chicago, IL (1995)

[5] B. Oerter, C.R. Conkling, “Accelerator Timing at Brookhaven National Laboratory,” PAC 1995, Dallas, Texas (1995)

[6] T. Kerner, C.R. Conkling, B. Oerter, “V123 Beam Synchronous Encoder Module,” PAC 1995, Dallas, Texas (1995)

[7] T. Hayes, K.S. Smith, “A Hardware Overview of the RHIC LLRF Platform,” PAC 2011, New York (2011)

[8] R. Michnoff, et. al., “RHIC 10 Hz Global Orbit Feedback System,” PAC 2011, New York (2011)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

A MODULAR APPROACH TO ACQUISITION SYSTEMS FOR FUTURE CERN BEAM INSTRUMENTATION DEVELOPMENTS

A. Boccardi, M. Barros Marin, T. E. Levens, B. Szuk, W. Vigano, C. Zamantzas
CERN, Geneva, Switzerland

Abstract

This paper will present the new modular architecture adopted as a baseline by the CERN Beam Instrumentation Group for its future acquisition system developments. The main blocks of this architecture are: radiation tolerant digital front-ends, a latency deterministic multi gigabit optical link, a high pin count FMC carrier used as a VME-based back-end for data concentration and processing. Details will be given on the design criteria for each of these modules as well as examples of their use in systems currently being developed at CERN.

THE CERN BEAM INSTRUMENTATION GROUP

The Beam Instrumentation (BI) Group is responsible for designing, building and maintaining the instruments that allow observation of the particle beams and the measurement of related parameters for all CERN accelerators and transfer lines, comprising ultra-relativistic and non machines, circular and linear accelerators and even an antiproton decelerator. Each machine has very specific characteristics and requirements and therefore rarely the same instrument can be deployed on several of them. Typical beam parameters to be measured and monitored are: position, loss, intensity (total and per bunch), tune and chromaticity, luminosity, transfer lines matching, transverse and longitudinal profiles. The BI group decided to adopt a modular architecture for its acquisition system to promote the reuse of electronics and code, to simplify the maintenance and the design of this large variety of instruments.

Despite the details of each implementation can be very different the generic architecture of an instrumentation acquisition system is in the majority of the cases similar (see figure 1): the signal from the sensor is first conditioned and then digitized before being in-line processed.

Figure 1: Typical structure of an instrumentation acquisition system. The signal from the sensor is conditioned, digitized and in line processed.

For the majority of BI systems the processing is performed in a back-end unit, installed in a radiation safe area, even for the cases in which the digitalization is performed at front-end level. This allows the standardization of the back-end itself. That is one of the main block of this modular architecture, and reduce the complexity of the rad-tolerant electronics eventually required, with a consequent increase of reliability.

A problem common to all the system with a remote digitalization is the transmission of the data from the front-end to the back-end with a constant delay to avoid the need of continuous recalibrations at each reset of the system. This is particularly difficult for systems requiring large bandwidth and therefore complex serialization protocols relying on synchronization and PLLs. BI decided to adopt the multi-gigabit and latency deterministic optical link designed in the CERN PH-ESE group as the second main block of its new design toolkit, the third one being a digital front-end module implementing this link in a rad-hard version for all the systems where the digitalization stage is not only remote but also exposed to radiations.

Those three elements, on which the system specific blocks plugs, are described in the next chapters.

THE BACK-END

The board designed to be the standard back-end for the BI applications is the VFC-HD [1]. It is an FPGA-based 6U VME 64x module with a high pin count FMC (VITA 57) slot, the possibility to connect to custom VME rear transition module (RTM), 6 small form factor pluggable (SFP) slots, 4 of which dedicated to user applications and 2 to system ones, and on board DDR3.

The board was specified to be an FMC compliant carrier for its users to be able to use commercially available mezzanines. The decision to adopt the high pin count standard is linked to the need to use the latest generation of fast ADCs and DACs. Nowadays those adopt multi gigabit serial communication for their interfacing and therefore requiring the use of lines not defined in the low pin count standard of the connector.

The RTM was specified to suit the needs of systems requiring part of their connectivity to be plugged in the back of the crate.

The decision for the VME standard came from the need of being able to integrate the new board with existing systems. BI made in the past years extensive use of custom VME crates for its developments. In those crates the beam synchronous clocks and triggers are distributed by a timing receiver over the back plane and the P0

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

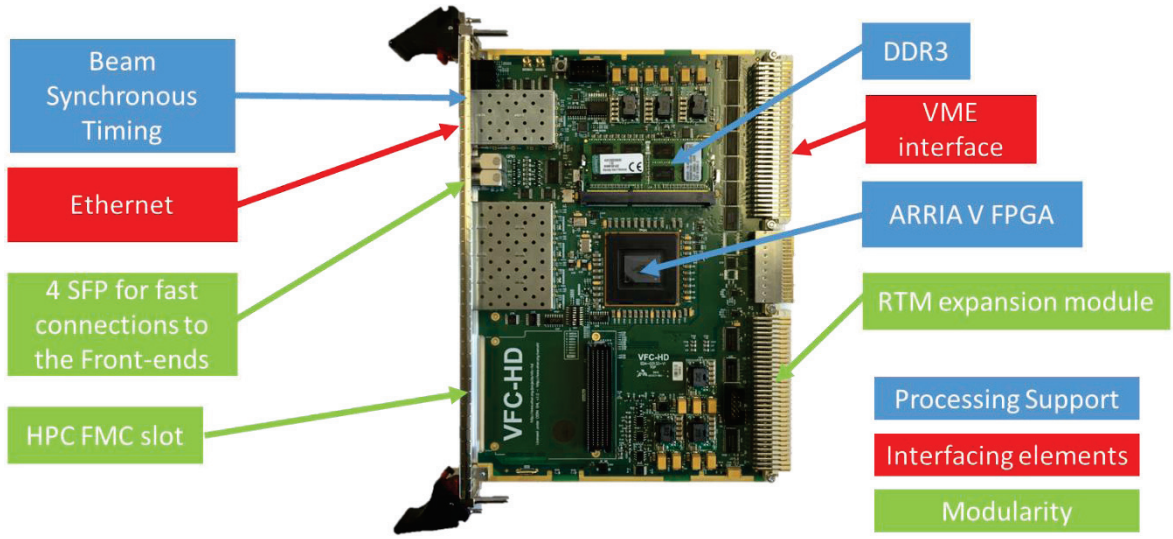


Figure 2: The VFC.

connector to the other VME modules. This led also to the decision to keep the BI specific definition of the P0 signalling in for the VFC-HD.

New systems on the other hand will be based on standard crates, where no timing distribution is implemented on the back plane. Each VFC-HD will need to receive and decode by itself the beam synchronous timing via an optical link, the same way was done by the timing receiver for the whole crate. The first of the system SFP connections is dedicated to this purpose.

In view of possible changes in the timing distribution protocol the VFC-HD was designed with White Rabbit [2] node capabilities, as this is one of the most likely to be new timing distribution protocol.

The VFC-HD can also work in standalone mode to suit highly distributed systems like for example the tune or the instability trigger ones. The board for this purpose can be accessed via an Ethernet connection implemented using the second of the system SFPs.

The 4 application SFPs are meant to connect the back-end to remote digital front-ends like the GEFE(see next chapters). Those SFPs are connected to the FPGA multi-gigabit transceivers in a scheme that takes into account the clocking requirements for the implementation of the chosen latency deterministic link (see next chapter). A single VFC-HD can connect to up to 4 of those front-ends without the need of a FMC optical connection module.

The decision to use on-board DDR3 memories was not immediate. Indeed this is the second high pin count carrier developed in BI: the first one used SRAM chips. The need for high density memories comes from few systems requiring long histories of raw measurements for post mortem analysis or specific beam studies. This requirement came after , i.e. the FPGA model was already chosen i.e. the ARRIA V GX. Neither this model nor the pin compatible GT version supports the fly-by topology used in the DDR3 DIMMs, but to limit the redesign effort and the cost, an important factor for the major users, was decided not to change it. The total memory on board is 512MB, considered enough, and could be further extended to 2GB if the compatibility with the new "TwinDie" chips from Micron will be confirmed.

THE LATENCY DETERMINISTIC LINK

The latency deterministic link chosen by BI for the communication between its high bandwidth frontends and the VFC-HD is based on the GigaBit Transceiver (GBT) link [3], developed by the CERN PH-ESE group. For non-radioactive environments, this link can be implemented in FPGAs in conjunction with commercial optical modules. For systems that are exposed to radiation, the GBT link is implemented using dedicated rad-tolerant chipsets, comprising an encoder-serializer/decoder-deserializer chip, the GBTx, and a custom optical module, the VTRx.

The latency determinism was a key parameter in the choice of this link. It allows to look at the link as a transparent pipe for the data with a constant delay that can be calibrated just once at the commissioning of the instrument instead of at each reset of either of the two communication ends.

Another element that made of this link the chosen one in BI is the reliability, indeed it features a protocol (GBT Frame) with a robust error detection and correction algorithm (Reed-Solomon) without the need of further interventions of the users.

Additionally, the GBT link also offers high payload bandwidth, 3.2Gbps for a 4.8Gbps communication, the availability of the radiation tolerant chipset, and finally the possibility to use as base clock the beam synchronous 40MHz of the LHC or SPS, making it available as recovered one on the front-end side.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

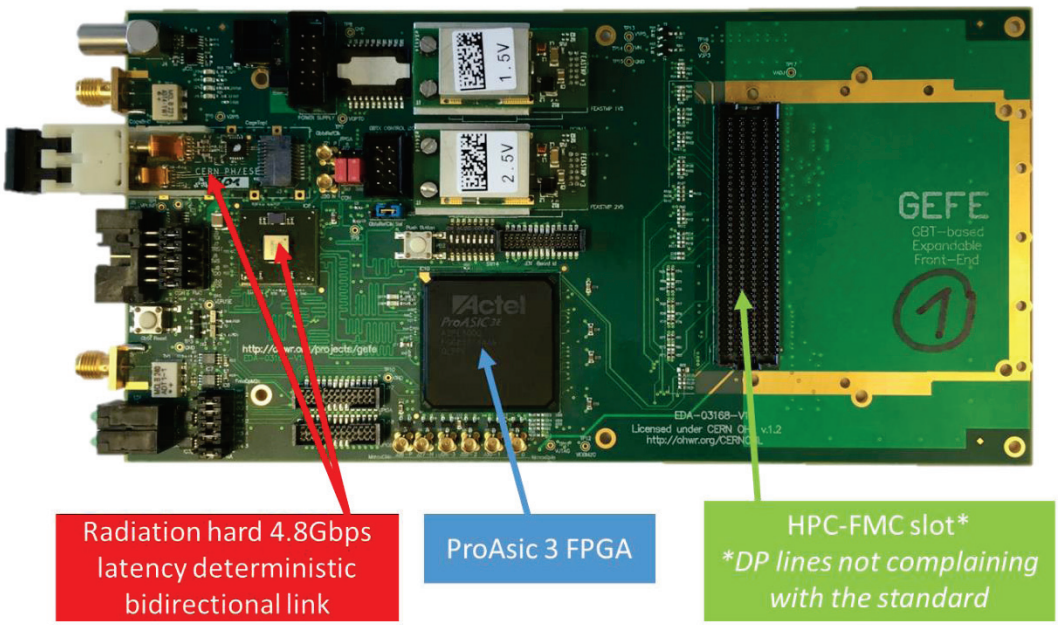


Figure 3: The GEFE.

THE RADIATION TOLERANT DIGITAL FRONT-END

The GBT-based Expandable Front-End (GEFE) [4] is a multipurpose FPGA-based radiation tolerant board designed by BI to be the new standard FMC carrier for radiation tolerant digital front-end applications. Its intended use ranges from fast data acquisition systems to slow control installed close to the beamlines. In the case of the GEFE the choice of the FMC format is for consistency with the VFC-HD and not in view of the use of commercial mezzanines.

It is important to note that the GEFE's FMC connector is for high pin count mezzanines, but that the multi gigabit lines are not implemented in accordance with the standard.

The main purpose of the GEFE is the implementation of the interface between the back-end and the digitalization module.

All the components chosen for this board are either radiation tolerant by design, like the GBTx chipset [3] and the power supplies [5] designed in the PH-ESE group, or have been qualified for a total ionising dose (TID) up to 75krad, a limit imposed by the ProAsic3 FPGA from Microsemi.

The presence of an FPGA limits the TID but increases the flexibility of the board. It allows an easy implementation of the glue logic required to interface the rest of the front-end system to the GBTx chip, to receive and decode commands and configurations from the back-end and to manage the clocking scheme.

The last important factor that was taken into account during the design phase was the size of the board. The aim was to have a module as compact as possible to ease the integration with other modules close to the beam, where often the space is limited. The GEFE is a dense 200mm by 100mm board.

SYSTEM EXAMPLES

This section introduces examples of systems already using or planning to use the proposed architecture.

The LHC DC Beam Current Transformer (BCT) and the AWAKE (a new experimental line for wake field acceleration) Beam Position Monitoring (BPM) systems both use the VFC-HD and have their sampling in a remote location not exposed to radiation. Both implement their communication between the front-end and the back-end via copper cables of up to 1km. The DC BCT uses slow sampling ADCs. The data rate is compatible with the transmission bandwidth and every sample is transmitted to the surface. The AWAKE BPM system samples at 40Msps. The data rate would be too high for the transmission of all the samples to the surface. For this reason the front-end implements a threshold based auto triggering algorithm to implement data reduction. The relevant data is buffered and transmitted with a 1Msps protocol taking advantage of the very low repetition rate of the experiment. Both those systems use custom, non-standard, digitizer front-end systems and FMC mezzanines on the VFC-HD.

The Beam Loss Monitoring (BLM) system based on diamond detectors as well as the LHC and SPS fast BCT systems will use a fast sampling commercial ADC FMC mezzanines, the FMC1000 from Innovative, plugged directly on the VFC-HD for their acquisition systems. A similar solution is under study for the new LHC interlocked BPM system, where the sampling speed will be most probably at 4Gsps.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The standard LHC BLM system has its digitalization modules in radiation exposed areas, but at the design time the GEFE was not yet specified. The front-end of this system integrates in the same board the analogue and digital part. The structure of the digital section of such system as well as its form factor is similar to that of the GEFE. It makes use of an anti-fuse FPGA from Actel to implement the digitalization control logic and the interface to the optical link based on the Gigabit Optical Link (GOL), the GBT predecessor with a 640Mbps payload bandwidth. This front end has been tested and qualified including the firmware for a total dose of 70krad. Two of such front-ends with redundant links will connect to a single VFC-HD in a future upgrade of the back end part of the system.

The new Multi Orbit POsition System (MOPOS) of the SPS will be the first instrument to use both the VFC-HD and the GEFE modules. The digitalization is performed on a custom mezzanine equipped with two 4 channel rad-hard 12bit ADCs @10Msps. The sampling clock will be derived from the beam synchronous one recovered by the GBTx. All the samples will be sent interleaved from the GEFE to the VFC-HD where they will be received as if the ADC would be on board thanks to the latency determinism of the link.

REFERENCES

[1] A. Boccardi *et al*, “Ongoing electronic development in the CERN Beam Instrumentation Group: challenges and solutions for the measurement of particle accelerator beam parameters”, TWEPP2012, Oxford, United Kingdom, September 2012

[2] The White Rabbit project page: http://www.ohwr.org/projects/white-rabbit

[3] The GBT project web page: https://espace.cern.ch/GBT-Project/default.aspx

[4] M. Barros Marin et al, “The GBT-based Expandable Front-End (GEFE)”, TWEPP2015, Lisbon, Portugal, September 2015

[5] The FEASTMP (DCDC converter) project web page: http://project-dcdc.web.cern.ch/project-dcdc/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE GLOBAL TRIGGER WITH ONLINE VERTEX FITTING FOR LOW ENERGY NEUTRINO RESEARCH*

Guanghua Gong, Hui Gong
Tsinghua University, Beijing, China

Abstract

Neutrino research is of great importance for particle physics, astrophysics and cosmology. A new global trigger scheme with online vertex fitting has been proposed, aiming at the ultra-low anti-neutrino energy threshold as down to 0.2MeV which is essential for the study of solar neutrino and supernova elastic scattering neutrinos on burst. With the scheme, the time of flight difference of photons fly through the liquid media from the interaction point to the surface of central detector can be corrected online, the trigger window to cover the whole spread of a specific neutrino generated photons can be significantly reduced which lessen the integrated dark noise introduced from the large amount of PMT devices hence a lower energy threshold can be achieved. The scheme is compatible, flexible and easy to implement, it can effectively help the low energy neutrino research topics.

INTRODUCTION

Neutrino research is a very popular and important topic for particle physics, astrophysics and cosmology. Several new neutrino experiments are now under construction or design such as the JUNO [1] (Jiangmen Underground Neutrino Observatory), a multi-purpose neutrino experiment, and CJPLNE [2] (China JinPing underground Laboratory Neutrino Experiment), a water-based LS neutrino experiment focusing on the solar neutrino and geo neutrino topics. The central detectors of those neutrino experiments consist large sphere or cylinder vessel to contain the giant target material which are 20kt Liquid Scintillator for JUNO and few thousand tons of water-based Liquid Scintillator for CJPLNE. PMTs are applied for photon detection due to the large sensitive area and mature technology. Around 16000 20inch PMTs will be installed for JUNO while around 15000 8inch PMTs are estimated to be deployed for CJPLNE.

A neutrino that interacts in the fiducial volume will release charge particles to further generate scintillation light. The light distribute evenly in all directions and finally be captured by the surrounding PMTs and produce electrical signals. The charge and position information of the related event could be further analysed from the PMT signals.

TRIGGER SYSTEM

The information related to one event must be collected to be distinguished by trigger system which could base on the number of PMTs that have generated signals (multiplicity trigger), or base on the total charge information (energy sum) [3]. Since the multiplicity trigger scheme requires only single hit bit information which is very easy to generate, transmit and process, it is the preferred baseline trigger method, especially for experiments that applies tens of thousands of PMTs.

For 20kt LS detector, the physical dimension will be 37meters in diameter, thus the light generated at the edge of the detector will take around 200ns time of flight before reach the PMTs installed on the opposite direction. To cover the TOF time and the slow scintillation components of the LS, the trigger system must open a 300ns time window to cover the full spread of all information from the same event.

PMT generates signal pulses due to thermionic emission which is called dark noise that can't be distinguished from the normal signal. Those dark noise will also be collected during the trigger window that could cause a coincident event. The correspond event rate is a function of trigger window width, dark noise rate and the number of PMT. Figure 1 shows the calculation result of dark noise caused coincidence rate compare to the physical event rate with parameters of 300ns trigger window, 50KHz dark noise for the new developed MCP PMT and 15000 tubes.

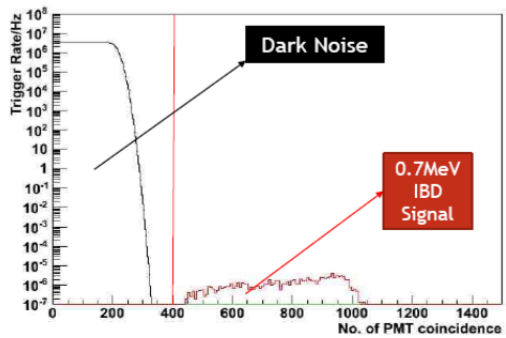


Figure 1: Event rate of PMT dark noise.

By setting the multiplicity threshold to around 400, the dark noise coincidence event can be easily separated from IBD neutrino events that have a minimum deposit energy of 0.7MeV. But for solar neutrino and supernova elastic scattering neutrino physics, many interesting phenomena are located at the energy range as low as 0.2MeV, the multiplicity distribution of such low energy event heavily overlaps with the PMT dark noise.

While approaching the low energy range, the trigger rate increases significantly. Since flash ADC is applied to

*Work supported by the National Science Foundation of China (No. 11275111), the State Key Laboratory of Particle Detection and Electronics.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

record the waveform of each PMT hit, the increased event rate requires high bandwidth DAQ and make the readout electronics difficult and expensive.

A possible solution to the problem could come from the awareness of the vertex position information in the trigger system. The vertex could be used to correct the propagation time difference to concentrate the correlated PMT hits into a much narrower time region. Then a smaller trigger window could be used which reduces the PMT dark noise and a much lower energy threshold can be achieved.

Furthermore, the threshold could even be adapted to the vertex position, which could better handle the issue of light attenuation effect, spatial effect of energy response and radioactive background at the edge of the scintillator volume due to supporting material.

VERTEX FITTING CONCEPT

To calculate the vertex position, a charge weighted mean position algorithm is widely used in offline analyse. To implement a similar algorithm in the trigger system is very challenging, and the readout electronics will also need to implement real time algorithm to extract the charge information for each hit, otherwise it could just simply transmit raw waveform data for triggered events.

Unlike direct calculation, the vertex fitting method tests the PMT information with all possible positions and finds the most likely one among them. The detailed process contains the following aspects:

- Divide the whole detector volume into certain number of blocks. Since each block is treated as one position, the time difference caused by the dimension of the block should be negligible compare to the trigger window; on the other hand, small block size will cause an unacceptable block quantity. The size of the block should be optimized to compromise the two influences.
- For each block, a TOF correction map can be calculated and applied to the original PMT signals respectively. For events located inside the block, the correction map could approximately compensate the propagation time and merge the widely spread PMT hits to a much narrower time region thus a small trigger window could be applied.
- For each block, a threshold that is adapted with the block position and radioactive background is compared to generate the local block trigger signal.
- All local block trigger signals are logically "OR"ed to generate the final trigger.

The TOF correction, multiplicity calculation and comparison are identical for all blocks that can be handled simultaneously by parallel process architecture.

The concept of vertex fitting is illustrated in Figure 2.

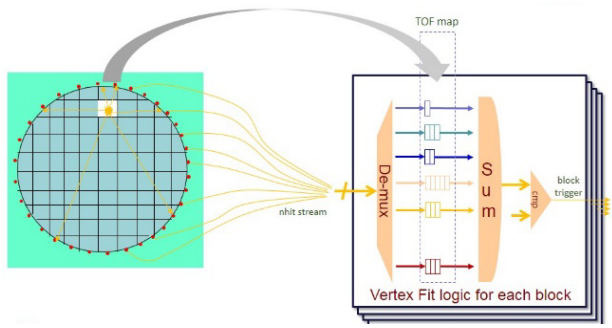


Figure 2: Vertex fitting concept.

SIMULATION AND FEASIBILITY

Simulation result in Figure 3 shows the dependency of event position and PMT hit time distribution in a 37 meter diameter sphere vessel.

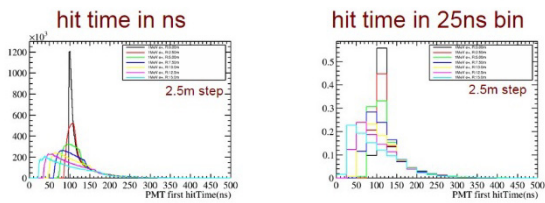


Figure 3: Hit time distribution with vertex position.

The time distribution of events with distance difference of 2.5 meters from the centre can be distinguished clearly even with 25 ns time resolution.

The improvement of energy threshold by reducing the trigger window is shown in the simulation result in Figure 4. With 50 ns trigger window, the PMT dark noise can be removed from 0.2MeV event.

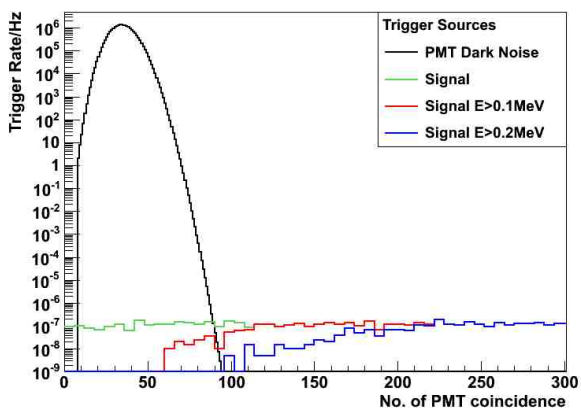


Figure 4: Event rate of PMT dark noise.

From the simulation, 5 meter is taken as a reasonable block size. The time difference inside the block itself is about 35ns, acceptable for 50ns trigger window.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

For a 35 meter diameter sphere, the total block quantity is 180.

To avoid events spread cross the 50 ns trigger window boundary, the hit signals from all PMTs will be sampled at 25 ns interval, and two consecutive hit samples are summed to form a 50ns trigger window for multiplicity comparation.

The TOF correction thus can be handled with time granularity of 25 ns. This could be easily implemented by delaying the real time hit information for certain 40MHz clock cycles via register/registers/FIFO.

SYSTEM ARCHITECTURE

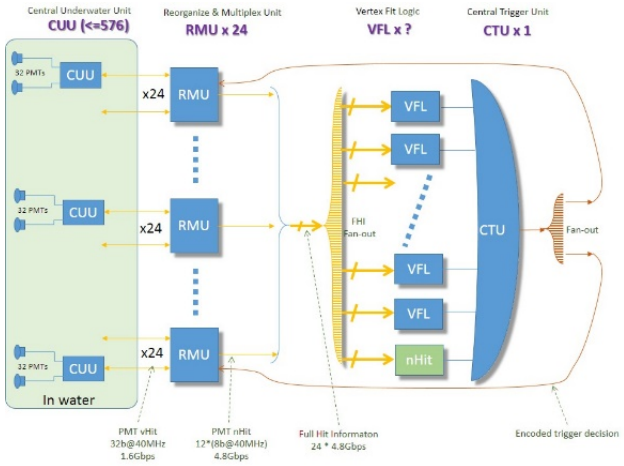


Figure 5 : Vertex fitting system architecture.

The system architecture is shown in Figure 5, the detail of all parts are described below:

PMT Grouping

It is neither possible nor worthy to collect hit signal of each individual PMT. The front end electronics (Central Underwater Unit) interface with 32 PMTs and thus can naturally provide the 32 hit signals at 40 MHz frequency.

For underwater experiment, the PMTs are connected to CUU in an interleaved arrangement to avoid group failure. The CUU sends each individual hit bit in 32 bit vector (called as *vHit32*) via a dedicated optical link. The link speed is 32 bit@40 MHz = 1.28 Gbps and 1.6 Gbps with 8 b/10 b encoding.

The individual hit bit for each PMT could be used for PMT status monitor and hit rate histogram.

Reorganize and Multiplex Unit

Links from 24 CUU are connected to one RMU (Reorganize and Multiplex unit). The hit from the most neighbouring 64 PMTs, forming an 8*8 square slice, are summed up as 8bit digit number called *nHit64*. The 12 nHit64 of the RMU are further multiplexed to transmit in a single optical link. The link speed is 12*8bit@40MHz = 3.84Gbps and 4.8Gbps with 8b/10b encoding.

The system architecture supports up to 24 RMU, which equals to 18432 PMT that is beyond the maximum number of PMT to be used for both JUNO and CJPLNE.

Vertex Fitting Logic/Unit

The output of 24 RMU composes the 24*4.8Gbps full hit information that are used for vertex fitting.

In each vertex fitting logic, the FHI are firstly de-multiplexed to 288 nHit64 channel, the TOF correction is then achieved by delaying each nHit64 via registers for certain cycles as specified in the correction map. The corrected value is then summed up and compare with a threshold value to generate a block trigger.

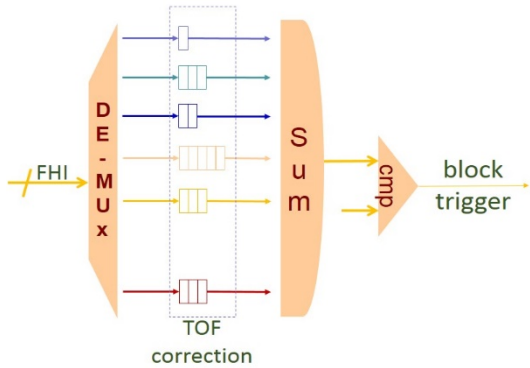


Figure 6: Vertex fitting logic.

The fitting logic has been evaluated on Xilinx V5LX220 device. 288 Delay lines with 8bit wide, 12 byte deep and configurable output tap together with accumulator and comparator utilize less than 5% of the available resource. Thus more than 10 fitting logics could be integrated in single device to reduce the number of vertex fitting unit to around 18 that could be settled in a single crate.

The block triggers of multiple VFL inside the VFU is encoded and transmitted via a single copper cable to central trigger unit.

FHI Multi-sharing

Since as much as 180 vertex fitting logics are running in parallel and they all require the full hit information. FHI multi-sharing is to duplicate the data stream transmitted by the 24*4.8 Gbps optical links for each of these 180 vertex fitting logic, which is done in several stages:

- For each RMU, the identical output is transmitted via several optical links to provide a duplicate factor of 4 to 12 according to the channel number of the optical transmit module.
- The de-multiplexed FHI could be routed to 10 VFL inside the same device which provides a duplicate factor of 10.
- Each optical link can be split into several branches which only depends on optical power budget of

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

the transmitter/receiver module. A factor of 4 to 8 can be easily achieved.

Combining all these stages, a multi-sharing factor of 160 to 960 is achievable which leaves great flexibility for further upgrade.

Central Trigger Unit

Block triggers from all VFL are logically "OR" in the central trigger unit for final trigger. The CTU also takes responsible to provide interface for calibration system or veto system. Diagnostic features like random/periodic trigger can be generated here as well as the flow control features like throttle, scale.

The fitted vertex position, timestamp, multiplicity number and trigger count information for each event can also be packed by CTU and saved for crosscheck.

The central trigger signal is fan-out and transmit back to RMU and further to all CUU.

Trigger Latency

Trigger latency affects the buffer requirement of the front end electronics. The processing in the trigger system is simple and fixed, the majority of the latency comes from the link delay between modules. 200 meter of fibre between CUU and RMU will end up with trigger latency about 3 us which is far less than the acceptable value.

STATUS AND CONCLUSION

New trigger scheme with online vertex fitting feature has been designed for next generation underground neutrino experiments that are under construction in China. Technical issues has been explored and the complete system structure has been designed. The system is feasible to achieve energy threshold of 0.2MeV that is of importance for related physics topics.

ACKNOWLEDGEMENT

This work is supported by the National Science Foundation of China (No. 11275111). The authors would like to thank Xiaoshan Jiang, Zhe Wang for the discussion.

REFERENCES

[1] JUNO CDR
[2] CJPLNE website and LOI, http://hep.tsinghua.edu.cn/CJPLNE/jinping.pdf.
[3] H. Gong, et al., "Design of the local trigger board for the Daya Bay reactor neutrino experiment" , NIMA, Volume 637, Issue 1, 1 May 2011, Pages 138–142.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

MAPPING DEVELOPMENTS AT DIAMOND

R. Walton, A.W. Ashton, M. Basham, P.C.Y. Chang, T. Cobb, S da Graca, A.J. Dent, J. Filik
M. Gerring,C. Mita, J. Mudd, C.M. Palmer, U. Pedersen, P.D.Quinn, , N. Rees
Diamond Light Source, Oxfordshire, UK

Abstract

Many synchrotron beamlines offer some form of continuous scanning for either energy scanning or sample mapping. However, this is normally done on an ad-hoc beamline by beamline basis. Diamond has recently embarked on an ambitious project to define how to implement continuous scanning as the standard way of doing virtually all mapping tasks on beamlines. The project is split into four main areas: 1) User interfaces to describe the mapping process in a scientifically relevant way, generating a scan description that can be used later; 2) The physical process of scanning and coordinating hardware motion and detector data capture across the beamline; 3) Capture of the detector data and all the associated meta-data to disk, deciding and describing the layout of the file (or files) for the main use cases; 4) Display and analysis of live data and display of processed data. In order to achieve this common approach across beamlines, the standard software used throughout the facility (Delta Tau motor controllers, EPICS, GDA and DAWN), has been built on.

INTRODUCTION

With the increase of flux obtained with most modern 3rd generation synchrotrons, collecting maps of surfaces is becoming increasingly popular. This process involves moving a sample in front of a focussed x-ray beam, then recording various interactions of this beam with the illuminated part of the sample. Interactions include, but are not limited to, fluorescence, diffraction, absorption and scattering, with each requiring specific detectors to be available on a beamline. Scans are often performed continuously as detectors are triggered and maps containing millions of individual measurements are routinely collected at specialist beamlines. The high flux available means that these scans can be performed increasingly quickly, providing significant technical challenges in all areas, from motion control and detector readout to processing and visualisation.

At Diamond Light Source there are many beamlines which can perform mapping experiments. Dealing with these challenges on a case by case basis has become unmanageable. At the time of writing we are six months into a cross beamline, cross support group project to standardise and modularise as many elements of this process as possible. As well as providing improved functionality and user experience on these beamlines a key aim is to make the solution supportable with as little effort as possible. Many aspects of the solution are applicable to techniques other than mapping (a line scan is just single row map scan) and we expect the impact of this project over the coming years to reach many of Diamond's beamlines.

To give the best chance of producing a general solution, the project targeted five beamlines with similar but unique mapping requirements. These include traditional XRF (X-ray Fluorescence) mapping [1], XRD (X-Ray Diffraction) mapping, Absorption mapping, Coherent Diffraction mapping [2] and ARPES (Angle Resolved Photoemission Spectroscopy) mapping beamlines.

At the heart of this project is a commitment to standardise the way data is handled with a new standard data flow, standardise the way continuous scans are controlled with a new EPICS control layer, and to improve the user interface and speed of collection. The remainder of this paper highlights these high level design decisions before presenting further details of the system and reporting progress.

STANDARD DATA FLOW

To simplify post processing and analysis it is important that the multidimensional and in some cases multimodal data, along with all the appropriate metadata such as experimental conditions, are stored in a standard format. Diamond uses the NeXus format [3] backed by HDF5 [4]. HDF5 performs very well, especially with the Lustre and GPFS file systems used at Diamond. Unfortunately, HDF5 cannot be read as it is written and so in order to provide live data processing and visualisation separate data paths have been maintained. To simplify this situation the HDF group was funded by Diamond, ESRF and Dectris to extend HDF5 to allow access to data as it is written. This extension is known as SWMR which stands for Single Write Multiple Read [5].

The access to live descriptive data SWMR affords greatly streamlines processing and visualisation as the same software used offline can, with some work, be used online. Further, because data can be read almost anywhere, it can be processed using tools running on a cluster which are available for post processing, allowing for sophisticated, and computationally expensive processing to be achieved in close to real time. Processed data will also be written with SWMR, providing the visualisation software live access to both processed and raw data.

MALCOLM EPICS CONTROL LAYER

The GDA data acquisition software has good support for dynamically conducting arbitrary scans, but when performance is important, such as with mapping

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

experiments, hardware control must be pushed down to the motion controller. Historically this has been done with a variety of solutions, but Diamond is converging on a solution that supplements Delta Tau motion controllers with a Zebra triggering box [6, 7]. Currently Diamond's GDA server [8] communicates with this range of hardware directly or via EPICS [9]. Although these solutions work they are expensive to maintain, and not that scalable.

To take best advantage of the Delta Tau with Zebra solution, a new EPICS control layer between the high level acquisition server and the low level EPICS software has been created. Malcolm aims to abstract some of the complexity involved with controlling multi-purpose EPICS drivers into simple devices that have parameters that would make sense to an end user [10]. For instance, an AreaDetector chain of 1 driver and 7 plugins controlling a detector would be modelled in Malcolm as 8 low level devices, and one top level device that captures the attributes that are relevant for a particular mode of operation, and orchestrates the low level devices. A set of motors, a trigger box, and a spiral scan would all be Malcolm devices, all of which implement a subset of the same configure-run state machine shown in Figure 1. This allows nested scans to be easily created, as all devices obey the same interface.

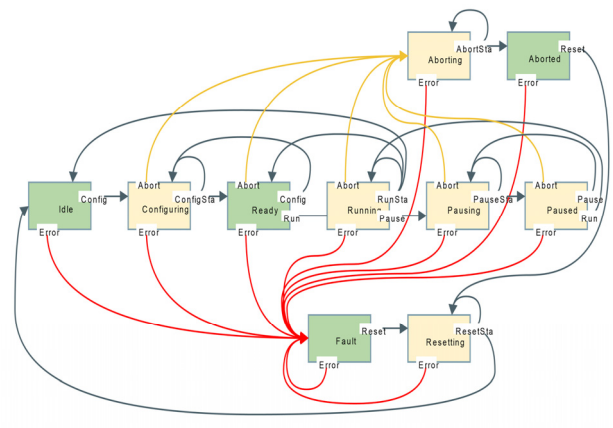


Figure 1: Malcom state machine.

The prototype system is written in Python and uses a JSON presentation layer over ZeroMQ or websocket transport. At the time of writing, work is progressing on device definitions for Diamond's I05 ARPES branchline.

When the architecture is proven, pvData over pvAccess will be added to the Python server, and a C++ port written to allow AreaDetector drivers and plugins to provide this interface natively. The hoped final outcomes are EPICS V4 transport and presentation layers, and device definitions for a large range of detectors, trigger boxes, motors, and complex scan types.

STEPS DURING AN EXPERIMENT

The software will help the user explore a sample's surface interactively by building up and visualising data from a number of scans and microscope images. Each scan might use different techniques, detectors or processing, but all will be visualised in the same framework. Figure 2 shows an example client.

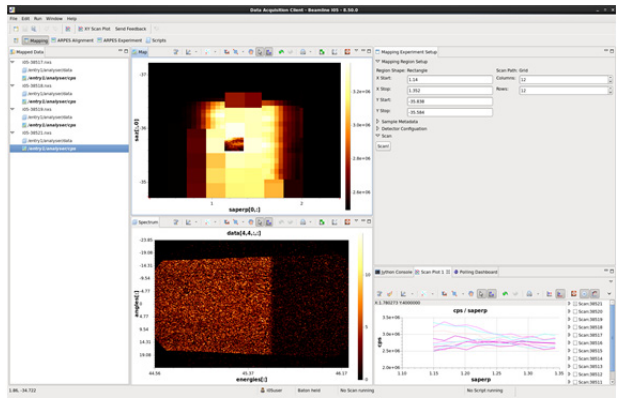


Figure 2: GDA client showing the new *mapped data view* and experiment setup pane on the I05 ARPES beamline. The *Mapped Data View* shows all data which is available for plotting, in this case 4 different resolution maps of the sample. The *Map View* shows all available maps overlaid, and new regions can be selected for scanning visually in this section. The *Spectrum View* shows the spectrum or image collected at the selected point on the Map View. Finally, the *Mapping Experiment Setup View* is an auto-generated view based on the experimental requirements of the beamline.

In most cases a user will start a session by taking a visual image of the surface of a newly mounted sample. This image will then form the basis for defining regions to scan over. A scan comprises a series of events, most of which are performed concurrently as data is collected. These events, with the component responsible for each indicated in italic, are:

1. **User defines scan parameters**. The user uses the *scan parameter editor* in concert with the *mapped data views* to describe a scan including detectors and, in many cases, a region over which to map and then requests that the *acquisition server* orchestrates a scan. Scans can also be performed using a command line interface suitable for scripting.
2. **Trigger and coordinate scan.** Having received (a possibly queued) request to perform a scan the *acquisition server* component connects to the *processing* and mapped data view components to inform them a scan will start and to await progress updates. It then starts the collection. The acquisition server is built into Diamond's GDA software, but could be easily deployed within other Java products.
3. **Sequence motion and trigger detectors.** Simple step scans will be sequenced by the acquisition server directly. Scans requiring continuous motion will use

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

the new *Malcom middle layer* software described above which provides a layer above *Epics AreaDetector*, Diamond's *Zebra* box used for triggering, and a motor controller.

4. **Record data and metadata to file**. The acquisition server writes a NeXus file that contains links to the HDF5 files written by Epics AreaDetector and Zebra:
   a. The acquisition server's *NeXus file writing* component writes the NeXus file using SWMR compliant HDF5 libraries.
   b. Epics AreaDetector and Zebra must write data using SWMR in any arbitrary order given knowledge of the scan's shape and path.
   c. *Remote Dataset* components make possibly down-sampled slices of data from disk available offsite and from workstations which, because SWMR works only across POSIX compatible file systems, cannot access the data directly.
5. **Process data**. The live processing component optionally creates a new NeXus file to store processed data; this will include a link to the original NeXus file containing only the raw data.
6. **Visualise data.** The user will see the raw data, and optionally the processed data, appearing live in the mapped data view. This is the same view available offline in Diamond's DAWN software and provides tools for interactive analysis.

COMPONENT DETAILS AND PROGRESS

At the start of the project a feasibility study showed that SWMR would work well in this. The HDF Group is currently in advanced stages of merging and testing SWMR along with other new features into the trunk of the HDF5 code base and appears to be on schedule for the HDF5 1.10 release by spring 2016. Despite the HDF5 SWMR feature only being available in a beta prototype form, HDF5 language bindings for Python (h5py) have already been extended to support this new feature.

Details of the components introduced above follow.

Mapped Data Views

The large and complex data produced by mapping experiments will be investigated and visualised with a bespoke user interface (a perspective in Eclipse RCP technology). This user interface, an example of which is shown in Figure 2, allows multiple maps, with raw and processed data, from multiple detectors, to be viewed at the same time. Due to the modular structure of RCP the same user interface will be available in DAWN [11] for offline analysis, and in a GDA client, for live data visualisation and experimental configuration.

Scan Parameter Editor

The GDA client will include an easy-to-use GUI which allows a user to draw a region on a map or camera image, set up other experimental parameters as required, and then trigger a new map scan of the defined region. When a region is drawn on a map the region coordinates are added to a Java bean. A simple auto-generated GUI for the bean allows other fields to be edited, and there is a scan button which passes the bean to the acquisition server to trigger a scan. The GUI is auto-generated using the Metawidget framework.

Epics AreaDetector & Zebra

Epics AreaDetector and Zebra write data that is linked to by the scan's NeXus file. Both must be updated to write SWMR compatible HDF5 files to allow live access to their data. Both are also being updated to write data frames into multidimensional datasets in arbitrary orders.

This latter feature will provide support for complex or more efficient scan mechanisms, from bi-directional raster scans to spiral or even random sampling scans. During these scans data must be written in multiple dimensions to match the dimensionality and path of the scan. For example, during a bi-directional mapping scan where each second line is reversed, AreaDetector must write detector data as nodes in a two dimensional dataset in the order that mirrors the path the x-ray beam traverses across the surface.

To achieve this, AreaDetector attaches the global position of each frame in a scan as meta-data immediately after the frame has been acquired from the detector. The file-writer then makes use of this meta-data when storing the frame in the multi-dimensional dataset. At the time of writing, this work is well advanced and has been rolled out to a non-operational beamline for beta-testing.

NeXus File Writing

At the time of writing, the GDA acquisition server includes an early implementation of the new SWMR features built using the Java bindings for the HDF5 library.

Remote Dataset Component

Remote Dataset is a technology available at Diamond which allows data situated on a remote disk to be accessed by a client. It exposes to the remote client something called a Lazy Dataset with which data can be optionally sliced and optionally down sampled then returned to be visualized on the client. It also allows MJPEG streams to be connected and plotted live in the plotting system with functioning visual tools on the remote live data. Remote Dataset works in Eclipse RCP using an OSGi service [12] but it also works with non-Java clients because the data is requested and returned by web sockets.

Live Processing Component

With all the data available through SWMR, it is possible for processing to be performed by any software

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

supporting this new protocol. As standard though the DAWN processing system will be used, as it has been used successfully for post processing in many fields such as diffraction imaging and ARPES analysis, integrates with GDA, and can easily be extended to make full use of SWMR.

Acquisition Server and Inter-process Communication

The acquisition server coordinates the actions of the Malcolm, processing and client-side processes during a scan. Figure 3 shows an early design of how these processes will fit together.

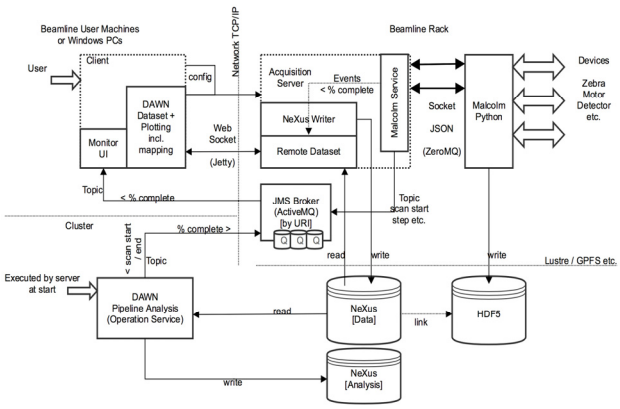


Figure 3: An early design showing the acquisition server at the middle of the system.

CONCLUSION

Diamond has targeted five beamlines with similar but unique mapping requirements. This variety, combined with a coordinated approach from the centralised controls, acquisition and analysis groups should ensure the result is modular, maintainable and expandable. The prototype Malcolm EPICS layer provides a promising API for continuous scanning; and the use of HDF5 SWMR to streamline data flow simplifies many components in the system and has almost removed the line between live and post processing and visualisation. The project is progressing well and it likely that over the coming years its impact will spread wider than its initial target of five beamlines.

AUTHOR CONTRIBUTIONS

RW, MB, AA, PCYC, TC, JF, MG, CM, JM, CP, UP and SG collaboratively developed the software packages involved. RW and MB jointly prepared the text for this article, with contributions from TC, JF, MG, CMP and UP. RW, AWA, AJD and NR directed the development effort and the writing of this article, and PQ represented the beamline staff's requirements and use cases.

REFERENCES

[1] Mosselmans, J. Frederick W., Paul D. Quinn, Andrew J. Dent, Stuart A. Cavill, Sofia Diaz Moreno, Andrew Peach, Peter J. Leicester et al. "I18-the microfocus spectroscopy beamline at the Diamond Light Source." Journal of synchrotron radiation 16, no. 6 (2009), 818-824.

[2] Rau, Christoph, Ulrich Wagner, Zoran Pešić, and Alberto De Fanis. "Coherent imaging at the Diamond beamline I13." physica status solidi (a) 208, no. 11 (2011), 2522-2525.

[3] Könnecke, Mark, Frederick A. Akeroyd, Herbert J. Bernstein, Aaron S. Brewster, Stuart I. Campbell, Björn Clausen, Stephen Cottrell et al. "The NeXus data format." Journal of applied crystallography 48, no. 1 (2015), 301-305.

[4] The HDF Group. Hierarchical Data Format, version 5, 1997-2015. http://www.hdfgroup.org/HDF5/

[5] HDF5 SWMR, https://www.hdfgroup.org/HDF5/docNewFeatures/NewFeaturesSwmrDocs.html

[6] T. Cobb, "Zebra: A flexible solution for controlling scanning experiments", ICALEPCS 2013, San Francisco, USA.

[7] Malcolm http://malcolm.readthedocs.org

[8] Generic Data Acquisition, www.opengda.org

[9] Dalesio, L. R., Kraimer, M.R., Kozubal, A. J., "EPICS Architecture," in Proceedings of International Conference on Accelerator and Large Experimental Physics Control Systems, C. O. Pac, S. Kurokowa and T. Katoh, Eds. (ICALEPCS, KEK, Tsukuba, Japan, 1991), pp. 278-282.

[10] I.S. Uzun, et.al, "PandA Motion Project - A Collaboration Between SOLEIL and Diamond to Upgrade Their Position and Acquisition Processing Platform", Proc. of ICALEPCS'2015, Melbourne, Australia.

[11] Basham, M., Filik, J., Wharmby, M. T., Chang, P. C., El Kassaby, B., Gerring, M., & Ashton, A. W. (2015). "Data Analysis WorkbeNch (DAWN)". Journal of synchrotron radiation, 22(3), 0-0.

[12] M. Gerring, "Open Source Contributions and Using OSGi Bundles at Diamond Light Source", ICALEPCS 2015, Melbourne, Australia.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REAL-TIME DATA REDUCTION INTEGRATED INTO INSTRUMENT CONTROL SOFTWARE

P. Mutti*, F. Cecillon, C. Cocho, A. Elaazzouzi, Y. Le Goc, J. Locatelli, H. Ortiz
Institut Laue-Langevin, Grenoble, France

Abstract

The increasing complexity of the experimental activity and the growing raw dataset collected during the measurements pushed the integration of the data reduction softwares within the instrument control. On-line raw data reduction allows users to take instant decisions based on the physical quantities they are looking for. In such a way, beam time is optimise avoiding oversampling. Moreover, the datasets are more consistent and the reduction procedure, becoming now part of the sequencer workflow, is well documented and can be saved for future use. We will report on the implementation of the on-line data reduction on several instrument at the ILL as well as on the obtained performances.

INTRODUCTION

NOMAD is the instrument control software in use at the Institut Laue-Langevin. It has been designed about 10 years ago as a client/server application. The server is written in C++ to have a direct access to the C driver layer while the main client is written in Java to have a portable and reactive GUI application. A number of other client applications have been developed. Among them we can cite the Nomad Web Spy to refresh a web page that displays monitoring information, the Plot Screenshot Generator to generate offline acquisition images, etc. [1]. In the current NOMAD environment, processes and applications are running in different languages while the communication between them is based on CORBA [2]. Processes are started and stopped on demand and they can crash: - continuous development of a small team compared to the number of instruments - module-based architecture with hundred of classes The crashes are part of the problem and taken into account and may not be considered as development mistakes. It is impossible to perform the tests to ensure a 100% robustness.

The integration of reduction or more generally computation methods at the ILL supposes:

- run heterogeneous code (multiple languages e.g. Matlab, Python, etc.) owned by scientists
- run on other computers than the instrument control PC to avoid interferences
- running on different systems (Linux, Mac OS X, Windows)
- acquisition data and computation results must be exchanged in an effective way

* mutti@ill.eu

One solution is to have a monolithic centralised server that processes every computation request and sends the results asynchronously. This solution requires some extra resources to maintain the services (list of computation methods and their update, load-balancing, etc.). Another solution is to have a "microservices" approach [3]. The computation methods are distributed along the existing control and scientific computers. For that we need a fluid, flexible and easy integration. We need Erlang [4] distributed proces functionalities. In our case, those functionalities need to be integrated in C++ and Java, including multi-process, multi-environment, synchronisation and message queue as well as crash management. To be able to achieve this goal, for the development of NAPPLI we have decided to leave CORBA since it is declining technology [5].

WHAT IS NAPPLI

NAPPLI is a very lightweight application server, very easy to install and usable everywhere. NAPPLI stands for *N applications*, where N can be both interpreted as the first character of NOMAD as well as an unknown number. A NAPPLI server provides services for starting, stopping, synchronising and making distributed applications communicate. We can say that it is an application-oriented middleware. The lifecycle of remote applications can be entirely managed within the application. The server is accompanied with a client API in Java and C++ with a modern asynchronous programming model using the *future* concept [6]. The available communication patterns between the applications are request/response, publisher/subscriber (synchronised or not) and return value at the end of the execution of the application. It is possible to use the application server in a non-intrusive way. Existing applications can be called directly without using the provided API. In this case, the application itself is directly responsible of communication with the outside world. The NAPPLI services are intended to be logic and network fault tolerant. An application can terminate with an exception but the remote caller will be notified with an error, so that it will be able to take the decision to restart or not the application. The network layer also implements features to survive to failures.

Simple Example

Figure 1 describe in a simple example the way NAPPLI can be used.

The application *App1* started by NAPPLI on the computer A, requests the start of *App2* on the computer B (Fig. 1 (A)). Notice that there both the computers A and B are running NAPPLI servers. In Fig. 1 (B) once *App2* is running and

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

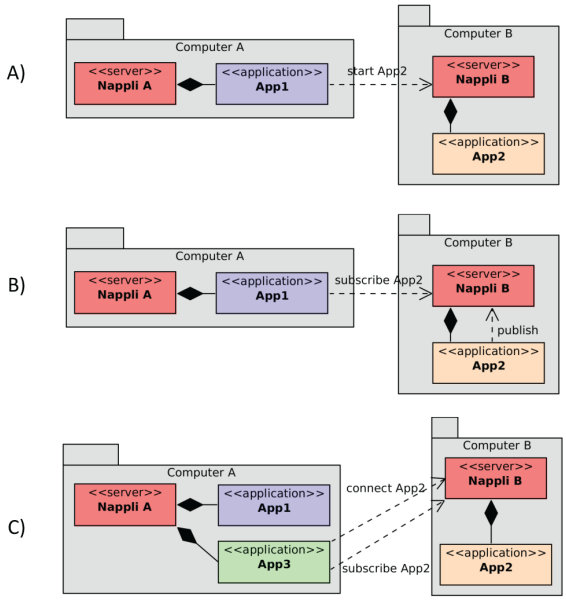


Figure 1: Example of the use of NAPPLI.

declared a publisher, *App1* subscribes to *App2* and it is ready to receive messages published by *App2*. In Fig. 1 (C) the new application *App3*, started on computer A, connects and subscribes to the existing *App2* and it is also ready to receive its published messages. Note that *App2* declares its publisher with a name (so that we can have multiple publishers in the same application) and a number of subscribers. When the number is a strictly positive n integer, the publisher can wait for the n subscribers to be registered before sending any message.

IMPLEMENTATION

NAPPLI was designed taking into account the disadvantages of CORBA. Unlike CORBA which shares data references through a naming service, NAPPLI shares application instances. This is a real different approach. Thus the applications have the responsibility to organise the sharing of their data through persistent services where NAPPLI provides patterns for communication. Internally, a NAPPLI server is written in pure Java 8 so that it only requires a compatible virtual machine for running. That makes it very portable and easy to install. Moreover, NAPPLI application instances are processes started by the server. We take advantage of the continuous Java improvement in its process API to have a unified way to start and monitor processes on different platforms (Linux, Mac OS X, Windows). To organise the network services, we use the robust and reliable ZeroMQ [7] message queue for which a 100% Java implementation called JeroMQ [8] exists. ZeroMQ is not only an open-source library, it also provides a precise documentation on how to use it in different network contexts. For example we followed the recommendations to implement a real synchronised publisher/subscriber pattern. Internally, we use the Protocol Buffers [9] library to serialise and parse messages exchanged by the application instances. Protocol Buffers offers a portable and fast data encoding and decoding. The main feature of a NAPPLI server is to start and stop applications on its own system. For that, a NAPPLI server is configured with a list of runnable applications. Each runnable application has a list of attributes so that an application instance can be seen as an enriched system process. We won't provide all the available attributes here but we can cite:

- *Name*: String identifier used by clients to start an application instance
- *Multiple*: yes or no, no meaning that only a single instance of the application can run
- *Restart*: An application instance is restarted in case of error termination
- *Stream*: The application publishes its output and error streams to the clients
- *Executable*: The path to the executable
- *Args*: The list of arguments that are always passed to the executable

Note that it is really important to make the difference between an application configuration and its instances. Applications have a workflow state shown in Fig. 2.

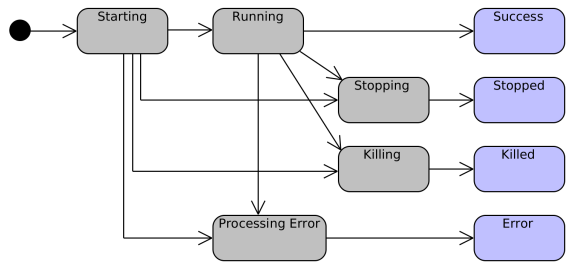


Figure 2: NAPPLI application state workflow.

Around the Running state, there are transitional states (Starting, Stopping, Killing, Processing Error) and terminal states (Success, Stopped, Killed, Error). Once the process of the application is launched, the application has the Starting state. It becomes Running when: it remains alive for a certain time, defined as an attribute of the application, or the application itself changes its state to Running. When a client requests the stop of the application, it immediately becomes Stopping until it terminates. At that moment, its state becomes Stopped. Notice that after the crash of an application (segmentation fault in C++ or exception in Java), its state becomes Error. The changes of state are all sent to the clients of the application.

The messages that are passed between applications can be of any type. NAPPLI provides binary messages as well as string messages so that the programmer can choose the encoding that can be Protocol Buffers or JSON. Arrays of

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

integer and floating point number are also provided by convenience to speed up the coding. Theses messages can be used in the different communication contexts:

- publisher/subscriber
- request/response
- return value

The return value of an application is implemented with a publisher/subscriber so that any connected client application receives the result. We provide a client API for C++ Java and a Python API is planned. A minimal example in C++:

```
// Get a reference to a remote NAPPLI server
Server server('tcp://computer.ill.fr:7000');

// Check its availability
If (!server.isAvailable()) {
  return;
}

// Start App and get a reference to the instance
auto_ptr<application::Instance> app;
app = server.start('App');

// Wait termination of app and get final state
application::State;
state = app->waitFor();

// Get the result from app with a text encoding
string result;
app  -> getResult(result);
cout << 'app returned ' << result << endl;
```

NAPPLI is clearly oriented towards a microservices software architecture rather than a monolithic one, where the various components (applications) are smaller and easy to update and replace. It becomes now obvious how NAPPLI fits perfectly the requirements for implementing data reduction within the instrument control workflow. Every scientific method, resident on remote computers, can, in principle, become a NAPPLI application providing it has the capabilities to accept command-line arguments or file inputs and returns results or file outputs.

COMPUTATION EXAMPLES

ZeroMQ provides different communication patterns that we have implemented in NAPPLI. It enables having different ways of writing the interactions between the instrument control software and the computation applications. We can define three interaction patterns depending on the computation purposes:

- Function application: the remote application is used as a function. The input data are passed to the application arguments and the return results are set by the NAPPLI result functionality. The application terminates after having set the result.
- Asynchronous server: the remote computation application is a server. The input data are passed by a subscriber connected to a publisher located in the control server. Another publisher is located in the application to publish the results asynchronously to the control server.
- Synchronous server: the remote application is a server. The input data are passed by a request of the control server and the results are returned by the response of the application. Even if the response can be asynchronous, we consider the entire procedure as synchronous as we need one response for one request that is not the case for the asynchronous server.

Matlab Q Space Transformation

A number of Matlab scripts were written at the ILL to transform raw detector data into Q space representations. We run the scripts using the Matlab engine library that we can access in C++ by a simple NAPPLI server application called *RemoteMatlab*. The sequence diagram in Fig. 3 illustrates the synchronous server as, in this case, we need to ensure to have one image for each acquisition.

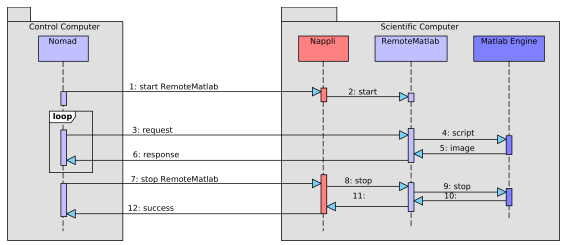


Figure 3: Synchronous server for Matlab Q space transformation.

Notice that there is a single *RemoteMatlab* execution for multiple acquisitions since the Matlab engine is very slow to initialise.

Nuclear Particle Physics Coincidences

When running long data acquisitions using a multi-detector system composed up to several hundred channels, the users need to have a multitude of monitoring information to survey the quality of the data taking. If some of those quantities can be obtained in real-time from the acquisition electronics, some others require the knowledge of the entire detection system and of the physical relation between the different elements. On the other hand, those indirect quantities that are not directly accessible in real-time, must be calculated and visualised within a reasonable time. Moreover, these computations must not disturb the live data taking which can consume lots of resources on the control server. An asynchronous NAPPLI server, as shown in Fig. 4, fits the requirements.

We obtain a two-way streaming of data. Partial acquisition data are sent to the *NPPCoincidence* application which computes and sends the results asynchronously. Notice that we can have a single execution of the computation application per acquisition.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

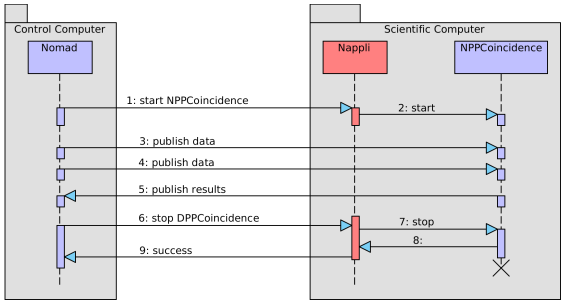


Figure 4: Asynchronous server for computing coincidence rates.

CONCLUSION

We developed successfully NAPPLI to manage and organise the execution of the different applications of the instrument control software environment at the ILL. More specifically it is the ideal solution to distribute and run the scientific computations on other computers without the need for a monolith server infrastructure. The NOMAD software organisation tends to a microservices architecture where different actors have more freedom to make evolve the services they are responsible for. At the ILL, we are at the first step of the integration of the scientific computations in the acquisition workflow. At present computations are used for monitoring the acquisitions but soon they will be used to take automatic decisions. NAPPLI is a generic tool that can be reused in larger environments than the ILL and is not reserved to data acquisition porposes only. Some more developments can be led to ensure its scalability and robustness. But further, the concept of an application-based middleware is proved.

REFERENCES

[1] P. Mutti et al., “Nomad more than a simple sequencer", Proc. ICALEPCS (2011), Grenoble, France.

[2] OMG CORBA, http://www.corba.org

[3] M. Fowler, “Microservices", http://martinfowler.com/articles/microservices.html

[4] J. Armstrong, “Erlang", (2010), http://cacm.acm.org/magazines/2010/9/98014-erlang/

[5] M. Henning, “The Rise and Fall of CORBA", (2006), http://queue.acm.org/detail.cfm?id=1142044

[6] Henry Baker et al.(1977), “The Incremental Garbage Collection of Processes”, Proc. of the Symposium on Artificial Intelligence Programming Languages, ACM Sigplan Notices 12, 8. pp. 55–59.

[7] iMatix ZeroMQ, http://www.zeromq.org

[8] JeroMQ, https://github.com/zeromq/jeromq

[9] Google Protocol Buffers, http://code.google.com/p/protobuf

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ON-THE-FLY SCANS FOR FAST TOMOGRAPHY AT LNLS IMAGING BEAMLINE

G. B. Z. L. Moreno[1], F. P. O'Dowd, H. H. Slepicka, R. Bongers, M. B. Cardoso
CNPEM - LNLS, Caixa Postal 6192, Campinas – 13083-970, Brazil

Abstract

As we go to brighter light sources and time resolved experiments, different approaches for executing faster scans in synchrotrons are an ever-present need. In many light sources, performing scans through a sequence of hardware triggers is the most commonly used method for synchronizing instruments and motors. Thus, in order to provide a sufficiently flexible and robust solution, the X-Ray Imaging Beamline (IMX) at the Brazilian Synchrotron Light Source [1] upgraded its scanning system to a NI PXI chassis interfacing with Galil motion controllers and EPICS environment. It currently executes point-to-point and on-the-fly scans controlled by hardware signals, fully integrated with the beamline control system under EPICS channel access protocol. Some approaches can use CS-Studio screens and automated Python scripts to create a user-friendly interface. All programming languages used in the project are easy to use and to learn, which allows high maintainability for the system delivered. The use of LNLS Hyppie platform [2, 3] also enables software modularity for better compatibility and scalability over different experimental setups and even different beamlines.

INTRODUCTION

The traditional methods for performing scans and controlling experiments at the beamline can present dead-time issues even when comparing the performance of point-to-point scans. Workarounds to the most common causes, like network latency or inefficient programming, may include the control of part of the experiment solely by hardware triggers, orchestrated by low-level implementations. At LNLS, off-the-shelf FPGA-based systems present a standard solution to controlling experiments and integrating equipment at the beamline. EPICS channel access transfers information between low-level device drivers and high-level user interfaces in the form of *EPICS Process Variables*, or EPICS PV's.

The communication between the top and bottom layers of this system architecture takes place over the network in TCP-IP protocol and thus, is non-deterministic. For fast-scan applications, one cannot disregard network latency, inherent to the communication between motion controllers, detectors, and other beamline instruments such as fast shutters and counters triggered from EPICS.

At the IMX Beamline, white beam photon density is of the order of 10^{14} ph/s/mm^2, with an energy range of 4 to 25 KeV. Hence, typical exposure times can vary from 50 to 300 ms for the most common samples. Adding this to the camera's 468 ms readout time at full resolution, frame period could reach more than 500 ms. Due to network overhead while interpreting EPICS PVs, frame period can reach almost a second in a conventional scan. The hardware synchronization is a way to eliminate this latency given the fact that, after its start, the scan runs independently under low-level applications, not relying on communication over the network.

[1]gabriel.moreno@lnls

SYSTEM DESCRIPTION AND DATA ACQUISITION ARCHITECTURE

The LNLS X-Ray Imaging Beamline (IMX) uses a National Instruments PXI-6602 timing board attached to an NI PXI-1045 chassis to read digital counters and trigger devices during scans. A Huber 409 rotation stage performs the sample rotation during tomography. The scan points are programmed directly on Galil DMC-4183 motion controllers as a function of parameters passed from EPICS PV's to reduce latency between scan points and enable quick tomography. The controller sends feedback through the I/O interface on trippoint arrival and transmits the present motion status, enabling hardware synchronization between motors and scan devices. A LabVIEW VI running on a dedicated Windows Machine controls a PCO.2000 camera, which is currently the main detector at IMX Beamline. To prevent high x-ray dose on the sample, a Uniblitz XRS25 shutter blocks the x-ray outside acquisition interval. The user interface screens run from a workstation on CS-Studio environment under Linux Red Hat operational system. CS-Studio inputs integrate to EPICS via Py4Syn scripts to run scan routines and set up experiments.

Simplifying the experiment control system to a three-layered architecture, the EPICS channel access protocol works as an intermediate layer (or "*Service Layer*"). It transfers information between low-level device drivers and the high-level user interface. The graphical user interfaces and experiment command sequences stay on the top layer, or so-called *"Application Layer"*, assigning parameters in the form of EPICS PV's to EPICS IOC's (*Input-Output Controllers*). These IOC's on the "*Service Layer*" translate the parameters received from above into commands and signals to the device drivers on the layer below, the "*Driver Layer"*. To connect the PXI's FPGA environment to EPICS Records, an in-house data exchange protocol called Hyppie [2, 3] creates a bridge between EPICS and the low-level hardware control. The FPGA side runs on LabVIEW Real-Time, as the EPICS side runs on Red Hat Linux. Arriving commands from the application layer land at the Linux side, passing to the RT side over shared memory on the PXI chassis. The hard-

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

ware synchronization is set through 5V TTL signals, exchanged between the scan participants, and centralized at the NI PXI-6602 board. Figure 1 shows the IMX beamline context diagram for experiment control.

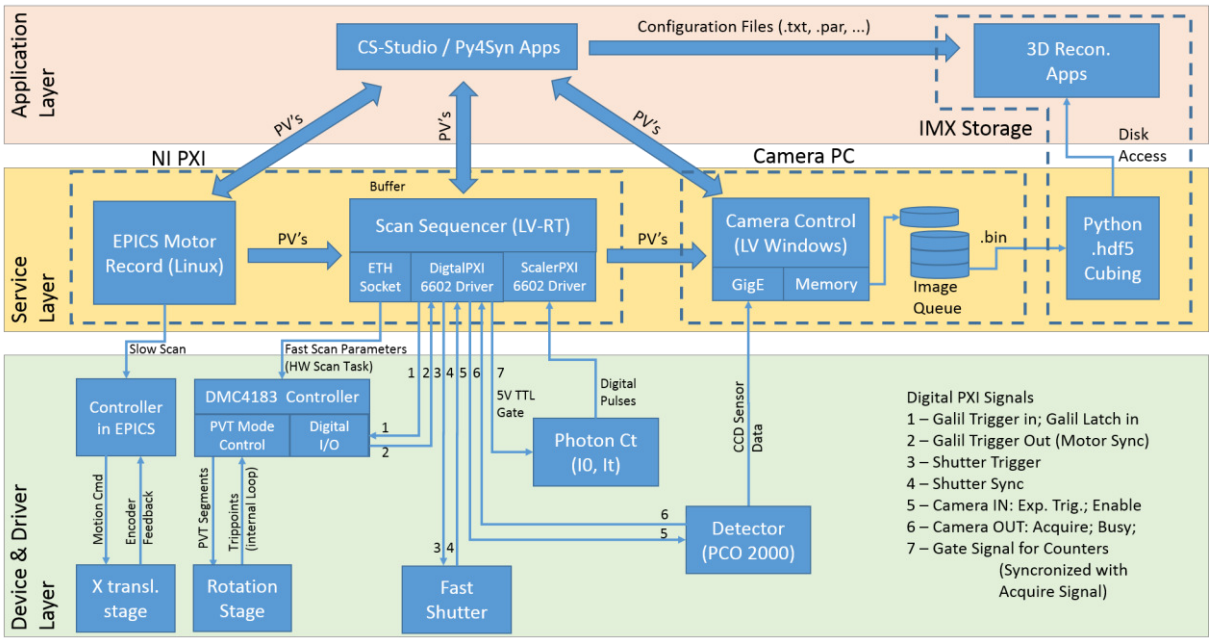


Figure 1: Data acquisition architecture using EPICS Records and LabVIEW drivers.

Scan Sequencer Under Hyppie System

To control the experiment via the NI PXI-6602 board, a new LabVIEW RT implementation developed as a Hyppie module controls the TTL signals triggering the instruments included in the scan task. Hyppie is a project conceived as collaboration between LNLS and NI Brazil to make a bridge between EPICS records and beamline devices connected to PXI-1045 chassis boards. There, the real-time Hypervisor runs both Windows-RT and Red Hat Linux, sharing I/O ports, RAM, and CPU cores. This way, EPICS IOCs, running on Linux, and LabVIEW VIs, on Windows-RT, can share information using a common memory.

The Scan Sequencer is a LabVIEW subVI implemented for handling the scan trigger sequence and send/receive feedback from the user in the form of EPICS PVs. It uses Hyppie's *ScalerPXI* and *DigitalPXI* modules to read and control the PXI-6602 board ports. An embedded EPICS Server in LabVIEW publishes the scan parameters in the form of changeable PVs. In the subVI, state machines run scan sequences oriented for each scan type in IMX, oriented by local variables containing captured PV values. After scan start, only acquired points publication and events expecting PV's for pausing scan are still active.

Synchronization between Hyppie System clock and the Scan Sequencer module proposes better performance scalability. In a performance upgrade opportunity, a reduction on Hyppie's clock period will affect all modules below, including the Scan Sequencer clock. The decision for centralizing trigger signals on the same board instead of connecting equipment in cascade also allows for modularity and scalability. On a device upgrade or exchange, for instance, as long as the new equipment is "triggerable", no reprogramming is necessary to integrate it into the system.

The sequencer is currently responsible for configuration of Galil DMC-4183 motion controllers via ethernet socket, using a VISA resource on LabVIEW. It receives parameters in the form of EPICS PVs and translates them into commands and parameters passed later by the socket.

Galil Controller Implementation

The DMC-4183 controllers, used for most of the IMX Beamline motors, allows the configuration of different motion modes, including Vector Mode, PVT Mode, and ECAM Table mode. The use of these motion modes cover both fly-scans and automated Point-to-Point scans. Sets of parameters passed coming from upper layer software configure the desired trippoints along the continuous or sequential motion path, changing the controller's operation according to each scan purpose. The use of *latch inputs* complements the trajectory programming by enabling position storing on acquisition start. At the IMX Beamline, the acquisition signal acts as a feedback for the DMC-4183 latch inputs, triggering the position storing event. Later, 3D reconstruction software can use the stored positions for error compensation. The execution splits into threads, simultaneously executing the latch position storage and the scan triggering. While the fast scan is not in use, the Galil controller switches back to the common mode of operation, disabling unnecessary threads, and re-enabling threads commonly used.

Camera Control Implementation

A dedicated machine running Windows OS controls the PCO 2000 camera through a GigE interface. The machine has two network boards to handle high data throughput

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

during scans. A dedicated TOE (*TCP/IP Offload Engine*) network board receives data directly from the camera, and a second network card sends data to the data storage location. Once the data reaches the storage, other computers can remotely access it to process or analyze information. In the data route from the camera to the storage, the network configuration in test phase at IMX relies on big package transmission to minimize package loss and network latency. Package size and number of coalescence buffers are high, and all the network switches between the camera and the storage have QoS (*Quality of Service*) priority configuration. The storage is a GPFS file system with the purpose of providing better cost-effective scalability. With such configurations, the data transfer rate reaches 100 MB/s, or around 12fps at full resolution. On the Camera Control application, a queue system is included to account for any additional latency on the network.

In this implementation, a LabVIEW code configures the camera, translating PVs into camera inputs, and writes binary projection data to the storage location. Unnecessary interruptions and EPICS Client PV checks are disabled during tomography and re-enabled afterward. As a way to avoid unnecessary file conversion tasks, external scripts access the binary files from the storage location to convert data to HDF5, TIFF or other image format requested. For the purpose of tomography reconstruction, all images are stacked into a single HDF5 format image. In this way, it is possible to retrieve each projection for the various angles and extract the sinogram directly by changing the stack's cutting plane dimension.

For faster acquisition rates, a 1GB internal storage is available on the camera. However, given the flux and intensity currently available at the LNLS storage ring, the use of binning is necessary for acquisitions with an exposure time of less than 50ms. As binning reduces file size, in some settings it is also possible to store an entire scan on the camera's internal memory, thus removing the readout time incurred during image transfer. As the same applies to scans of regions of interest, the use of such combinations permits higher rate fly-scan testing at IMX Beamline.

CS-Studio / Py4Syn Applications

CS-Studio (Control System Studio), an open source toolset developed by Kay Kasemir of ORNL [5] is the main graphical interface of the beamline. It comprises a set of Eclipse tools that can interact with EPICS PVs. Through CS-Studio, users can view all necessary beamline components, setup scan parameters, and monitor scan progress through a single interface, as well as view the acquired images. For more elaborate routines, such as automated microscope focusing or sample and detector alignment, Python scripts handle and process raw data.

The new Python-based library, Py4Syn, provides a high-level abstraction for device manipulation, scan routines, interactive data fitting and plots [4]. While the scans are complex in nature, there is minimal interaction with the user is minimal, requiring a few parameters at most to run a full experiment. CS-Studio interface calls all python scripts to maintain consistency for the end user

EXPERIMENT SEQUENCE

The flowchart in Fig. 2 shows the process flow in the usual tomography experiment performed at IMX. To perform fast scans, the execution separates into EPICS control and Hardware Control, which runs the scan independently from the motor record. In this occasion, the motor record is used only for updating motor positions. Slow and non-sequential tasks stay controlled via EPICS while the sample rotation runs directly from DMC code. For frame rates higher than 5 Hz, the Uniblitz shutter control can be disabled to stay open during the entire acquisition.

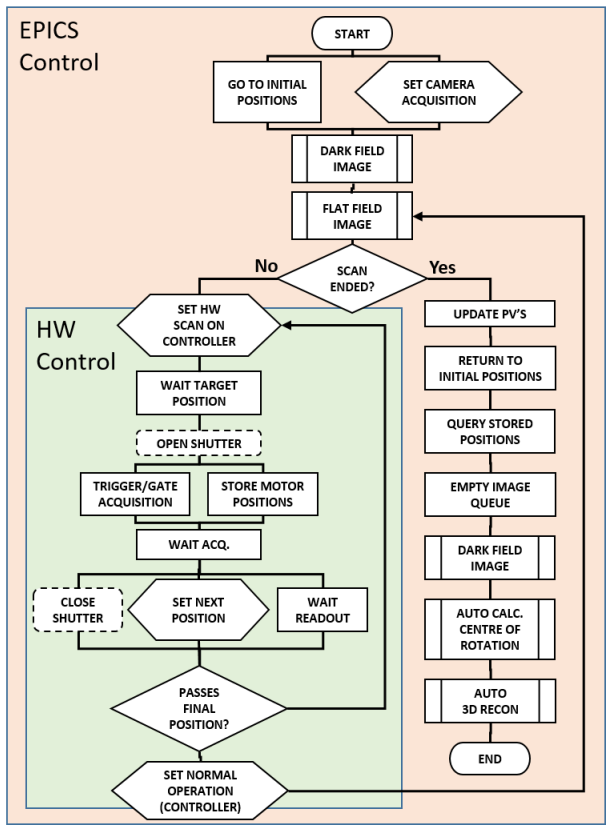


Figure 2: Fast Tomography Experiment Flowchart.

After the execution of the fast parcel, all PVs related to the motor record are updated. As soon as the scan ends, fully automated 3D reconstruction starts, using in-house software [6] or ESRF's PyHST2 [7] software. In-house development automatically calculates sample's center of rotation and the PyHST2 recon parameters are set following current system PV conditions.

EXPERIMENTAL RESULTS

The tests performed at IMX beamline comprised a 1000 projections tomography from 0 to 180 degrees, using 1x8 binning, 1 ADC, and high rate pixel clock on the PCO camera. This experiment configuration brought the acqui-

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

sition and readout times to 10 ms, and 14.2 ms respectively. This way, a frame rate of 20 Hz or about ten times faster than normal acquisition in the same conditions was tested. Table 1 compares the performance of the three scan modes available at IMX Beamline.

Table 1: Trigger Times for IMX Scans

Times (ms)	Normal Scan	HW. Tr. Pt-to-Pt	HW. Tr. Fly-Scan
Exposure time	10 ms	10 ms	10 ms
Read-Out	14.2 ms	14.2 ms	14.2 ms
Motion	193 ms	71.0 ms	46.8 ms
Frame Period	409 ms	88.0 ms	49.2 ms
Latency	206 ms	4-6 ms	2.4 ms
Dead Time	399 ms	78 ms	39.2 ms

The Sample used for this fast tomography test was a common bamboo toothpick, with the reconstructed slices oriented in a cross-section along the direction of the bamboo wood fibers. The effective pixel size was 0,82 µm along the slice, and 6,56 µm vertically along transversal planes. Figure 3 shows the comparison between the slices obtained from the scans, with histogram comparison below.

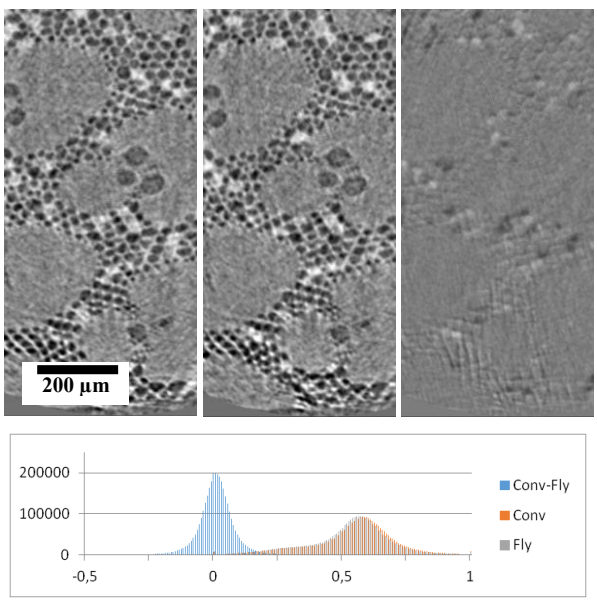


Figure 3: From left to right, normalized slices reconstructed from 8.5 min conventional tomography, 50 seconds Fly-Scan, and error between them.

As shown by the image on the right, the slices from continuous scans have comparable quality with the slices obtained from conventional point-to-point scans. Tests showed a reduction of 99% in the scan latency per point, and 88% decrease in the acquisition time, bringing capability closer to 4D experiments.

CONCLUSION

The tests proved the capability of fast and continuous scan alternatives, with comparable error and frame times at least one order of magnitude shorter than conventional scan methods. Short-term gain for IMX beamline includes fast low-resolution exploratory tomography, reduction of shifts per user, and applications closer to 4D tomography. For long-term applications in Sirius, one cannot disregard data throughput and motion system stabilization. Faster experiment execution will push the beamline infrastructure to its limits, and even sample loading efficiency will have to improve.

Future test phases in IMX involve integrating a new UPR160-Air rotation stage from PI-Micos, which should improve the speed capacity and motion stability for Fly-Scans. Planning includes regular improvements in the network capacity, a robotic sample changer, and even detector upgrades.

The system is modular and scalable, being already implemented and adapted to other beamlines at LNLS [8]. These qualities ensure the extension of this system to other LNLS beamlines shortly.

ACKNOWLEDGMENT

This work acknowledges the entire IMX beamline staff, SIL, GAE and SOL groups from LNLS for sharing ideas and making this work possible.

REFERENCES

[1] O'Dowd, F. et al., "X-ray micro-tomography at the IMX beamline (LNLS)", MEDSI 2014 Proceedings, Australia, 2014.

[2] Piton, J. R. et al., "Hyppie: A hypervisored PXI for physics instrumentation under EPICS", BIW 2012, Newport News, MOPG031.

[3] Piton J. R. et al., "A Status Update on Hyppie: A Hypervisored PXI for Physics Instrumentation under EPICS", ICALEPCS 2013, San Francisco, CA, USA TUPPC036.

[4] Slepicka, H. et al., 2015. "Py4Syn: Python for synchrotrons". *Journal of Synchrotron Radiation,* Volume 22, pp. 1182-1189.

[5] *CS-Studio Guide.* http://csstudio.sourceforge.net/docbook/css_book.pdf

[6] Miqueles, E.X. et al., "High Precision Algorithm for the Center of Rotation in Micro-Tomography", Tech Report, LNLS, 2015.

[7] Mirone, A. et al., 2014. "PyHST2 hybrid distributed code for high-speed tomographic reconstruction with iterative reconstruction and a priori knowledge capabilities". *Nuclear Instruments and Methods in Physics Research Section B,* Volume 324, pp. 41-48.

[8] Perez, C. A., et al., "Development of Fast Scanning X-Ray Fluorescence Microscopy at the LNLS D09B-XRF Beamline", ICXOM23, USA, 2015.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

BEAM PROPERTY MANAGEMENT AT KEK ELECTRON / POSITRON 7-GeV INJECTOR LINAC

K. Furukawa*, N. Iida, T. Kamitani, S. Kazama, T. Miura, F. Miyahara, Y. Ohnishi, M. Satoh, T. Suwada, K. Yokoyama, KEK, 1-1 Oho, Tsukuba, Ibaraki, 305-0801, Japan

Abstract

The electron / positron injector linac at KEK has injected a variety of beams into the electron accelerator complex of an asymmetric collider and light sources for particle physics and photon science experiments for more than 30 years. The beam property of electrons and positrons varies in energy from 2.5 GeV to 7 GeV and in bunch charge from 0.2 nC to 10 nC, and their beam emittance and stability requirements are challenging dependent on the injected storage rings. They have to be switched by pulse-to-pulse modulation at 50 Hz. The emittance control is especially crucial to achieve the goal at SuperKEKB and is under development. The beam energy management becomes more important as it affects all of the beam properties. Beam acceleration provided by 60 high-power microwave stations should be properly arranged considering redundancy and stability. Thus, the equipment controls are also restructured in order to enable the precise control of the beam properties, based on the synchronized event control system and EPICS control system. The strategy and status of the upgrade is discussed from the practical aspects of device controls, online simulation and operation.

INTRODUCTION

At High Energy Accelerator Research Organization (KEK), SuperKEKB, the electron-positron asymmetric collider, is under construction [1]. This project is expected to be able to elucidate the flavor physics of elementary particles with 40-fold improved luminosity compared with the initial KEKB, by doubling the stored beam current, and also by the nano-beam scheme to shrink the beam size down to a twentieth at the interaction point. SuperKEKB is composed of the electron-positron injector, high-energy electron ring (HER), low-energy positron ring (LER), and a positron damping ring (DR).

The Electron-positron injector will perform the first injection to SuperKEKB in Japanese fiscal year 2015. It will fill the dual rings of SuperKEKB as well as two light source rings in top-up mode in 2017 with significantly different beam properties while switching beams at 50 Hz as shown in Fig. 1. The beam energies are 2.5 GeV for Photon Factory (PF), 6.5 GeV for PF Advanced Ring (PF-AR), 4.0 GeV positron for LER and 7.0 GeV electron for HER, respectively [2].

Especially, it is a major challenge to inject beams to SuperKEKB with a small emittance of 20 mm·mrad and an energy-spread of 0.1% under the large beam current of 5 nC per bunch. In order to achieve such a high quality beam, a new RF gun for high-current and low-emittance electron beam and a flux concentrator for high-current positron capture are introduced. It is crucial for the injection operation with higher stability and accuracy to achieve lateral equipment alignment less than 0.3 mm over 600 m linac, and less than 0.1 mm in a short segment. The designed beam should be delivered with iterative corrections understanding the static and dynamic properties of the accelerator equipment and evaluating the beam properties in every conceivable way.

These beam operation management processes need to be improved for SuperKEKB on the basis of the techniques achieved in KEKB.

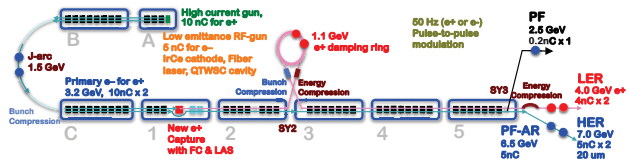


Figure 1: Layout of injector linac and its beam delivery to multiple experimental facilities.

ACCELERATOR EQUIPMENT

The injector linac is operated with the EPICS control framework at the lower and middle layer [3], the event-based control system (MRF) at the lowest layer, and the script languages including SADscript for online accelerator design. This combination was quite successful at the both KEKB injector and collider rings, and is maintained for SuperKEKB as well, with many improvements such as embedded EPICS systems [4,5]. This environment is supported by the carefully-managed control databases.

Example of static database of accelerator equipment for beam-property management is shown in Fig. 2.

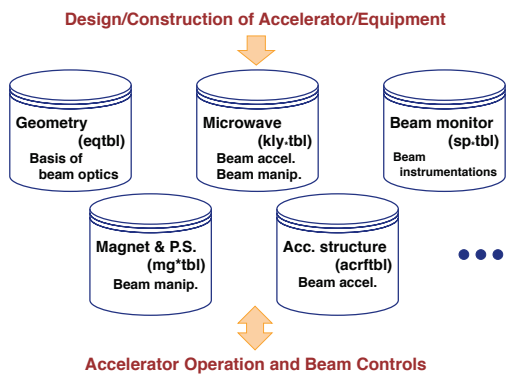


Figure 2: Construction of static database and its contribution to linac beam operation.

* kazuro.furukawa@kek.jp

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Geometrical Arrangement

Each device comes with a name that is up to six characters, the two-charactor device type name, a sector name that represents the installation location (A, B, R, C, 1, 2, 3, 4, 5, 6, etc.), an unit number (1-8, etc.), an accelerating structure or magnet number (0-6), and a sequence number. Each equipment database is a table that has the device name as the key.

As basic information, the equipment database (eqtbl) contains the group name of each device, the distance from the electron gun, the mechanical length, the effective length, and so on. Presently, such geometrical information and the CAD drawings are not automatically reflected each other. A certain mechanism to directly exchange information is being developed.

Accelerator Devices

For the beam transport, the magnet and its power supply information, conversion factors to/from control value, electric current and magnetic field as well as optics correction factors based on beam measurement are maintained in several databases (mgtbl, mgbtbl, mgbftbl). The convention how to define those factors varies between facility to facility, and reasonable one was redefined for the KEKB project, and utilized so far to manage the beam optics.

For beam acceleration, the microwave modulation information (klytbl) and beam acceleration gain information (acrftbl) are prepared and refined using beam measurement, and it is employed to calculate the beam energy and other beam properties along the linac.

There are many other databases for accelerator equipment, beam instrumentations, vacuum system, utilities, and so on. The database for beam instrumentation such as beam position monitors (sp*tbl) has a large number of parameters per a device.

CONFIGURATION OF THE INJECTOR

The injector in SuperKEKB project as in Fig. 1 has eight sectors (A~C, 1~5) of 80-m length each with the exception of the sector A of about 40 m. Each sector has typically eight 10-m long accelerating units, with a high-power RF modulator and four 2-m long accelerating structure (cavity).

Entire 600-m injector is divided into four sections from the viewpoint of beam operation. In each section, one or two units are designed to be redundant for a margin in energy gain. At the end of each section an energy knob is assigned to a pair of units to define the beam energy. Energy knobs and many other units are equipped with fast RF phase shifters, to enable the energy change at 50 Hz through the event control mechanism [6]. Each unit in an energy knob shifts RF phase to the opposite direction to suppress the energy spread as shown in Fig. 3.

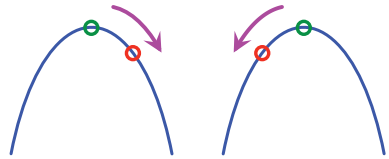


Figure 3: Energy knob to adjust beam energy from crest phases (green) to operation phases (red) compensating energy spread.

BEAM ENERGY MANAGEMENT

It is necessary to manage the beam energy at each injector location by estimating the energy gain at each unit, in order to enable the beam optics calculation.

Adjustment of the Microwave Source

At first, RF conditions for the maximum energy gain at each unit would be stored into EPICS online database. While there is a method to determine the proper RF phase measuring the beam-induced field, the precision is not enough under our circumstances. Therefore, the beam position measurement at large dispersion function is employed to determine the crest RF phase as well as pulse timing for the maximum acceleration.

Based on the above condition, it is necessary to shift the RF phase off the crest in order to suppress the energy spread due to the longitudinal wake field depending on the beam current. Furthermore, the pulse timing should be adjusted at the both shoulders of the RF pulse in order to accelerate two bunches in a pulse 96-ns apart with the same energy.

Determination of the Beam Energy

By using the RF and acceleration database, the corrected energy gain at each unit is calculated and the accumulated energy is stored into the online database. In order to increase the overall reliability, the energy gains are regulated and optimized every week observing the high-voltage discharge rate, the field emission condition, the stability of the power supply, and so on. Otherwise, the related interlocks for equipment protection may disturb the beam injection. This process is important since the beam energy at each location directly affects the accuracy of the beam optics calculation [7].

Energy Stabilization

Depending on the injected ring among HER, LER, PF and PF-AR, the compensating off-crest phases and longitudinal wakefield as well as the beam energies are different because the beam currents are different. Thus, the operating points of energy knobs to secure the energy are also different. Therefore, it turns out that a single injector linac behaves as one of four separate virtual accelerators that are switched one another every 20 ms by the event control mechanism. Actually, operating points of more than 200 devices are changed between those virtual accelerators, and each virtual accelerator must be treated in different manner [5].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

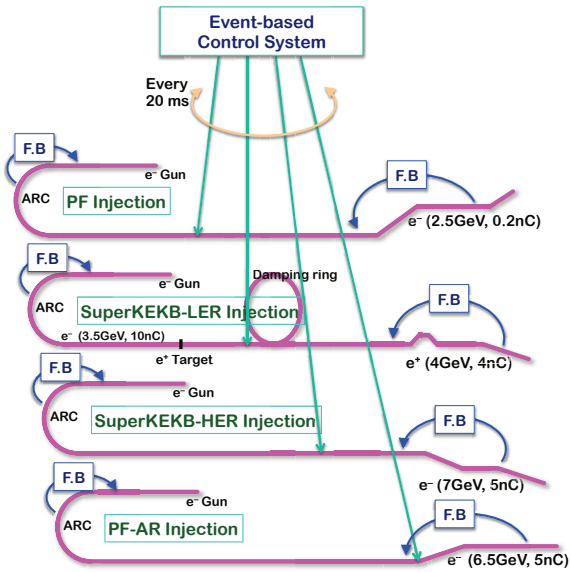


Figure 4: Each virtual accelerator (VA) would be associated with several beam feedback loops, independent of loops in other VAs.

For the end of each section of those virtual accelerators, independent energy stabilizing feedback loops need to be installed as shown in Fig. 4. Similarly, the energy equalizing feedback loops for two bunches within a pulse are required by adjusting the pulse timings. These will be operated with energy knobs and timing adjustments by the beam position measurements at locations of large dispersion functions [8].

BEAM ORBIT AND EMITTANCE MANAGEMENT

During KEKB operation, a large number of orbit stabilization feedback loops were required because of poor air conditioning stability and alignment accuracy, and an attention was paid to maintain the beam orbit not to loose the beam [9].

Towards SuperKEKB, it is necessary to have more accurate orbit controls in order to suppress emittance blow-up. According to a simulation, including the random alignment errors of the accelerating structures and the quad magnets, the beam may have 100-times large emittance easily. The simulation suggests it is possible to deliver a required beam only if their alignment is less than 0.3 mm (rms) for overall 600-m injector and 0.1 mm (rms) for a short segment as shown in Fig. 5. In the method the choice of an orbit can cancel the longitudinal bunch deformation. To that end, the angular accuracy of the steering coil needs to be less than 1 μrad [10, 11]. The beam position read-out system of precision less than 10 μm is also developed to support the method [12, 13].

If it is possible to maintain the orbit, the emittance can be preserved. For example, several orbit stabilization feedback loops may be installed so as to fix the beam angle and position at several locations along the injector. For

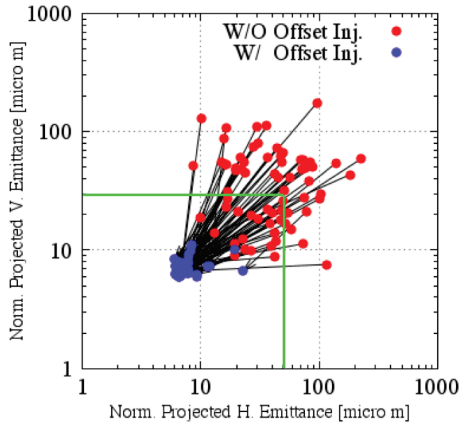


Figure 5: Controlled recoveries (blue) of blew-up emittances (red) with adjusted initial beam positions and angles for 100 random initial simulation orbits.

SuperKEKB multiple feedback loops are necessary with respect to the both HER and LER injection virtual accelerators independently. In order to operate the mechanism flexibly and reliably it is required to refine the quality of the machine database and to maintain the advanced beam optics model.

However, as it has been found in the measurement of long-term floor joint movement that each part of the building move up to a millimeter yearly. More precise and continuous measurements will be performed and absorption or correction mechanism including a mover girder will be installed in order to cure the issue. We need to design the corresponding operational software for emittance preservation.

In addition, not only the transverse beam emittance, but also the beam energy spread or longitudinal beam emittance should be controlled by bunch shape measurement in accordance with integrity of beam generation and bunch compression [14].

DATABASE IMPLEMENTATION

Presently, each control database is configured as a text file of a table in a predetermined format, in which each row corresponds to a device name as a key, and each column represents a device property. Further comments can be written at any row, such as the device history.

Procedures to deal with databases are available in several programming languages. For C/C++ language the database is converted into a hash database in memory, and it can be utilized efficiently [15]. For scripting languages it is converted into a associative array. Also, a set of programs is prepared for the database integrity check and management.

Because a database management system (PostgreSQL) is already used for multiple purposes in the operation, it seems to be natural to manage those accelerator property databases in PostgreSQL, and it is implemented as a test for the next phase of the beam commissioning. At the same time, more and more database elements are put into the EPICS database, and conversion procedures are prepared.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

On performing beam optics calculation, a part of the information is taken from database and other information is directly taken from EPICS online process variables through the channel access protocol. For such purposes, SAD / SAD-script environment for the beam control is quite flexible and is used extensively [16, 17]. Many of such operation programs are designed to refine databases scanning device and beam parameters.

CONCLUSION AND FUTURE

The management of beam properties in the KEK electron positron injector linac was considered based on the controls and databases. Especially for the realization of SuperKEKB, a reliable operation management is required for advanced beam quality and complex virtual accelerators (beam operation modes) with robust databases, calibrations based on beam measurements, iterative refinements of beam optics model in each virtual accelerators, and preservation of fine beam emittance and energy. Based on the experiences in KEKB project the controls and management of beam properties should be further improved during SuperKEKB beam commissioning phases.

REFERENCES

[1] Y. Ohnishi *et al.*, "Accelerator design at SuperKEKB", Prog. Theor. Exp. Phys., **2013**, 2013, 03A011.

[2] K. Furukawa *et al.*, "High-intensity and low-emittance upgrade of 7-GeV injector linac towards SuperKEKB", *Proc. IPAC'13*, Shanghai, China, 2013, pp.1583-1585.

[3] EPICS: http://www.aps.anl.gov/epics/

[4] A. Akiyama *et al.*, "Accelerator control system at KEKB and the linac", Prog. Theor. Exp. Phys., **2013**, 2013, 03A008.

[5] K. Furukawa *et al.*, "Control System Achievement at KEKB and Upgrade Design for SuperKEKB", *Proc. ICALEPCS'11*, Grenoble, France, 2011, pp.17-19.

[6] M. Satoh *et al.*, "Control System Status of SuperKEKB Injector Linac", *Proc. ICALEPCS'15*, Melbourne, Australia, 2015, MOPGF036.

[7] Y. Ohnishi *et al.*, "Design and Performance of Optics for Multi-energy Injector Linac", *Proc. LINAC'08*, Vancouver, Canada, 2008, pp.413-415.

[8] K. Furukawa *et al.*, "Beam feedback system challenges at SuperKEKB injector linac", *Proc. ICALEPCS'13*, San Francisco, USA, 2013, pp.1497-1500.

[9] K. Furukawa *et al.*, "Beam feedback systems and BPM readout system for the two-bunch acceleration at the KEKB linac", *Proc. ICALEPCS'01*, San Jose, USA, 2001, pp.266-268.

[10] L. Zang *et al.*, "KEKB linac wakefield studies of comparing theoretical calculation, simulation and experimental measurement", *Proc. IPAC'11*, San Sebastian, Spain, 2011, pp.739-741.

[11] S. Kazama *et al.*, "Emittance Preservation in SuperKEKB Injector", *Proc. IPAC'15*, Richmond, USA, 2015, pp.239-241.

[12] F. Miyahara *et al.*, "High position resolution BPM readout system with calibration pulse generators for KEK e+/e- linac", *Proc. IBIC'15*, Melbourne, Australia, 2015, TUPB023.

[13] M. Satoh *et al.*, "Development of BPM Readout System Software for SuperKEKB Injector Linac", *Proc. ICALEPCS'15*, Melbourne, Australia, 2015, MOPGF035.

[14] M. Yoshida *et al.*, "Longitudinal manipulation to obtain and keep the low emittance and high charge electron beam for SuperKEKB injector", *Proc. IPAC'13*, Shanghai, China, 2013, pp.1337-1339.

[15] Tb2lib: http://www-linac.kek.jp/cont/libinfo.html#tb2

[16] N. Akasaka *et al.*, "Operation software for commissioning of KEKB linac programmed with SAD", *Proc. APAC'98*, Tsukuba, Japan, 1998, pp.495-497.

[17] SAD: http://acc-physics.kek.jp/SAD/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

COMPONENT DATABASE FOR THE APS UPGRADE*

S. Veseli, N.D. Arnold, D.P. Jarosz, J. Carwardine, G. Decker, N. Schwarz, Argonne National Laboratory, Argonne, IL 60439, USA

Abstract

The Advanced Photon Source Upgrade (APS-U) project will replace the existing APS storage ring with a multi-bend achromat (MBA) lattice to provide extreme transverse coherence and extreme brightness x-rays to its users. As the time to replace the existing storage ring accelerator is of critical concern, an aggressive one-year removal/installation/testing period is being planned. To aid in the management of the thousands of components to be installed in such a short time, the Component Database (CDB) application is being developed with the purpose to identify, document, track, locate, and organize components in a central database. Three major domains are being addressed: Component definitions (which together make up an exhaustive "Component Catalog"), Designs (groupings of components to create subsystems), and Component Instances (“Inventory”). Relationships between the major domains offer additional "system knowledge" to be captured that will be leveraged with future tools and applications. It is imperative to provide sub-system engineers with a functional application early in the machine design cycle. Topics discussed in this paper include the initial design and deployment of CDB, as well as future development plans.

OVERVIEW

The Component Database (CDB) application is a tool for organizing and tracking components and designs used for the APS storage ring upgrade. It helps capture component documentation, provides a repository for inspection and measurement data (e.g., travellers), and supports logging of component history through the component’s life cycle.

CDB also serves as a user portal for finding all known information about a particular component or a design. To that end, it provides links and interfaces to external systems commonly used at APS, such as various drawing and document management systems, procurement applications, etc.

Although CDB has been designed and developed from the ground up in order to satisfy APS-U requirements, in many respects it draws ideas from IRMIS2 [1], which is still in use by Controls Group at APS.

SOFTWARE COMPONENTS

CDB is built around relational database, web portal and REST web service [2] technologies. The architecture, shown in Fig. 1, provides users with a number of options for accessing the system. At the same time, it also offers developers a fair amount of flexibility for integration with external applications. The most important CDB system components are described below in more details.

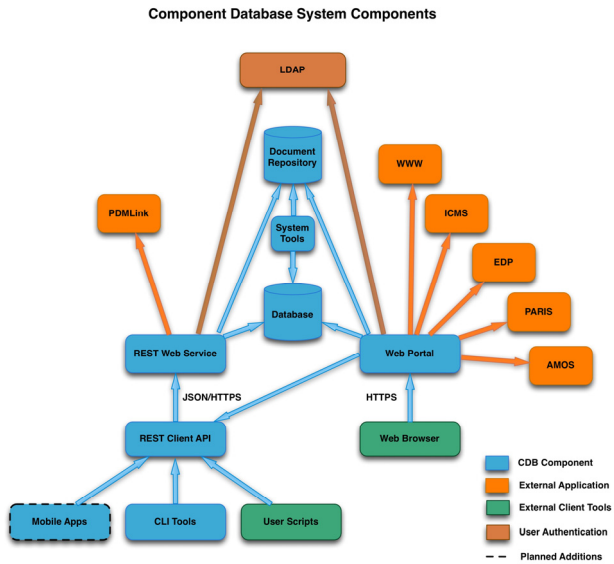


Figure 1: CDB system architecture. Dashed lines indicate future components.

Relational Database

The database contains all system data, other than uploaded documents. Users cannot access the database directly. Except for a small number of administrative tools, other software components access the database through object-relational mapper (ORM) libraries and APIs, which provide an abstraction layer from the rest of the system. CDB currently uses a MySQL database [3].

Document Repository

The Document Repository is a storage area designated for use and managed by the CDB software. It stores various documents, images, and log attachments uploaded by CDB users.

Web Portal

The User Web Portal is a Java EE application running in a GlassFish application server [4], and built using modern technologies like Java Persistence API (JPA) and Java Server Faces (JSF) [5]. In particular, CDB uses the PrimeFaces component suite [6].

Web Service

A REST Web Service and its Client APIs provide programmatic interfaces for accessing the system that are based on JSON data-interchange format [7] over the

* Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under contract DE-AC02-06CH11357.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

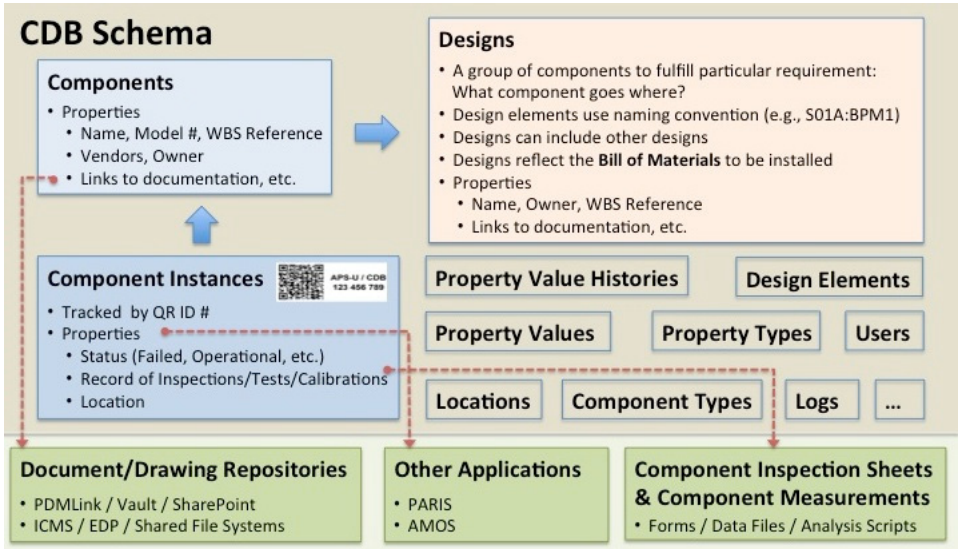


Figure 2: Major CDB domains and their relationship with external sources of information (depicted as rectangles with green background).

HTTPS protocol. The service is implemented in Python using CherryPy web framework [8] and SQLAlchemy ORM [9].

CDB users and administrators can write scripts and higher-level applications on top of the provided Python and Java APIs. It is worth pointing out that the CDB Web Service supports user sessions. This allows users to add new or modify existing CDB objects. Also, note that some of the Web Portal functionality, such as integration with a drawing management system used at APS, also requires accessing the Web Service.

Command Line Interfaces

Command Line Interfaces (CLIs) are built on top of REST Client APIs and expose Web Service interfaces for use within a UNIX shell. All CLI tools have a common set of options including those for command usage, debug level, output display format, etc. In addition, all commands have uniform session and error handling, which simplifies shell scripting.

SYSTEM FUNCTIONALITY

CDB software captures information about component definitions (to form a "Component Catalog"), component instances ("Inventory") and designs, which represent groupings of components used to create subsystems. These three major domains are illustrated in Fig. 2.

Component Catalog

A core CDB purpose is to provide a "Component Catalog" for the MBA. This catalog contains all components planned for use on the new machine, both custom-fabricated and commercially available. For example, each design of a gate valve, magnet, or a vacuum chamber and each unique VME module will have an entry in the Component Catalog.

The minimum metadata required for a component definition is a component name, type (representing generic components used on an accelerator), owner and owner group. Optional metadata may include a description, sources (i.e., vendors), and various other properties, such as images, links to documentation and external systems, etc. The system also supports "complex components" or "assemblies" via a special property that links a component to its associated design.

Inventory

Component entries in the Component Catalog describe a specific component design or a particular model number of a commercially available component. The actual hardware device fabricated or procured is referred to as a "Component Instance" of that component.

Component instances require inspection, testing, storage, installation, and maintenance. Their tracking becomes an inventory management challenge. Each component instance for the MBA will be uniquely identified with a QR code. If possible, a sticker with the QR code will be adhered to the device in a visible location. A database entry will relate the component instance to a particular component definition; thereby allowing all relevant information about this component instance to be referenced using the QR code.

Figure 3 shows an example of the component instance details view in CDB Web Portal. Both the component and component instance properties are displayed, and additional information is easily accessed from links in the property tables.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

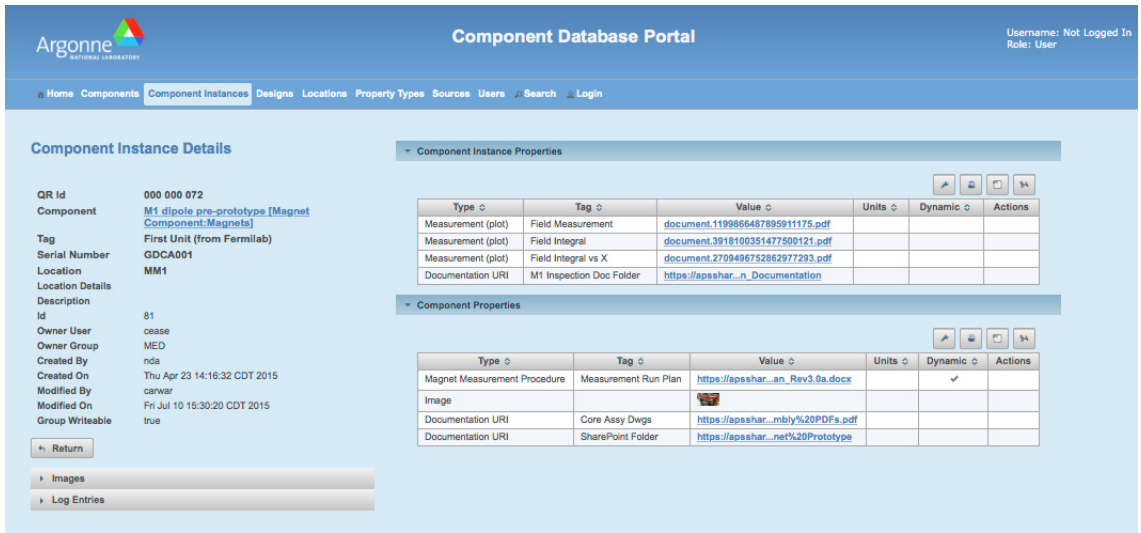


Figure 3: Component instance view in CDB Web Portal.

Designs

The CDB allows a user to define "Designs", which consist of several components grouped together to fulfill a particular functional requirement. One example would be a design for a BPM Processing System consisting of an analog front end chassis, an ADC chassis, four cables for the BPM buttons, and two cables between the other units.

Key concepts of Designs are described below:

- Designs are made up of "design elements".
- Each design element may be a component, a complex component (assembly), or another design (allows hierarchical designs).
- Each "design element" is given a unique element name, normally derived from the official naming convention.

"Designs" are the mechanism by which an exhaustive Bill Of Materials (BOM) can be acquired for the MBA. Groups will define designs necessary to fulfill their particular technical system requirements and by so doing will be contributing to a detailed list of the all the components required to build the new machine. Since this data resides in a relational database it can be "viewed" or analyzed in numerous ways.

Properties

The information one would like to capture in the CDB varies widely depending on the *type* of component or design being defined. Hence, it is impractical to attempt to define a single set of metadata that would be common to all objects. To provide a flexible mechanism for capturing object-dependent information, the CDB associates *properties* with individual components, designs, and instances. This allows each object to have its own unique set of metadata.

For example, a VME Chassis component might have properties of number of slots, height, and AC power requirements. In contrast, a quadrupole magnet component would have properties of maximum current, slot length, weight and maximum field.

Some of the most important features of CDB properties are as follows:

- Property types may be associated with a restrictive set of "allowed values".
- Property types may be linked to a unique "handler" class, which enables different view or edit modes in the Web Portal, or integration with an external system.
- A time-stamped history of each property value is kept to provide a historical log of each property.

Authentication and Authorization

The CDB allows any user to view and retrieve information without logging into the system. However, any changes to the system, such as adding new or editing existing entities, requires users to be registered, authenticated, and (in case of modifying existing objects) authorized to make changes. The authorization model is based on an object's user and group ownership, and the user's group membership. This model has been implemented for both Web Portal and Web Service. This allows, for example, adding new components or loading lattice designs using scripts reading spreadsheets.

DEVELOPMENT PROCESS

The CDB development process is based on lessons learned from developing similar systems in the past. Keys for a successful tool include management support, flexible software design, and an agile development process driven by user requirements and the need to provide solutions for real problems. Hence, our goal is to get new features into users' hands as quickly as possible, and our planning for future software releases heavily relies on user feedback from previous versions.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

FUTURE PLANS

Near term plans include the addition of design instances, adding relationships between design elements (e.g. powered-by, controlled-by, etc.), and ability to capture cable connections.

CONCLUSION

The Component Database (CDB) application has the potential to capture a complete Bill of Materials for the new APS-U accelerator well before the installation timeframe. Having an exhaustive BOM in a relational database will facilitate careful planning and tracking of the construction and installation process, a prerequisite for an ambitious schedule anticipated by the APS-U project.

REFERENCES

[1] D.A. Dohan and N.D. Arnold, "Integrated Relational Modeling of Software, Hardware, and Cable Databases at the APS", p. 365, Proceedings of ICALEPCS 2003, Gyeongju, Korea (2003).
[2] R.T. Fielding and R.N. Taylor, "Principled design of the modern Web architecture", p. 407, Proceedings of ICSE 00, Limerick, Ireland (2000).
[3] MySQL website: https://www.mysql.com
[4] GlassFish website: https://glassfish.java.net
[5] For overview of Java EE features see http://www.oracle.com/technetwork/java/javaee/overview/index.html
[6] PrimeFaces website: http://primefaces.org
[7] JSON website: http://json.org
[8] CherryPy website: http://cherrypy.org
[9] SQLAlchemy website: http://sqlalchemy.org

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

REPLACING THE ENGINE IN YOUR CAR WHILE YOU ARE STILL DRIVING IT*

E.Bjorklund#, LANL, Los Alamos, NM 87545, USA

Abstract

Replacing your accelerator's timing system with a completely different architecture is not something that happens very often. Perhaps even rarer is the requirement that the replacement not interfere with the accelerator's normal operational cycle.

In 2011, The Los Alamos Neutron Science Center (LANSCE) began the purchasing and installation phase of a nine-year rolling upgrade project which will eventually result in the complete replacement of the low-level RF system, the timing system, the industrial I/O system, the beam-synchronized data acquisition system, the fast-protect reporting system, and much of the diagnostic equipment [1]. These projects are mostly independent of each other, with their own installation schedules, priorities, and time-lines. All of them, however, must interface with the timing system.

INTRODUCTION

LANSCE had its beginning in 1972 as an 800 MeV "meson factory" [2]. Since then it has expanded its missions to include such diverse projects as pion treatment for inoperable cancers, spallation neutrons, ultra-cold neutrons, medical isotope production, and proton radiography.

In preparation for the new MaRIE project [3], we are undertaking an ambitious overhaul of a large part of the facility while still trying to maintain a viable user program.

This paper will focus mostly on the timing system replacement project, its conversion from a home-built, centralized, discrete signal distribution system, to a commercial event-driven system from Micro Research Finland [4]. We will explore some of the challenges faced by having to interface with both the old and new equipment until the upgrade is completed.

PROJECT STRATEGY AND SCHEDULE

The installation/operations schedule can be compared to driving through mountainous terrain on a road with many peaks and valleys. When you start down a valley, you shut down your engine, replace as much of it as you can, then try to get it running again before you have to start up the next peak. At the bottom of each valley there is a relatively flat stretch of road representing the "startup period" – during which you mostly coast while you discover how your changes affected the machine's operation (for good or for ill).

The current installation and operation schedule is shown below in Figure 1. The green blocks represent the operating periods, the red blocks represent the installation and maintenance periods, and the yellow blocks represent the startup periods. The durations of the operation, maintenance, and startup periods vary as the project progresses. The first three years of the schedule call for longer operational periods (seven to nine months), shorter upgrade periods (four months), and shorter startup periods (one month). The middle three years – during which the most complex upgrades take place – have longer maintenance periods (four to five months), longer startup periods (three months), and shorter operational periods (three to four months). During the last three years, things theoretically get easier and we go back to longer operations, shorter maintenance, and shorter startup periods.

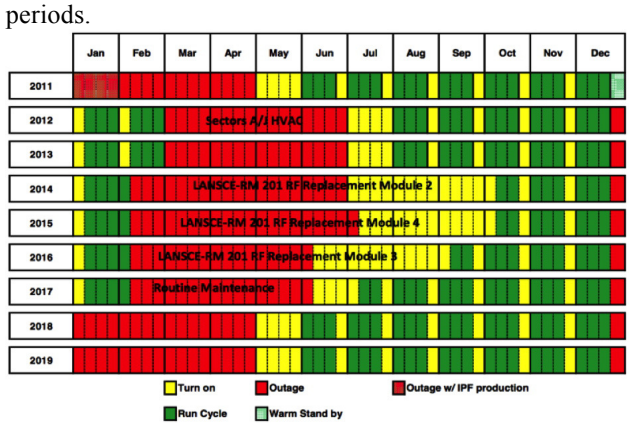


Figure 1: Installation And Operation Schedule

Budget Schedule

Our controls group adopted a budgeting strategy of purchasing all the equipment for each project at once. With some exceptions, for each fiscal year a different project had its own "year of profligate spending". The first year it was the network installation. The second year it was the industrial I/O system. Next was timing, etc. There were a number of reasons for adopting this strategy.

One technical reason was uniformity. All of the equipment for a sub-system would be from the same vendor, with the same firmware level, and therefore have a uniform interface to the controls software.

A scheduling reason was flexibility. Once you have had "your year" your system doesn't have to worry about having enough equipment to meet other projects' sometimes unpredictable schedules. This, of course, implies that those sub-systems most depended on by other sub-systems (for example, network and timing) will need to be financed earlier.

One financial reason was an uncertain funding profile (as described in [1]). Purchasing everything at once

*Work supported by US DOE under contract DE-AC52-06NA25396
bjo@lanl.gov

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

guarantees that at least some part of the accelerator will be improved if funding dries up the next year.

OBSERVATIONS AND RECOMMENDATIONS

As of this writing (September 2015), we are currently coming to the end of the fifth year startup period. This is approximately the mid-point of our upgrade project. We are now far enough along in the project to a) make us think we have at least some idea about what we are doing, and b) provide us with a little hindsight into what worked well for us and what didn't. With this in mind, we offer a few of our general observations and recommendations – with specific examples from the timing system upgrade project.

Observation 1: You Can't Replace The Whole System At Once

In fact, in most cases you can't even replace a whole subsystem at once.

Admittedly, this is a pretty obvious observation. After all, we did schedule nine years for the project! But the implications of this observation can sometimes be less obvious. Upgrade tasks need to be broken up into sub-tasks that are small enough to be accomplished during the scheduled outage periods. Interfaces to other projects need to be considered along with the other projects' installation schedules. Long lead-time equipment needs to be budgeted and acquired in time to meet the installation schedules.

The timing system upgrade installation is complicated by the fact that it interfaces with so many other systems – each of which have their own installation schedules. We were, however, able to come up with the following general plan:

- First, install the event link distribution infrastructure. Here we were able to use the same fiber-optic cables as the network distribution. So that part was easy.
- Install the new timing pattern generator and use it to generate timing signals for a small number of other upgrade project installations. This was probably the most difficult part of the project.
- Interface the entire machine protection system to the new timing system. This may well be the most ambitious part of the project. Once done, however, it will make life easier for machine protection, industrial I/O, and the beam-synchronized data systems.
- Provide timing for other projects based on their installation schedules.

Observation 2: Some Compatibility Must Be Maintained Between The Old And New Systems

This follows from Observation 1. If you can't replace an entire system in one outage, then you will have to run your machine with both the new and old systems working in parallel. For us, this reality prompted the following question:

> *"Can one accelerator be served by two timing masters?"*

In our case, the initial answer was "yes", but the ultimate answer was "no".

Early on in the upgrade project, the only equipment requiring the new timing system was the new wire-scanner system. The new wire scanners required only one timing output signal that could be switched between one of five different beam gates. The Micro Research Finland system has the ability to sample up to eight signals and replay them across the event link. This allowed us to "slave" the new timing system to the old timing system by simply digitizing the desired gates. When we started installing the low-level RF systems, however, we needed to provide more than just eight gates, so slaving through sampling was no longer an option.

We tried running the old and new timing masters in parallel, but in the end we could not keep the AC zero-crossing circuitries synchronized and the jitter between the two systems was unacceptable.

What we finally ended up doing was constructing a big 15-slot VME system with ten 16-gate event receiver modules and programmed it to generate all 96 of the original timing gates (plus a few ancillary gates). We were then able to connect the gates replicated by the new timing system to the old system's distribution network.

Instead of slaving the old timing system to the new timing system, we slaved the old distribution network to the new timing system.

Recommendation 1: Always Have A Way To Fall Back

We have found it prudent to keep the old equipment around for at least a year while we work out the kinks in the new equipment. The first year we installed the new timing system, we ran both systems in parallel – each system producing exactly the same timing gates. As we mentioned above, the jitter between the two systems was unacceptable, so only the new system was connected to the distribution network. However, if for some reason the new system failed, we could easily switch back to the old system by simply relocating four ribbon cables.

Sometimes you may have to fall back even if your equipment is working perfectly. Within a week after installing the new timing system we got a request to put the old system back because of continuous and unexplained machine protection faults. This posed a problem for us because the new low-level RF systems (also installed that year) needed timing features that were only available from the new timing system. Fortunately, we were able to resolve the problem, but if we hadn't we would have been required to roll back to both the old timing system and the old low-level RF system.

Sometimes, the fall back does not have to be to the old system. One useful strategy we have found for the timing system was to have a separate, redundant, set of hardware

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

we could switch over to whenever we needed to do maintenance or a software update on the system.

Observation 3: You Will Be Surprised

One thing you will be surprised at is how long old technology can keep running! It can continue running long after its designers have retired, long after the original implementers have left, and certainly long past the time that any spares are still available. Obviously this equipment is ripe for replacement, if only someone could remember how it works!

We are now the second and third generation of engineers and programmers to work on this accelerator. Frequently, when asked why something is done a particular way, the only answer we can give is "STOLA" – which stands for "**S**acred **T**radition whose **O**rigins are **L**ost in **A**ntiquity". During an upgrade, however, the antiquities resurface and the origins are revealed.

Sometimes we start out "knowing"< how the equipment works only to discover hidden design "features" when we try to replicate its functionality. Even more insidious are the undocumented inter-system dependencies waiting to be uncovered.

After we got the machine protection problem sorted out, we went back to work on the low-level RF. Once we reached the point where we were ready to try sending beam, we suddenly started getting machine protection faults again. What we found was that the "I'm OK" signal the low-level RF system sends to the machine protection system was being derived from a timing signal generated locally by the new timing system. However, the masking gate used by the machine protection system to determine when to look for the "I'm OK" signal was coming from the old distribution system. Even though both these gates originated in the new timing system, the skew between the old distribution system and the locally generated gates was enough to cause the fault.

The lesson learned here was that all the gates going to a given system should come from the same source. In fact, it might be best if all the gates in a given geographical area came from the same source.

Recommendation 2: Sympathy For The Operations Staff

Accelerators are complicated machines and change is hard. Even a change that simplifies operation will initially make operation more difficult simply because it is different.

With the new timing system, we changed from a gate-oriented system to an event-oriented system. There are a lot of things an event system can do better than a gate-oriented system. There are also a lot of things that a gate-oriented system can do better than an event system. The gain of new capabilities from a new system is often eclipsed by the loss of accustomed capabilities from the old system. The difference can sometimes be dramatic. In one instance, the change to the new timing system completely altered the way an entire section of the accelerator behaved because of a change in where the beam chopping occurred! These types of changes can seriously impede a startup period if the operations staff is not kept in the loop.

Below are a few suggestions for keeping the operations staff up to speed:

- Training sessions on what has changed during the last maintenance period.
- Involve operations personnel in design reviews – especially regarding the operator interface.
- Involve operations personnel in the installation of the new systems.

Training sessions are certainly important, and many facilities (ours included) will have a "changes meeting" shortly before a startup period begins. A single "changes meeting" is certainly the most efficient way of communicating what's new, but it can sometimes be quite lengthy which does not contribute to retention.

Involving operations personnel in the design and installation activities can be very productive, both for the operators and the systems engineers. Operators tend to have a more global perspective then the system engineers and can help spot those inter-system dependencies that a system engineer might miss.

CONCLUSION

As mentioned above, we are now around the halfway point in our upgrade project. There will no doubt be many more observations to make and recommendations to share in the coming years. In the meantime, we hope these observations have been useful – especially if you are contemplating a similarly ambitious upgrade project. If you are, however, you might want to check back with us at the 2019 ICALEPCS.

REFERENCES

[1] M. Pieck et al., "LANSCE Control System Upgrade Status and Challenges", MOPGF161, these proceedings, ICALEPCS'15, Melbourne, Australia (2015).

[2] S.C. Schaller, "A LAMPF Controls Retrospective", *Proceedings of the Third International Conference on Accelerator and Large Experimental Physics Control Systems* (Elsevier, 1994), p. 516

[3] M. Pieck et al., "MaRIE – Instrumentation and Control System Design Status and Options", MOPGF162, these proceedings, ICALEPCS'15, Melbourne, Australia (2015).

[4] Micro Research Finland website: http://www.mrf.fi

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE MeerKAT GRAPHICAL USER INTERFACE TECHNOLOGY STACK

M. Alberts*, SKA SA, Cape Town, South Africa
F. Joubert#, SKA SA, Cape Town, South Africa

Abstract

The South African MeerKAT radio telescope, currently being built some 90 km outside the small Northern Cape town of Carnarvon, is a precursor to the Square Kilometre Array (SKA) telescope and will be integrated into the mid-frequency component of SKA Phase 1. Providing the graphical user interface (GUI) for MeerKAT required a reassessment of currently employed technologies with a strong focus on leveraging modern user interface technologies and design techniques. An extensive investigation was performed to evaluate and assess potential GUI technologies and frameworks. The result of this investigative study identified a responsive web application for the frontend and asynchronous web server for the backend. In particular the AngularJS framework used in combination with Material Design principles, Websockets and other popular javascript layout and imaging libraries, such as D3.js, proved an ideal fit for the requirements of the MeerKAT GUI frontend. This paper will provide a summary of the user interface technology investigation and further expound on the whole technology stack adopted to provide a modern user interface with real time capabilities.

INTRODUCTION

MeerKAT is a mid-frequency "pathfinder" radio telescope and precursor to building the world's largest and most sensitive radio telescope, the Square Kilometre Array (SKA). MeerKAT builds upon its own precursor, KAT-7 (Karoo Array Telescope), a seven-dish array currently being used as an engineering and science prototype.

During the preliminary stages of the MeerKAT control and monitoring design it became evident that the current KAT-7 user interface and its underlying technologies have various shortcomings and will not scale well. This prompted an investigation into modern user interface technologies, especially web frameworks with all its accompanied benefits.

This paper starts with a brief description of the design methodology adopted. Next, this paper provides a summary of the user interface technology investigation. As main focus, this paper presents the new MeerKAT GUI architecture with detail on the architecture and technology stack.

METHOD

High level requirements for the MeerKAT user interface were defined by System Engineering and further refined through bi-monthly discussions with the relevant stakeholders, especially the telescope operators and commissioners. During these meetings all requirements were clarified, additional operator requirements were defined, and mock-up displays were drawn to ensure the resulting interface will fulfill all the needs of the end users.

Following an iterative development approach, combined with monthly demonstrations of prototype displays to relevant stakeholders, we were able to obtain valuable feedback that we could include in the development cycle.

TECHNOLOGY INVESTIGATION

After conducting a research exercise into GUI technologies for Responsive Web Design (RWD), various frameworks and libraries were identified. Categorising these GUI technologies according to their main function showed that few were all-in-one solutions for a user interface development platform. Most of them focus on a specific area of user interface design and should be used in conjunction with other libraries to construct a complete frontend development platform solution.

To overcome analysis paralysis we've limited our evaluation to only a select few technology solutions based on industry popularity and peer recommendations, namely AngularJS [1], EmberJS [2] and CS-Studio BOY [3].

With the help of the Control and Monitoring (CAM) team we identified criteria to evaluate the chosen technologies against while building prototypes using each technology as a frontend design platform. The prototypes were based on a typical antenna control and monitoring use case. The evaluation criteria included open source licensing, MVW (Model View Whatever) framework, good documentation, large example base, large active developer community, large scientific user community, pre-built standard widgets, in-house knowledge, user-defined widgets, rapid prototyping toolset, template system, support for testing frameworks, deployment/server side simplicity, easy learning curve, security support (authentication/authorisation), Python support, flexible data-bindings and flexible layouts.

Final scoring of the three prototyped user interface technologies made it clear that web technologies are best suited to our needs, with AngularJS coming out as the favourite.

MEERKAT GUI ARCHITECTURE

Architecture Design Overview

On a high level, the MeerKAT user interface implements a client-server architecture. A frontend component provides the client-side functionality and a

* talberts@ska.ac.za
# fjoubert@ska.ac.za

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

backend component provides the server-side functions, as illustrated in Figure 1.

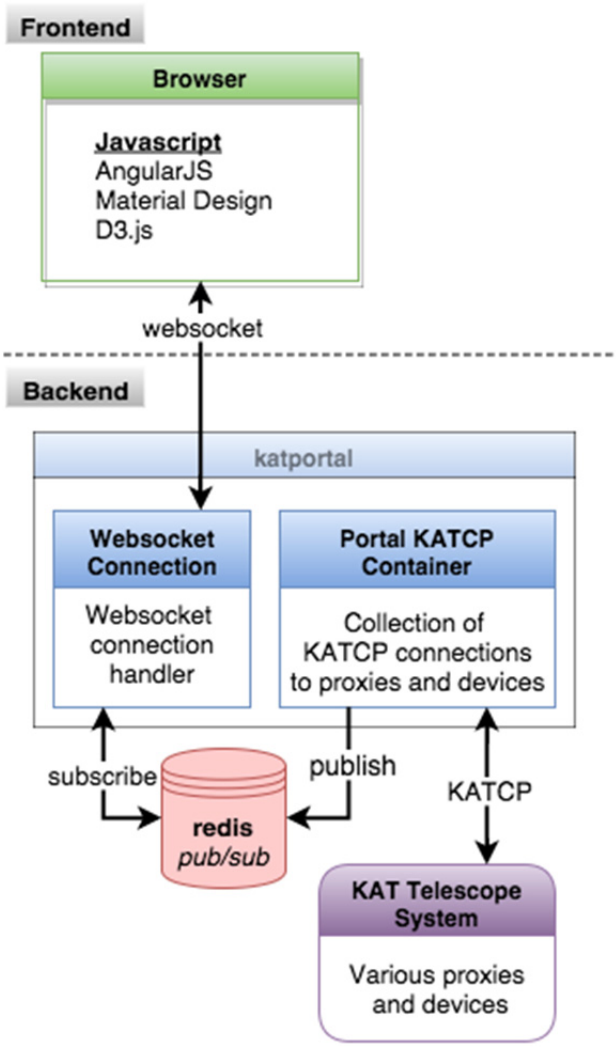


Figure 1: MeerKAT GUI architecture overview.

MeerKAT GUI Backend

Many of the MeerKAT GUI displays require real-time updates of monitored data. Therefore, careful consideration was given to the delivery of a continuous data stream to the frontend.

To enable an interactive session with real-time flow of data between a browser and server, the modern web communication technology of choice is the Websocket [4]. This full-duplex, single socket connection allows a client to send messages to a server and receive event-driven responses without having to poll the server for a reply.

Combining Websockets with a publish-subscribe design pattern (PubSub) allows a GUI display to subscribe to all monitoring points of interest and handle real-time updates to those monitoring points as published by the server.

The Backend Technology Stack comprises:

- Ubuntu 14.04 LTS Server [5] as operating system
- Nginx [6] as Hypertext Transfer Protocol (HTTP) server and reverse proxy
- Redis server [7] for PubSub
- Python 2.7 [8] as the main programming language
- Tornado web framework [9] for the various webservers supporting both normal HTTP requests and Websockets

Katportal is the Python package providing all the backend functionality. It consists of webservers for authentication and authorisation, monitoring, control, querying system configuration and to read data from storage [10].

The monitor webserver exposes a PubSub interface through a Websocket connection. The protocol used for communicating over the Websocket is JSONRPC [11], which enables the server to expose methods as remote procedure calls.

Typically, a client opens a Websocket connection to the monitor webserver and subscribe to a set of monitoring points on the telescope system. The monitor webserver maintains a collection of KATCP (Karoo Array Telescope Control Protocol) connections to the proxies and devices making up the telescope system.

Whenever a subscribe request is received from a client, the monitoring point(s) of interest are registered on the KATCP container and also on Redis. Should the value of any of subscribed monitoring point change on the telescope system, the container will notice the value update event almost immediately and publish the update to Redis. In turn, the existing session for the client's Websocket connection will be notified by Redis and the updated value will be sent to the client.

Default sampling strategies for monitoring points - such as periodic, event-based, event-based-with-rate - are defined through configuration on the server, but can also be set to a custom update strategy by the client through the PubSub interface.

The control webserver exposes a RESTful [12] application programming interface (API) for all control related tasks. Control tasks can only be performed with the proper authentication and authorisation.

The authentication webserver enforces user authentication and authorisation depending on user roles. All web requests influencing the telescope system state (i.e. control related) have to be authorised with a login-unique session identifier as per the Javascript Object Notation (JSON) Web Token standard [13]. If the authorisation information inside the web token matches the information stored at the server for the authenticated user making the request, the action is allowed and executed on the server.

MeerKAT GUI Frontend

The frontend is implemented as a Single-Page Application (SPA), which downloads all necessary HTML - and JavaScript files when the browser loads the frontend's Uniform Resource Identifier (URI). The frontend then communicates with the backend via a RESTful API and Websockets.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

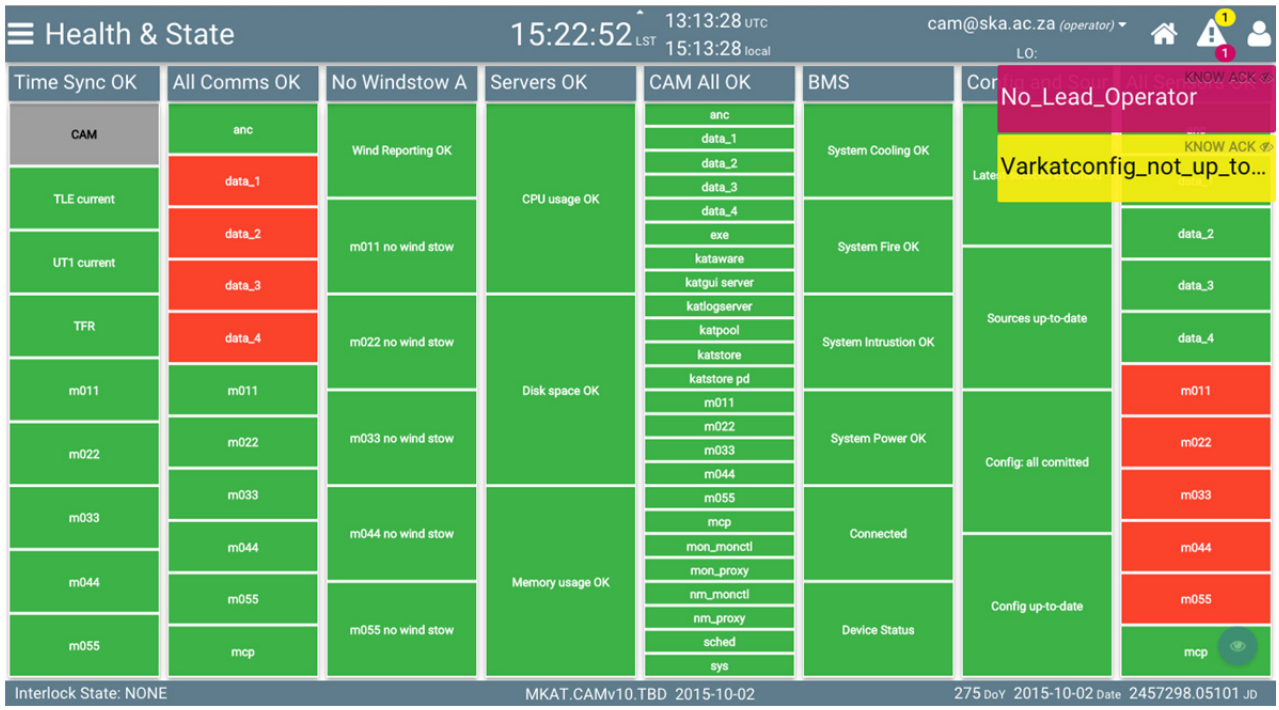


Figure 2: The telescope system health overview display. The system alarms are shown as an overlay at the top right. The main toolbar is displayed at the top.

The Frontend technology stack comprises of:

- The Google Chrome web browser [14]
- AngularJS 1.4.x [1]
- Document Driven Data (D3.js) [15]
- Angular Material [16]
- Numerous JavaScript utility libraries

Each browser window or tab connects to the backend and can make concurrent modifications. Users go through an authentication process that assigns user roles, which in turn limits access and modifications to certain parts of the frontend.

MEERKAT GUI FEATURES

Main Display

The frontend layout contains the following:

- Main toolbar
- Bottom toolbar
- Side navigation
- Alarm notification overlay
- Selected display's content

The main toolbar contains items like navigation links, current UTC, local solar and sidereal time, alarm counter badges, and the logged-in user's information. The bottom toolbar shows the current system interlock status, current version information and date information, including Julian date. The side navigation bar helps with quick navigation between displays. The selected display content occupies the rest of the space.

Landing Page

After a successful login, the user sees the landing page. The landing page is a dashboard that can be configured with widgets depending on the user's needs. Currently, there is a navigation widget and a NASA Astronomy Picture of the Day [17] widget.

Health Displays

Graphical health displays were developed to assist in fast and efficient fault-finding. These can be placed on large, heads-up displays in the operator control rooms. These include the telescope system health overview (see Figure 2), represented as columns of coloured blocks and interactive, customisable, tree-views to show the health of the antennas.

The sensor list display shows all the monitor points of a selected resource as a scrollable list, using colours to emphasise the status of the monitor points.

The user can use regular expressions in the custom health view, in order to filter sensor names and build custom health views, displayed as blocks of colour, and then export these views as a URI for reuse.

Pointing displays show where all the antennas are pointing, in terms of azimuth and elevation as well as right ascension and declination. The pointing displays can be configured to use either the equatorial or the horizontal coordinate system.

A special weather display is used to display the current local weather conditions at the site.

Alarms

The alarm display gives the user the ability to acknowledge and clear alarms. Alarm notifications are shown as counter badges on the main toolbar and as an

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

overlay on each browser tab, which stays visible until the operator acknowledges the alarm. Every time an alarm is received, a different sound is played, depending on the severity.

Observation Scheduling

Observation scheduling displays allow for the organization and scheduling of observations per subarray. Subarrays are logical collections of receptors and other resources. The workflow starts with setting up the subarray by assigning of resources, setting the frequency band, current configuration, end product and a Control Authority for the observations. The subarray is then activated. Only a user who has a role as a Lead Operator or a Control Authority can modify the subarrays.

After the subarray activation, the designated control authority assigns small units of work, known as schedule blocks, to the subarray. The schedule blocks are then verified and scheduled, which moves them into a list of observations that will be executed automatically or manually, depending on the scheduler mode of the subarray in the telescope system.

Operator Control and Intervention

The operator control display allows the user to execute various emergency operations when needed. This display shows the current status of all the antennas, which can be used to monitor the execution of the emergency operations. Operations include: stop observations, resume operations, shutdown computing etc.

Historical Data

The telescope system has robust data storing and archiving capabilities. The historical data can be queried using the frontend, which plots the data on a chart. Multiple data lines of the same type can be drawn on the same chart. Changes in discrete data (e.g. an antennas mode, which switches between 'STOP', 'STOW' and 'POINT') can also be plotted.

User Logs

Users can create different types of logs on the user log and reporting display. These would typically include shift logs, observation logs, time-loss logs, maintenance logs etc. The display allows the user to specify a start and end time for the selected log type, a text log message and functionality to upload files associated with the log. The user can, at any time, generate a report for a log type and include system activity logs in the report for later review.

Configuration and Theming

Many of the features in the frontend can be customised and configured in configuration display. Configuration options include hiding alarm notifications, disabling alarm sounds, and configuring a colour theme for the frontend. The theming in the frontend allows the user to specify the colour of toolbars and buttons as well as selecting a dark background colour for night operations.

CONCLUSION

Employing a framework based on of web technologies allows even a complicated system to reap the various benefits of web based applications. The growth of computing power enables complicated user interface processing to be handed over to thick clients, reducing unnecessary overhead on the backend servers.

PubSub messaging provides an excellent isolation between consumers and producers of data and allows for easy scalability.

ACKNOWLEDGEMENTS

We wish to thank all SKA-SA operators, engineers and commissioners who participated in discussions and provided us with insightful comments and ideas.

REFERENCES

[1] AngularJS website: https://angularjs.org/
[2] Ember website: http://emberjs.com/
[3] CS-Studio BOY website: http://sourceforge.net/projects/cs-studio/
[4] Websocket website: https://www.websocket.org/
[5] Ubuntu website: http://www.ubuntu.com/
[6] Nginx website: http://nginx.org/
[7] Redis website: http://redis.io/
[8] Python website: https://www.python.org/
[9] Tornado web framework website: http://www.tornadoweb.org/en/stable/
[10] M. Slabber, "Overview of the monitoring data archive used on MeerKAT", these proceedings, ICALEPCS, Melbourne, Australia (2015).
[11] JSONRPC standard website: http://www.jsonrpc.org/specification
[12] R.T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", Ph.D dissertation, Dept. Inf. and CompSc. , Univ. California, Irvine (2000).
[13] JSON Web Token (JWT) standard website: http://jwt.io/
[14] Chrome web browser website: https://www.google.com/chrome/
[15] Document Driven Data website: http://d3js.org/
[16] Angular Material website: http://material.angularjs.org/
[17] NASA Astronomy Picture of the Day website: http://apod.nasa.gov/apod/astropix.html

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

EFFORTLESS CREATION OF CONTROL & DATA ACQUISITION GRAPHICAL USER INTERFACES WITH TAURUS

C. Pascual-Izarra#, G. Cuní, C. Falcón-Torres, D. Fernández-Carreiras, Z. Reszela, M. Rosanes,
ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain
T. Coutinho, ESRF, Grenoble, France

Abstract

Creating and supporting Graphical User Interfaces (GUIs) for experiment control and data acquisition has traditionally been a major drain of time and resources for laboratories. GUIs often need to be adapted to new equipment or methods, but typical users lack the technical skills to perform the required modifications, let alone to create new GUIs. Here we present the Taurus [1] framework which allows a non-programmer to create a fully-featured GUI (with forms, plots, synoptics, etc.) from scratch in a few minutes using a "wizard" as well as to customize and expand it by drag-and-dropping elements around at execution time. Moreover, Taurus also gives full control to more advanced users to access, create and customize a GUI programmatically using Python [2]. Taurus is a free, open source, multi-platform pure Python module (it uses PyQt [3] for the GUI). Its support and development are driven by an active and welcoming community participated by several major laboratories and companies which use it for their developments. While Taurus was originally designed within the Sardana [4] suite for the Tango [5] control system, now it can also support other control systems (even simultaneously) via plug-ins.

INTRODUCTION

Taurus is a framework for creating user interfaces (both GUIs and command-line based) to interact with scientific and industrial control systems as well as with other related data sources.

In this work we first give a brief overview of Taurus and then we focus on one of its key assets: the possibility of deploying powerful, customizable and flexible control and data acquisition GUIs within minutes without requiring to write a single line of code. Finally, we discuss the current development efforts and the future plans.

OVERVIEW

Background

Taurus was originally conceived (under early internal names such as Tauico, Tauiwi, and Tau) as the ALBA [6] synchrotron's in-house solution for connecting client side applications to Tango device servers [7]. It provided the user interface code for the Sardana suite [8], which is used in ALBA for control and data acquisition of both the accelerator and all the beamlines.

After its first public release in 2011, Taurus has been adopted by several large laboratories and companies, and it has become very popular among many newcomers to the Tango Collaboration[1].

Since 2012 much effort has been put into bringing control system agnosticity into Taurus: proof-of-concept plugins for supporting EPICS [9] and SPEC [10] were developed and the Taurus core has been refactored in order to isolate the Tango dependencies into an optional plugin for the next major release (Taurus 4)[2].

Community

The Taurus Community largely intersects with the Sardana Community (from which it spun off to reflect the fact that Taurus can be used independently of Sardana) and shares its organizational characteristics and open development model [11, 12]: public code review process, proposal-driven decision taking [13], periodical meetings (~yearly), free licensing (LGPLv3+ [14]), etc.

FAST GUI CREATION

Model-View-Controller Approach

Taurus uses the Model-View-Controller (MVC) pattern [15] to build interfaces[3].

The *taurus.core* module uses plugins (known as *schemes*) to provide *TaurusModel* objects that abstract the interactions with specific sources of data and/or control objects. Some schemes are already implemented for accessing control system libraries (the “tango”, “epics” and “spec” schemes) as well as for data-processing via a Python interpreter (the “*evaluation*” scheme).

Every TaurusModel object can be of type *Authority*, *Device* or *Attribute*, and has a unique name in the form of a Unified Resource Identifier (URI). See Table 1 for some examples of model names. Each scheme implements a

#cpascual@cells.es

[1] While early adopters of Taurus were mostly synchrotrons (*Desy, MAX-IV, Solaris, ESRF,...)* other scientific institutions such as the *ELI-ALPS* and *LULI-APOLLON* large laser installations or the *ONERA* wind tunnel are currently using it. Companies such as *Cosylab*, *Nexeya*, *Tata Consultancy Services* and *Observatory Sciences are* providing services based on Taurus to a growing number of institutions, including the world largest radio telescope (SKA).

[2] Unless explicitly stated otherwise, in the rest of the article the situation described corresponds to the current Taurus4 state.

[3] Many Taurus components combine the View and Controller roles. Those cases could be referred to as “Model-View” instead of MVC.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Table 1: Taurus Model Name Examples

#	Model name (URI)	Model type	Scheme	Represented source of data/control object
1	*tango://foo:1234*	Authority	tango	Tango database listening to port *1234* of host *foo*
2	*tango://foo:1234/a/b/c*	Device	tango	Tango Device *a/b/c* registered on the above database (#1)
3	*tango:a/b/c/state* [(1)]	Attribute	tango	*state* attribute of Device #2 when database #1 is the default [(1)]
4	*tango://foo:1234/a/b/c/d* (or *tango:a/b/c/d* [(1)])	Attribute	tango	Tango Attribute *d* of device #2 (and its implicit DB form [(1)])
5	*epics:XXX:m1.VAL* [(1)]	Attribute	epics [(2)]	EPICS process variable *XXX:m1.VAL*
6	*eval:({tango:a/b/c/d}+{epics:XXX:m1.VAL})*0.5* [(1)]	Attribute	evaluation	Calculated average of the values of #4 and #5
7	*eval:rand(256)* [(1)]	Attribute	evaluation	Random generated array of 256 values
8	*msenv://foo:1234/macroserver/bar/1/ScanDir*	Attribute	msenv [(3)]	ScanDir environment variable from Sardana's msenv scheme
9	*h5file:/mydir/myfile.hdf5* [(1)]	Device	h5file[(3)]	File in HDF5 format saved at */mydir/myfile*
10	*h5file:/mydir/myfile.hdf5:data/energy* [(1)]	Attribute	h5file[(3)]	HDF5 dataset *energy* of group *data* from file #9
11	*ssheet:myfile.ods:Sheet1.A1* [(1)]	Attribute	ssheet[(3)]	Contents of cell A1 of Sheet1 of *myfile.ods* spreadsheet

Notes:
[(1)] Some parts of the model URI (e.g. the authority segment) may be omitted if default values are defined.
[(2)] The epics scheme implementation is still a proof-of-concept. URI syntax may vary in its final implementation.
[(3)] These schemes are only in discussion stage. URI syntax may vary when implemented.

Factory object that takes model names and returns the corresponding TaurusModel objects.

The Taurus view and controller components can be implemented in many forms: command line interfaces such as Sardana's *spock*, web based applications such as those demonstrated in the *taurus.web* module (just a proof of concept for now) or, the most common, PyQt based GUIs.

The *taurus.qt* module provides a set of basic widgets (labels, LEDs, editors, forms, plots, tables, buttons, synoptics,...) that extend related Qt widgets with the capability of attaching to Taurus core models in order to display and/or change their data in pre-defined ways. For example, a *TaurusPlot* widget will display a curve for each attribute model to which it is attached if its value is a one-dimensional numerical array. Similarly, a *TaurusForm* widget will allow the user to interact with the data represented by its attached models (Fig. 1). The actual association of a view (widget) with a model is done by providing the model name to the widget.

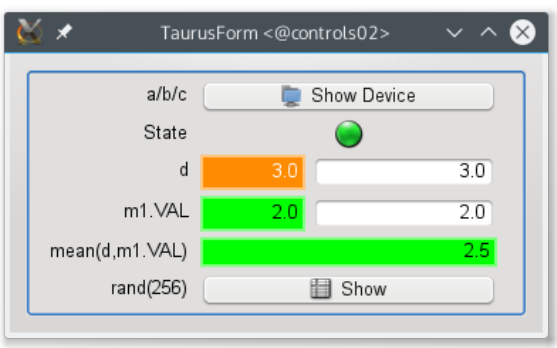


Figure 1: TaurusForm widget attached to models from tango, epics and python evaluation schemes (#2, #3, #4, #5, #6 and #7 from Table 1).

Note that thanks to the model abstraction provided by the scheme plugins, Taurus based applications can transparently mix data from different sources, as demonstrated in Fig. 1.

TaurusGui vs *Qt Designer*

The Taurus widgets behave just as any other Qt widget, and as such, developers could use them to create GUIs in a regular way, both programmatically or using the Qt designer (Taurus extends the Qt designer catalogue with its own widgets).

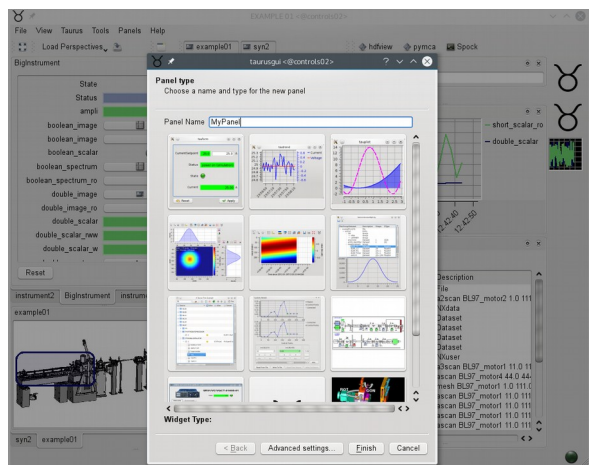


Figure 2: TaurusGUI (in the background) showing the "*New Panel Catalogue*" in the foreground.

However, Taurus provides an even simpler and much more dynamic way of creating GUIs: the *TaurusGUI* framework, which allows the users to create a skeleton of a GUI with a few clicks on a wizard and then populate it at run time with *panels* containing any arbitrary widget from the Taurus catalogue or from an external module (see Fig. 2). The layout of the panels in the application is

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

completely customizable (they can be dragged around, stacked, hidden, made into tabs,...), and different view configurations, called *perspectives*, can be saved and retrieved at any moment to adapt to the task or to the user preferences.

The Taurus widgets can be associated with models when they are added to the GUI and, in many cases, they may also accept drag & drop of the model name(s) from other widgets (or from a model selector) at any time, e.g.: the user can start plotting the time evolution of a given value by just dragging its name from a TaurusForm and "dropping" it into a *TaurusTrend* widget.

TaurusGUIs can be modified (e.g., an extra plot widget may be added on-demand) without coding and users even create whole new temporary TaurusGUIs from scratch to solve particular tasks.

Simplicity for Users, Control for Experts

All the convenience described above does not limit the control that experts may want for tweaking and adapting the system to their needs. This extra control can be exerted at several levels:

First, it is possible to edit the *configuration files* that define a TaurusGUI-based application. These are declarative python and XML files (editable as plain text) complemented by Qt settings files (editable with the provided *taurusconfigbrowser* application).

On a lower level, custom specific widgets (created either programmatically or via the Qt designer) can be added as panels to a TaurusGUI application. At this level, it is also possible to do simple inter-panel communication thanks to the *SharedDataManager* broker component provided by the TaurusGUI framework (see *broker pattern* in [15]). This is used by many Taurus-based GUIs such as VACCA [16] to provide synoptic-based GUI navigation: the visual representation of an instrument in a synoptic can be clicked to show the panel associated with it and, conversely, the synoptic element gets highlighted if its associated panel is selected.

Finally, the maximum level of control can be achieved by programmatically accessing the TaurusGUI class itself. In this way, all the higher level features described before are still available, while there are no limitations on the customizations that can be done.

CURRENT & FUTURE DEVELOPMENTS

Taurus 4.x

At the moment of writing, the latest production-ready release of Taurus is at the 3.6 version, and the next major version (Taurus 4.x) which is currently under development (the core is ready and the widgets are being adapted to it), is planned to be released soon.

Taurus4 implements the TEP3 and TEP14 enhancement proposals [13], simplifying the Application Programming Interface (API) of the core, e.g.: the distinction between an Attribute and its Configuration disappears, many redundant methods are deprecated and the model name syntax is made RFC3986-compliant [17]. This facilitates the creation of new scheme plugins. Also, as part of this refactoring, all the dependencies on PyTango, as well as many tango-influenced APIs have been isolated in the tango scheme plugin, making it possible to run Taurus on a machine without Tango.

Another key improvement from TEP14 is the support for measurement units: all numerical values from the models representing a physical quantity (e.g., the read value, the limits, etc.) will be *Quantity* objects from the *Pint* module [18], enabling user-friendly unit conversions and enforcing dimensional consistency when operating with them.

Taurus4 also brings a significant leap in unit test coverage since many of its features have been implemented using test-driven development.

One of the main concerns when implementing the above mentioned changes was to maintain the backwards-compatibility with the previous version (i.e., that programs developed for Taurus 3.x should ideally work with 4.x). This has been achieved to a large degree by implementing a backwards-compatibility API that translates 3.x API calls to their 4.x equivalents while issuing deprecation warnings). Only in a few cases it has not been possible to implement such an automatic transition (mostly related to tango-centric APIs that cannot be generalized to other schemes), and in these cases a deprecation error is raised. We expect that most GUIs will be able to transition to 4.x without modification (with possibly some non-critical warnings that can be solved at a later moment). The rest may need some minor modifications to run on 4.x, but as long as the basic Taurus widgets are not heavily modified, the update effort should be moderate.

Technology Stack Update

Keeping Taurus up-to-date with the evolution of the technologies is a continuous effort that allows us to guarantee the maintainability of Taurus.

At this moment Taurus is compatible with Python 2.6 and 2.7. We intend to add support to Python 3 too (probably we will drop the support for 2.6 to facilitate maintaining a common codebase for 2.7 and 3.x).

Similarly, we currently support PyQt 4.4 and newer but not PyQt 5.x or PySide since we are limited by PyQt 4.4 to use old-style signals. In the near future, we intend to adopt new-style signals and support all PyQt versions from 4.8 (5.x included) as well as PySide.

Regarding graphics visualization, the *plot* and *extra_guiqwt* Taurus modules[4] depend on the no longer

[4] *taurus.qt.qtgui.plot* provides 2D plots and trends, while *taurus.qt.qtgui.extra_guiqwt* provides contour/colour plots and image visualization as well as alternative implementations of the 2D plots and trends

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

maintained PyQwt 5.2 module [19]. The Taurus Community is currently discussing about the best alternatives for the future of the graphics in Taurus. The options being considered are: a) base all the plotting on *guiqwt* [20], or b) to contribute to the incipient PythonQwt module [21] to bring it to a point where it is a viable drop-in replacement for PyQwt in Taurus, or c) start a new plotting infrastructure based on PyQtGraph [22] or PyMca [23] (which would require more initial effort but would open the possibility to provide OpenGL-based 3D data visualization).

Unification of Extension APIs

Another pressing priority in Taurus is the implementation of a formal API for providing extensions (plugins). At this moment, different mechanisms to support extensions are already implemented in various subsystems of Taurus, such as the schemes in the core, the widgets in *taurus.qt.qtgui*, the panel catalogue in TaurusGUI, the icons, the extension API in the tango factory, etc. (see the TEP13 [13] for more details). Most of these mechanisms, being ad hoc implementations, are quite specific and present limitations such as requiring the plugin to have privileged access to Taurus installation directories, or not having a well defined interface, or not managing dependencies/incompatibilities among plugins.

This situation will be improved by adopting a generic extension mechanism (e.g., stevedore [24] or yapsy [25]) and using it throughout the whole Taurus library. Apart from facilitating the improvement and maintainability of the code (removing multiple different implementations and APIs), the Taurus library will become simpler to extend and lighter, since many sub-packages that are currently monolithic may be reimplemented as a collection of optional extensions to be installed and/or loaded on-demand.

As a side-effect, other projects currently extending Taurus like Sardana, VACCA or PANIC [26] will also benefit from a supported extension API in Taurus: first, by formally registering themselves as Taurus extensions and second, by internally using the same API for their own plugins (e.g., the macros, controllers and recorders in Sardana).

From a community point of view, we expect that the increased modularization resulting from this change will facilitate collaborations since external developers will find it easier to experiment and contribute their changes with less worries of breaking critical components.

Multi-Model API

Another foreseen improvement in Taurus is to extend the existing API for attaching widgets to models. The current API only supports one model per widget, and those widgets requiring to attach to more than one model need to improvise ad hoc solutions. A proposal based on using class decorators to provide multi-model support is being evaluated.

New Schemes

The core refactoring done in Taurus 4 has been the main blocker for the creation of new schemes. Once Taurus 4 is released, we expect that several schemes will be (re)implemented: apart from those mentioned as incomplete in Table 1 (*epics*, *h5file*, *msenv*, *ssheet*), other schemes have been proposed to support the following data sources: the MADOCA-II control system [27]; LIMA devices [28]; archiving systems [29]; SQL databases; plain text files containing tabular data, etc.

CONCLUSION

Our experience shows that the TaurusGUI framework greatly improves the way that control and data acquisition interfaces are deployed: users (even the non-programmer ones) create and/or customize their own GUIs, gaining in autonomy and minimizing both their waiting time and the burden on support engineers.

Taurus is already a reference within the Tango community, but its full potential outside it (both in large and small installations) is still open to be explored and will be enabled by the release of Taurus 4.

ACKNOWLEDGEMENT

We would like to thank the Taurus, Sardana and Tango community members and the ALBA Controls Group and, specially, to Teresa Nuñez (Desy), Jan Kotanski (Desy), Valentin Valls (ESRF) and Sergi Rubio (ALBA) for their contributions to Taurus. We would also like to thank Frederic Picca (Soleil) for his suggestions and for packaging Taurus and Sardana for Debian. Finally, we are indebted to all those who participated in the seminal discussions that gave birth to Tau.

REFERENCES

[1] Taurus website: http://www.taurus-scada.org
[2] Python website: http://www.python.org
[3] PyQt website: http://www.riverbankcomputing.com/software/pyqt/
[4] Sardana website: http://www.sardana-controls.org
[5] Tango website: http://www.tango-controls.org
[6] ALBA website: http://www.albasynchrotron.es
[7] D. Fernandez-Carreiras et al., "ALBA, a Tango Based Control System in Python", THP016, ICALEPCS2009, Kobe, Japan, (2009).
[8] T. Coutinho et al. "Sardana, the Software for Building SCADAs in Scientific Environments", WEAAUST01, ICALEPCS2011, Grenoble, France, (2011).
[9] EPICS website: http://www.aps.anl.gov/epics/
[10] SPEC website: http://www.certif.com/content/spec/
[11] Z. Reszela et al., "Sardana – a Python Based Software Package for Building Scientific SCADA Applications", WCO206, PCaPAC2014, Karlsruhe, Germany, (2014).

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

[12] Z. Reszela et al., “Bringing Quality in the Controls Software Delivery Process”, MOPGF171, ICALEPCS2015, Melbourne, Australia, (2015).
[13] Taurus Enhancement Proposals: http://sf.net/p/tauruslib/wiki/TEP/
[14] Lesser General Public License: https://www.gnu.org/licenses/lgpl.html
[15] F. Buschmann et al., Pattern-oriented software architecture: a system of patterns, (New York: Wiley, 1996), 125.
[16] S. Rubio-Manrique et al., “Unifying All TANGO Control Services in a Customizable Graphical User Interface”, WEPGF148, ICALEPCS2015, Melbourne, Australia, (2015).
[17] Berners-Lee et al., "Uniform Resource Identifiers (URI): Generic Syntax". Internet Engineering Task Force. http://tools.ietf.org/html/rfc3986
[18] Pint website: http://pint.readthedocs.org
[19] PyQwt website: http://pyqwt.sourceforge.net
[20] guiqwt website: https://pythonhosted.org/guiqwt/
[21] PythonQwt website: http://pythonhosted.org/PythonQwt/
[22] PyQtGraph website: http://pyqtgraph.org
[23] V.A. Solé et al., A multiplatform code for the analysis of energy-dispersive X-ray fluorescence spectra, Spectrochim. Acta Part B 62 (2007) 63-68.
[24] stevedore website: http://docs.openstack.org/developer/stevedore/
[25] yapsy website: http://yapsy.sourceforge.net/
[26] S. Rubio-Manrique et al., “PANIC, a suite for visualization, logging and notification of incidents”, FCO206, PCaPAC2014, Karlsruhe, Germany, (2014).
[27] T. Matsumoto et al., “Next-Generation MADOCA for the SPRING-8 Control Framework”, TUCOCB01, ICALEPCS2013, San Francisco, USA, (2013).
[28] A. Homs et al., “LIMA: a Generic Library for High Throughput Image Acquisition”, WEMAU011. ICALEPCS2011, Grenoble, France, (2011).
[29] S. Rubio-Manrique et al, “Validation of a MySQL based archiving system for Alba Synchrotron”, WEP010, ICALEPCS2009, Kobe, Japan, (2009)

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

NATIONAL IGNITION FACILITY (NIF) EXPERIMENT INTERFACE CONSOLIDATION AND SIMPLIFICATION TO SUPPORT OPERATIONAL USER FACILITY

A. Casey, E. Bond, B. Conrad, M. Hutton, P. Reisdorf , S. Reisdorf.
Lawrence Livermore National Laboratory, Livermore, CA 94550 USA

Abstract

The National Ignition Facility (NIF) at the Lawrence Livermore National Laboratory is a 192-beam 1.8 MJ ultraviolet laser system designed to support high-energy-density science. NIF can create extreme states of matter, including temperatures of 100 million degrees and pressures that exceed 100 billion times Earth's atmosphere. At these temperatures and pressures, scientists explore the physics of planetary interiors, supernovae and thermonuclear burn. In the past year, NIF has transitioned to an operational facility and significant focus has been placed on how the users interact with the tools necessary to conduct an experiment at NIF. The current toolset was developed with a view to commissioning the NIF and thus allows flexibility that most users do not require. The goals of this effort include enhancing NIF's external website presence, easier proposal entry for NIF experiments, reducing both the amount and frequency of data the users have to enter, and simplifying user interactions with the tools while reducing the reliance on custom software. This paper will discuss the strategies adopted to meet the goals, highlight some of the user tool improvements that have been implemented and planned future directions for the toolset.

INTRODUCTION

In the past year, the National Ignition Facility (NIF) [1] has transitioned to an operational User Facility and the tools accessed by users needed to be updated accordingly. The current tools were designed to support facility commissioning and have the following characteristics:

- Designed for commissioning a specific function and are integrated with other tools after the fact
- Provide maximum flexibility and options over simplicity
- Tool access is assumed to be from on-site
- Users assumed to have expert level knowledge and experience with the tools

To get an idea of what this looks like to the user, Figure 1 lists the tools that are be used to define an experimental setup on NIF. Even for the expert user, this is a significant number of tools and hence options that the user has to understand. This ultimately detracts from the real task of defining and executing experiments.

Figure 1: Users Need to Interact with Several Experiment Setup Tools at NIF

The goal of the team was to make interacting with NIF analogous to buying an airplane ticket, Figure 2. The user of the airline website needs to only know some high level details of what they want to accomplish such as departure and return dates, number of people in the party etc. They do not need to know about plane weight, fuel load balances, airport operating hours etc. that are needed by the airlines in order to operate safely and get the customer to their chosen destination. In the same way, to conduct an experiment, the user should only need to provide the information relevant to the task that they are trying to complete.

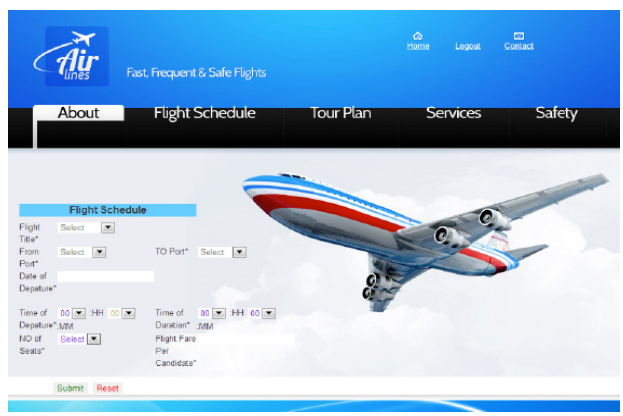


Figure 2: A web site that supports simple user interaction.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

DEFINING THE PROBLEM SCOPE

Solving all of the usability issues in a single update was considered to be too risky and too difficult to implement whilst the tools were being used to support production. So, very early in the process, it was decided that the problem should be broken into three phases.

- Phase one would focus on the start of the experiment lifecycle where the users respond to the request for proposals.
- Phase two would implement a common data framework supporting integration of all of the tools as well as a simplified experiment setup
- Phase three would flow this data through the rest of the processes of experiment planning, experiment definition, facility configuration and ultimately, execution and data analysis.

For soliciting proposals, the stakeholders wanted an external website that could be used without the formality of a login plus additional credentials. It would also allow for collaboration between users; be a source of initial experiment setup that could be used by other tools later in the lifecycle and would have the capability to expand to offer other features such as document configuration, experiment status information and facility metrics to name but a few.

To achieve all of these goals, the team needed to consider a number of different criteria (Figure 3) that could have significantly impacted how they were to proceed. Being on the outside of the lab firewall had significant implications for the IT infrastructure in terms of the cost of procuring the hardware; responsibility for maintenance and patching; as well as monitoring of user events such as administrators logging in, updating of records etc.

The team could have built a custom tool from scratch but there would be worries about security and increasing the amount of code that needed maintenance. Not only that, but the solicitation of proposals is really a customer relationship management (CRM) problem and there were many vendors in the market providing solutions to this kind of process. That said, other tools that the team looked at were either highly integrated into the tool developers own process (as at other national labs for example) or tended to focus more on a sales model that had a look and feel which did not fit with our scientific mission.

Our final consideration was integration with our current tools and the skills that our own developers had. Writing the application ourselves would not present a problem but using a commercial off the shelf (COTS) product might. There would need to be a rich API to access the data and it would need to be compatible with Oracle database technologies that are widely used at NIF.

SELECTING A PARTNER

After much deliberation, the decision was made that the team needed a partner to help create an external website with all of the security requirements that that entails. The team examined Amazon Web Services [2], Google Cloud Platform [3], Microsoft Azure [4], SalesForce [5] and hosting a web page based on Oracle Apex [6] to name but a few. In most cases, infrastructure-as-a-service (IaaS) would require the team to develop all of the software for the application on that infrastructure. Oracle Apex could provide a more complete software solution but the team would need to maintain the hardware and infrastructure externally.

In the end the team decided to go with a platform-as-a-service (PaaS) approach and selected the Salesforce platform. The selection was based on the fact that that CRM platform came closest to meeting all of our critical requirements which are:

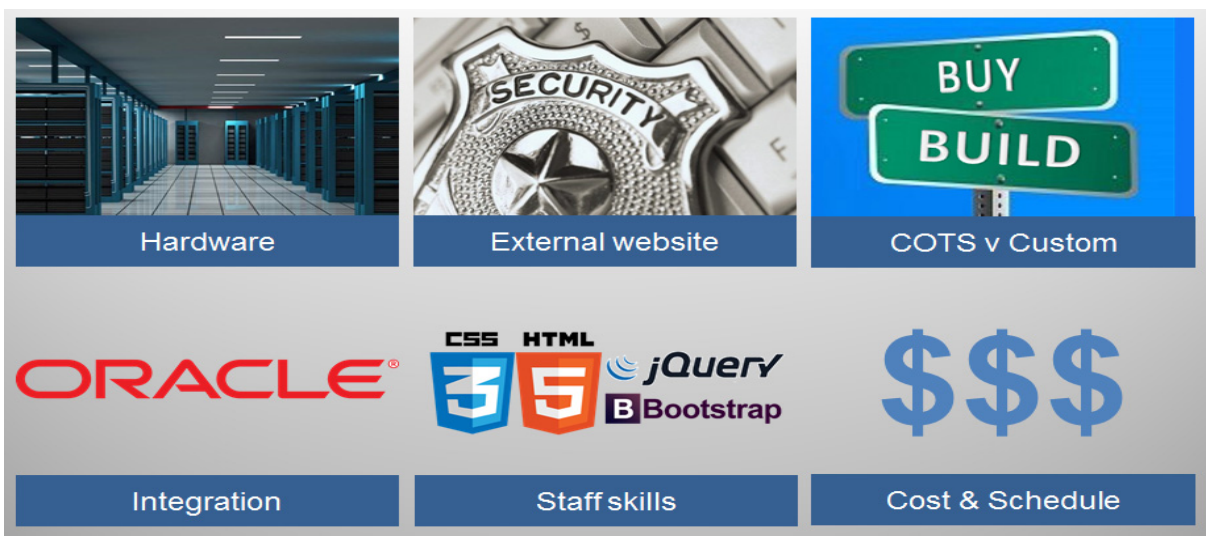


Figure 3: Considerations for selecting a platform for developing an external website.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Figure 4: With all tools there are advantages and disadvantages, SalesForce is no different.

- Hardware maintained by vendor
- Security handled by vendor
- Oracle based like current NIF tools
- Rich API integrates with commonly used technologies; JS, HTML5 etc.
- Minimized development time and effort
- Ability to customize

EXPERIENCE WITH THE SALESFORCE PLATFORM

As this was the first experience with SalesForce, the team decided to use consultants in order to develop the design. The primary goal was to ensure that they followed best practices in terms of SalesForce design and the use of the available components and most importantly, that the team understood the user requirements sufficiently that they could be described to a partner unfamiliar with the problem.

Working with NIF Computer Security was critical in getting the application deployed. Fully understanding their needs and concerns was more difficult than initially anticipated. Despite initiating contact with them well in advance of the projected delivery date, assembling the necessary paperwork and getting an approval to operate (ATO) came close to the delivery date.

Building an application on a platform provided by a vendor always has its strengths and weaknesses and that is also true with the SalesForce platform (Figure 4).

Implementing the proposal tool, the team was required to develop a data model that allowed for users to have different roles depending on the proposal they were working on. They could be a principle investigator (PI) and a collaborator in one call and be a reviewer in another. Fortunately, the platform has very strong role management built into it. Utilizing user profiles and object level security, the team was able to develop the necessary access logic within the framework on the platform.

A proposal tool is essentially a process and workflow engine; requesting proposals, handling the user responses, and then providing notifications to the user of their status. Using triggers and processes, the platform supports automating manual activities such as sending email when a value in an object changes.

Unit testing is a vulnerable part of the development lifecycle as, when deadlines approach, it is often reduced in scope in order to meet the schedule. In order to promote code from a development sandbox into production, SalesForce requires that the module has at least 75% source code coverage. This does not guarantee the quality of the testing being performed, but it does set a minimum expectation from the outset of development that cannot be ignored.

From the beginning of the project, users wanted a way to communicate and collaborate. The SalesForce platform has its own tool Chatter for this purpose. It interacts with email and also provides a history of your communication with an individual or group. The inclusion of this feature meant that the team did not have to integrate an existing tool or write a custom app which was very much in accordance with the desire to minimize the development of new code as much as possible.

The biggest issue the team had with the platform was the native SalesForce development environment, and the lack of a debugger to help diagnose runtime problems. For integrated development environments (IDEs) such as Eclipse [7] and IntelliJ [8] there are plugins available that support the SalesForce platform and they go some way to providing modern development features such as code completion, source code validation, and integrated API documentation. At DreamForce '15 [9], SalesForce announced that their winter '16 platform release would include a debugger.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

The last issue of note is that sometimes the developer is not able to do everything that they want because of the way the framework operates. For example, the developers found that in certain circumstances, they could only get the ID of a proposal object to display when the name of the proposal was desired. There were ways around these types of limitations, but it takes more work and research than might be expected.

THE NIF USER PORTAL & FUTURE WORK

In July 2015, the team launched the NIF User Portal with a single application, the proposal tool on the SalesForce platform (Figure 5), completing phase one of the planned updates. It has been used on a Discovery Science call for proposals and will be used on all future calls. The new tool supports the issuing of the call itself, the handling of the users' proposals, and the reviews of those proposals.

Now that the platform has been established, the team will be focusing on phase two of the upgrade, the creation of an experiment editor and a common data framework. These updates will reduce the amount of data initially required from the user and allow what the user does enter to flow into the rest of the NIF toolset beginning the process of simplifying setup.

CONCLUSION

Having developed an external website for NIF using the SalesForce platform, there are a number of recommendations that the team believes should be followed by any organization thinking of a similar undertaking:

- Get expert help with the design to make sure that the application follows best practices.
- Limit the scope of the initial deployment to reduce risk and provide for a greater chance of success.
- Get computer security involved with the project as early as possible.
- Limit the amount of customization to reduce the amount of ongoing maintenance

ACKNOWLEDGEMENT

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344

REFERENCES

[1] E. I. Moses, "Overview of the National Ignition Facility", Fusion Science and Technology 54 (2008).
[2] AWS website, https://aws.amazon.com
[3] Google Cloud, https://cloud.google.com
[4] MS Azure website, https://azure.microsoft.com
[5] SalesForce website, http://www.salesforce.com
[6] Oracle Apex website, https://apex.oracle.com
[7] Eclipse website, https://eclipse.org
[8] IntelliJ website, https://www.jetbrains.com/idea
[9] DreamForce'15 website, http://www.salesforce.com/dreamforce/DF15

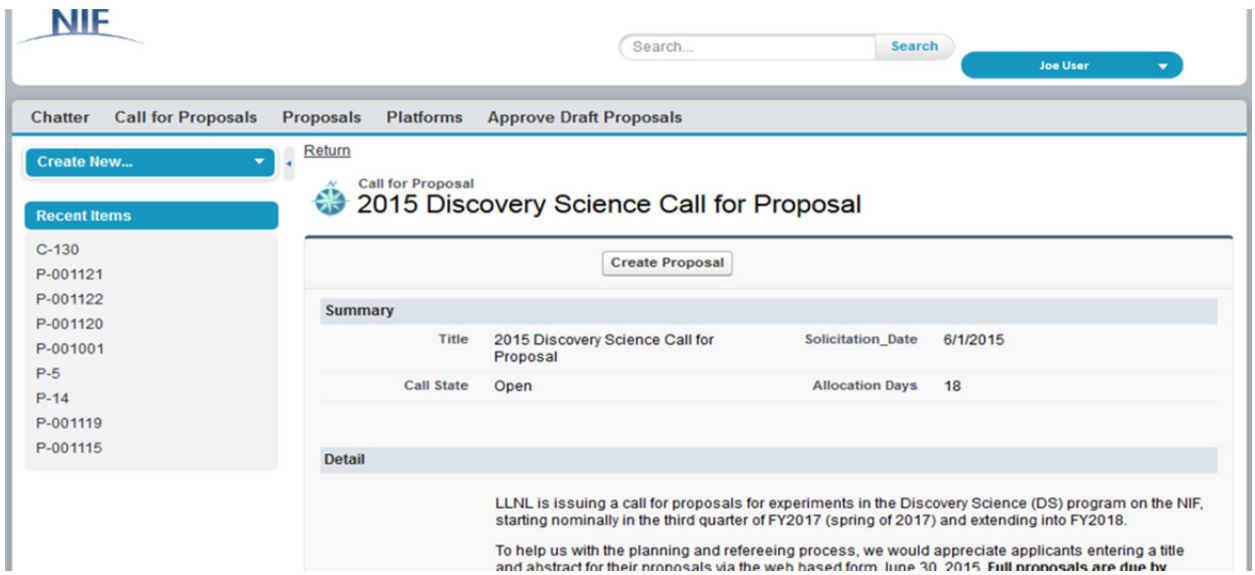


Figure 5: The NIF User Portal proposal management tool went live in July 2015. http://nifportal.force.com/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

CONTROL SYSTEMS FOR SPALLATION TARGET IN CHINA INITIATIVE ACCELERATOR DRIVEN SYSTEM*

Zhiyong He*, Qiang Zhao,Wenjuan Cui,Yuxi Luo, Ting Xie, Xueying Zhang, Lei Yang, Hushan Xu
Institute of Modern Physics, Chinese Academy of Sciences, Lanzhou, China

Abstract

In this paper, we report the design of the control system for the spallation target in China initiative accelerator driven sub-critical (ADS) system, where a heavy-metal target located vertically at the centre of a sub-critical reactor core is bombarded vertically by the high-energy protons from an accelerator. The main functions of the control system for the target are to monitor and control thermal hydraulic, neutron flux, and accelerator-target interface. The first function is to control the components in the primary and secondary loops, such as pumps, heat exchangers, valves, sensors, etc. For the commissioning measurements of the accelerator, the second function is to monitor the neutrons from the spallation target. The three-layer architecture has been used in the control system. In the middle network layer, in order to increase the network reliability, the redundant Ethernet based on Ethernet ring protection protocol has been considered. In the bottom equipment layer, the equipment controls for the above-mentioned functions have been designed. Finally, because the main objective of the target is to integrate the accelerator and the reactor into one system, the integration of accelerator's control system and the reactor's instrumentation and controls into the target's control system has been mentioned.

INTRODUCTION

Driven by the national demand for safe disposal of nuclear waste as well as the potentials for advanced power generation, the Chinese Academy of Sciences initiated an accelerator driven sub-critical (ADS) program in 2011 under the frame of "Strategic Priority Research Program". The ultimate goal of the China ADS program is to build an industrial demo facility for ADS technology. In the China ADS system, a heavy metal spallation target located at the centre of a sub-critical core is bombarded by the high-energy protons from an accelerator. An overall control system is required to exactly couple the high-energy beam from the accelerator to the spallation target in the reactor core, by controlling and coordinating three facilities, i.e. an accelerator, a spallation target and a reactor.

In this paper, we report the design of control system for the spallation target. When designing this control system, the following two objectives have to meet. The first objective is to control the demo facility of spallation target during the commissioning of the target without proton beam. The commissioning without beam may be performed either within the reactor core or outside the core. The second objective is to control the spallation target as a part of the China ADS system, when three facilities, the accelerator, the spallation target and the reactor, have been integrated into one system. Therefore, the same architecture as the overall control system should be used in the control system for the spallation target.

* corresponding author, Email: zyhe@impcas.ac.cn
This work is supported by Strategic Priority Research Program of Chinese Academy of Sciences (XDA03010000 and XDA03030000).

ARCHITECTURE OF CONTROL SYSTEM

The spallation target used in the China ADS system includes several sub-systems. According to the locations of the devices, the subsystems can be classified as the target core subsystem, the primary cooling loop, the secondary cooling loop, the target window and its cooling loop, fill and drain subsystem, cover gas subsystem, and other auxiliary subsystems. The control system for the spallation target controls and coordinates these subsystems during startup, ascent to power, power operation, and shutdown conditions of China ADS system. The control system provides automatic as well as manual controls, monitoring and diagnostics capabilities for these subsystems. Furthermore, since the spallation target is a critical part of the ADS system, its control system should be able to be integrated into the overall control system for the China ADS system.

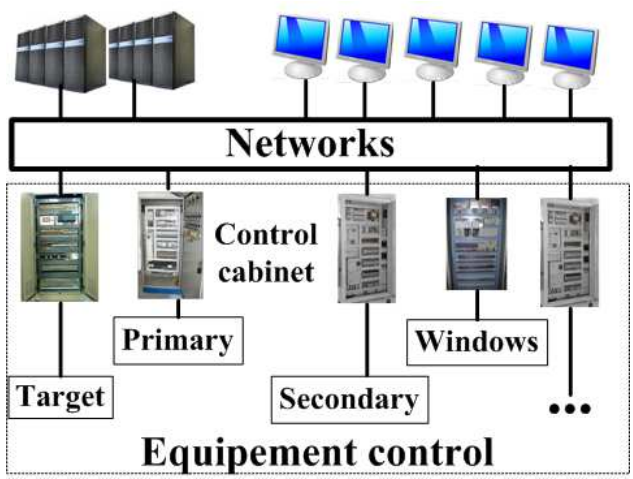


Figure 1: The three-layer architecture of the control system for the spallation target. The legends target, primary, secondary, and window represent the devices for these subsystems.

As shown in Fig. 1, the three-layer architecture is used in the control system, i.e. the top operation layer for the

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

human machine interface, the middle network layer for the data communication and the bottom equipment layer for the devices in each subsystem. In the bottom layer, the legends target, primary, secondary, and window represent the devices for the target core subsystem, the primary cooling loop, the secondary cooling loop, the target window and its cooling loop, respectively. The control system provides control support for all phases of each subsystem's life-cycle, including commissioning and operation.

It is noteworthy to mention that there are an overall control system and several local control systems in the China ADS system. The main functions of these local control systems are to control the auxiliary subsystems for the accelerators, the target and the reactor. For example, the following operations can be controlled in the local control system, such as the maintenance of the target, the handling and storage of the coolants in each cooling loop, and so on.

NETWORKS IN THE MIDDLE LAYER

A communication network is the backbone of the networked control systems. Six networks have been suggested to use in the overall control systems for the China ADS facility [1]. Among these six networks, three networks are required in the control systems for the spallation target, i.e. a central operation network, a personnel protection network, and a data archiving network. In the central operation network and personnel protection network, network reliability is the critical issue while choosing the communication type. The redundant Ethernet based on Ethernet ring protection (ERP) protocol has been considered for the central operation network [1].

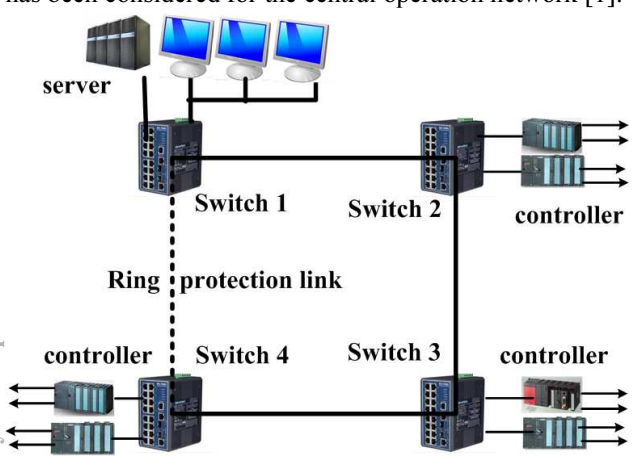


Figure 2: Redundant Ethernet used in the control network for the spallation target. The four switches are equipped with Ethernet ring protection protocol, with which users can establish a redundant Ethernet network with ultra high-speed recovery time.

The ERP protocol, defined in ITU-T G.8032 [2], builds a logical ring topology while maintaining a loop-free forwarding mechanism by logically blocking a link port in the ring, referred to as ring protection link (RPL). Figure 2 shows the redundant Ethernet based on Ethernet ring protection protocol for the spallation target. Although multiple rings may be used in the central operation network for the China ADS facility, only one ring is considered for the spallation target.

The four industrial switches in Figure 2, named as switch EKI-7657C, come from Advantech company (http://www.advantech.com/). The EKI-7657C supports seven Fast Ethernet ports and three Gigabit combo ports with 2x Digital Input and Digital Output ports. To create reliability in the network, the EKI-7657C is equipped with a proprietary redundant network protocol, which provides users with an easy way to establish a redundant Ethernet network with ultra high-speed recovery time.

Based on the report in the user manual of switch EKI-7657C, the redundant Ethernet in Figure 2 can recover from network connection failure within 10ms or less and make the network system more reliable. The algorithm in the redundant Ethernet is similar to spanning tree protocol (STP) and Rapid STP (RSTP) algorithm but its recovery time is less than STP/RSTP algorithm. The ring protection protocol maintains a loop-free forwarding mechanism by logically blocking a link port in the ring, referred to as ring protection link (RPL) in the ERP protocol or X-ring backup path in the user manual of switch EKI-7657C. As shown in Fig. 2, the leftmost link from switch 1 to switch 4 is blocked as a ring protection link. Once a link fails, the vertices adjacent to the failure block the failed link, and the backup path is unblocked. For example, if the link from switch 2 to switch 3 fails, the switches 2 and 3 block the failed link, and the ring protection link from switch 1 to switch 4 is unblocked. Then, the protocol recovers the failed link from switch 2 to switch 3 within 10 ms or less.

EQUIPMENT CONTROL IN THE BOTTOM LAYER

The main function of the equipment layer is to control the devices in the above-mentioned subsystems. In this section, we discuss two main parts of I&C (the instrumentation and controls) used in the equipment layer, i.e. I&C for monitoring the spallation neutrons from the target and I&C for the cooling loops.

I&C for Neutron Monitoring

I&C for the target core subsystem provide both neutron and temperature monitoring for the target. In the China ADS system, a solid tungsten target based on a granular flow method is used. The temperature monitoring for the target core can be done by attaching some thermocouples on the target container. The temperature monitoring method with a thermocouple will be discussed in the next subsection.

The monitoring for the neutron flux level of the spallation target can be done by detecting leakage neutrons from the target core. Since the target is located vertically at the centre of the reactor core, the neutron

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

detectors (e.g. fission chambers) should be put within the reactor core. As discussed in [3], the spallation neutron flux should be measured at multiple vertical locations, because the neutron flux at the central location is more than two orders of magnitude higher than the flux at the lower locations. The multi-point measurement of the spallation neutrons can be performed either by several fixed detectors at the fixed locations or by a movable detector.

When using a movable detection system, the drive system for the insertion and withdrawal of the movable detectors is similar to the incore neutron monitoring system used in the pressurized water reactor. It consists of drive units, limit switch assemblies, rotary transfer devices, and isolation valves. Each neutron detector is attached to a flexible drive cable in the drive units that can be driven into selected core locations by the operating staff.

I&C for Cooling Loops

I&C for three cooling loops, the primary cooling loop, the secondary cooling loop and the cooling loop for the target window, are similar to the I&C in the cooling loops for a reactor. Currently, the analogue I&C systems are used in the reactors of most nuclear power plants. In order to successfully integrate three systems, an accelerator, a target and a reactor, into an entity, digital I&C technology has to be used in the equipment layer for the target. In this subsection, as an example, we study digital I&C technology for the temperature control of several cooling loops.

A temperature control system comprises one or multiple temperature sensors and one or multiple temperature controllers. Two types of temperature sensors, the thermocouple and the resistance temperature detectors (RTD), are used commonly in the temperature control system. The temperature controller applies a dynamic analysis to a plurality of measured temperature parameters, and then, to generate a set of control values by considering the temperature ranges. For different cooling loops, e.g. the primary or secondary cooling loop, the models for the dynamic analysis are different. Finally, the switch-on or switch-off of the valves, the flow speed of the coolants, and the operation of the pumps in the cooling loops are driven by the temperature controllers.

Figure 3 shows the three-layer architecture of a temperature control system. In the middle network layer, several switches EKI-7657C have been used. In the bottom equipment layer, the following products from National Instruments (NI) company are used. PXIe-4353 32-channel thermocouple input module and TB-4353 front mounting and isothermal terminal block are used for the temperature measurement with a thermocouple, while PXIe-4357 RTD input module and TB-4357 terminal block accessory are used for RTD. For the measurement of flow speed and the control of valves, a multifunction data acquisition (DAQ) board PXI-6238 is used. The PXI-6238 can be used to read from encoders, flow meters, and proximity sensors as well as control valves, pumps, and relays; and connect up to other common 20 mA sensors. The green device in Figure 3 is a screw terminal connector for the PXI-6238.

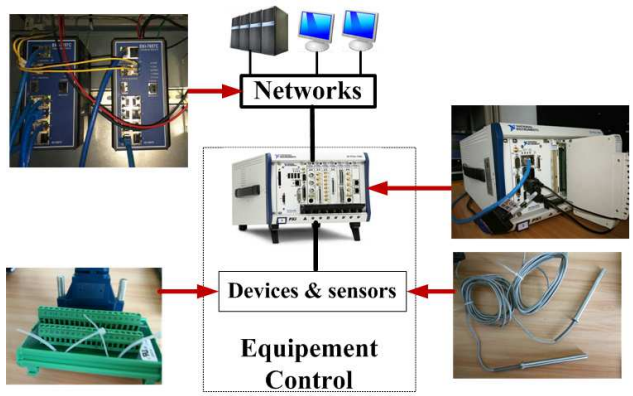


Figure 3: The three-layer architecture of a temperature control system used in the cooling loops. It can be used for the measurement of temperature and flow speed, as well as the controls of valves and pumps.

The temperature control system in Figure 3 is managed with EPICS (experimental physics and industrial control system) software [4]. The control system studio (CSS) is used as a user interface framework for the control systems. Figure 4 shows the temperature values measured with two thermocouples. During the measurement, a thermocouple was put in the room temperature, while the other was put in the water. The red line in Figure 4 shows the water temperature when the water was heated from 20°C to 60°C.

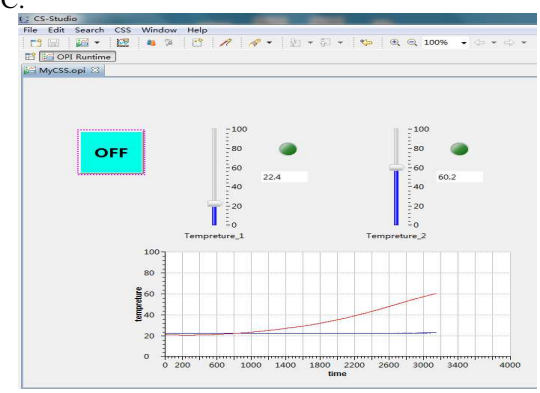


Figure 4: The temperature measurement displayed in the control system studio. Temperatures 1 and 2 represent the values measured with two thermocouples.

Robust Electronics for Equipment Controls

In the China ADS system, the spallation target is located at the centre of a sub-critical reactor core and the control electronics in the equipment control layer for the spallation target has to be put in the reactor room. As shown in Figure 3, the 8-Slot 3U PXI express chassis, PXIe-1082, is used in the equipment control layer. The Intel Core i7 Processor with the lithography of 22 *nm* is used in the control chassis. To be used in the field, a compact control chassis, such as cRIO-9033, can be used

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

to reduce the space occupied by the controller. cRIO-9033 includes a Kintex-7 FPGA device produced by Xilinx company.

An important issue for the application of these advanced semiconductor devices, including RIO-1082 and cRIO-9033, in the radiation environment of the ADS reactor room is how to mitigate the radiation effects caused by nuclear radiation exposure. In the China ADS system which will demonstrate the ADS concept at 10 MW power level with the maximum incore neutron flux of 2×10^{14} *neutron* $/cm^2/s$, the total excore neutron flux in the reactor room is estimated to be 10^7 *neutron*$/cm^2/s$. Our researches in [5] have indicated that these advanced semiconductor devices can be used in the reactor room after being shielded with the thick polyethylene material, where the thickness of the polyethylene should be several dozen centimeters.

.

INTEGRATION OF ACCELERATOR CONTROL SYSTEM AND REACTOR I&C

As a critical part of the China ADS system, the spallation target is used to integrate two totally different facilities, an accelerator and a reactor, into one system. When designing the control system for the target, we have to consider the difference between the accelerator's control system and the reactor's I&C. The following are two examples.

Control Software

The EPICS control system framework, running on a Linux operating system, has been used for the control software in the two accelerator injectors of the China ADS system. Furthermore, EPICS will also be used in the main accelerator of the China ADS system. An open question is whether EPICS control system framework can be used in the reactor. Are the modules used in the reactor's I&C compatible with EPICS? If two sets of software have to be used for the accelerator and the reactor, which one will be used in the overall control system of the China ADS system?

Naming Convention

The naming convention used in the accelerator of the Chinas ADS system is based on a standard developed for the Superconducting Super Collider in the U.S. in the 1980s, and later adopted by other large research facilities, including SNS, FRIB, ITER, CEBAF, and ESS (e.g. [6]-[7]). The format of the naming convention is:

SSSS-BBBB:DDDD-III:TTTIIIXXX

where SSSS is the system name, BBBB is the subsystem name, DDDD is the device identifier, III is the device qualifier, and TTTIIIXXX is the signal part of the name with type (TTT), instance (III) and suffix (XXX).

On the other hand, other naming conventions have been used in a reactor or a power plant, such as CCC (common core code) in England, EDF code proposed by Electricite De France, EIIS (energy industrial identification system) in USA, ERDS (European reliability data system) in European, and KKS (Kraftwerk Kennzeichen system) code in Germany. If two sets of naming conventions have to be used for the accelerator and the reactor, the same naming convention as in the reactor will be used in the spallation target, because the target is located in the center of the reactor core and the cooling loops for the target are put in the reactor room.

CONCLUSION

For the first time, we have reported the design of control system for the spallation target used in an ADS system. The three-layer architecture has been used in the control system, i.e. the top operation layer, the middle network layer and the bottom equipment layer. In the subsystems for the target, such as the target core subsystem, the primary and secondary cooling loops, and the target window and its cooling loop, the main functions of the control system are to monitor and control thermal hydraulic. Therefore, the instrumentation and controls (I&C) for the cooling loops, including the temperature control system, have been discussed in detail. Finally, we discuss the issues about the control software and naming conventions used in the two totally different facilities, the accelerator and the reactor.

REFERENCES

[1] Zhiyong He, Yuxi Luo, et. al, "Design of control networks for China initiative accelerator driven system", the 15th International Conference on Accelerator and Large Experimental Physics Control Systems, Melbourne, Australia, October 17-23, 2015.

[2] Ethernet Ring Protection Switching ITU-T Rec. G.8032/Y.1344, 2010.

[3] Qiang Zhao, Zhiyong He, et. al, "On the measurement of incore neutron flux in accelerator driven sub-critical system", The 23th International Conference on Nuclear Engineering, Chiba, Japan, May 17-21, 2015.

[4] Experimental Physics and Industrial Control System. http://www.aps.anl.gov/epics/index.php.

[5] Wenjuan Cui, Zhiyong He, et. al, "Application of FPGA in the radiation environment of accelerator driven sub-critical system", The 23th International Conference on Nuclear Engineering, Chiba, Japan, May 17-21, 2015.

[6] Anders Wallandera, Lana Abadie, et. al, "ITER instrumentation and control—Status and plans", Fusion Engineering and Design, vol. 85, pp. 529-534, 2010.

[7] Garry Trahern, "Status of the ESS control system", the 13th International Conference on Accelerator and Large Experimental Physics Control Systems, Grenoble, France, 2011.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

STANDARDS-BASED OPEN-SOURCE PLC DIAGNOSTICS MONITORING

B. Copy, M. Zimny, H. Milcent
CERN, Geneva, Switzerland

Abstract

CERN employs a large number of Programmable Logic Controllers (PLCs) to implement industrial processes. These PLCs provide critical functions and must be placed under permanent monitoring. However, owing to their proprietary architecture, it is difficult to both monitor the status of these automates using vendor-provided software packages and integrate the resulting data with the CERN accelerator infrastructure, which itself relies on CERN-specific protocols and configuration facilities.

This paper exposes the architecture of a stand-alone "PLC diagnostics monitoring" Linux daemon, which provides live diagnostics information through standard means and protocols, namely file logging, CERN protocols, and Java Management Extensions. Such information is currently consumed by the EN-ICE MOON supervision software [1] used by the EN-ICE Standby Service to monitor the status of critical industrial applications used in the LHC and the CERN DIAMoN monitoring console [2] used by the LHC operators. Both applications are used daily to monitor and diagnose critical PLC hardware running all over CERN.

PLC DEVICES AND THE RELEVANCE OF INDUSTRIAL STANDARDS

PLCs are off-the-shelf industrial components designed to operate in a sheltered network environment. Moderate, predictable network traffic to and from the equipment and a small number of concurrent network-based accesses (in order to perform control with near real-time precision) are both essential parameters for ensuring reliable behavior on the part of the controller.

Many such controllers are in operation inside the LHC complex. While PLC device implementations are typically subject to IEC standards, ensuring that code is somewhat portable between vendors, there are no such standards for PLC diagnostics. In order to be able to query the internal status of a PLC (such as its last known cycle time, its operational mode or the current usage of network resources), it is necessary to resort to mechanisms specific to each PLC vendor. However, owing to the large number of PLCs deployed at CERN, we cannot simply monitor them and configure their location one-by-one; we must have a common, vendor-independent way of identifying the PLCs under monitoring.

Finally, the data resulting from diagnostic calls made to PLCs is also non-standard. It must be interpreted, transformed to a standard data format, and then stored if it is to be integrated with an established infrastructure monitoring system, such as the EN-ICE MOON platform or the CERN DIAMoN console.

To summarize, obtaining diagnostics information from a PLC is subject to three requirements:

- **Requirement 1:** To minimize as much as possible the impact of status monitoring on the PLC itself and impose absolutely no changes to the program the PLC is running (for instance, we cannot require that the PLC asset owner implements a diagnostics routine for us; it must remain completely transparent).
- **Requirement 2:** To be able to configure the inventory of PLCs to be monitored in a vendor-independent manner.
- **Requirement 3:** To bridge between the vendor-specific diagnostic data formats and a centralized device status history visualization facility.

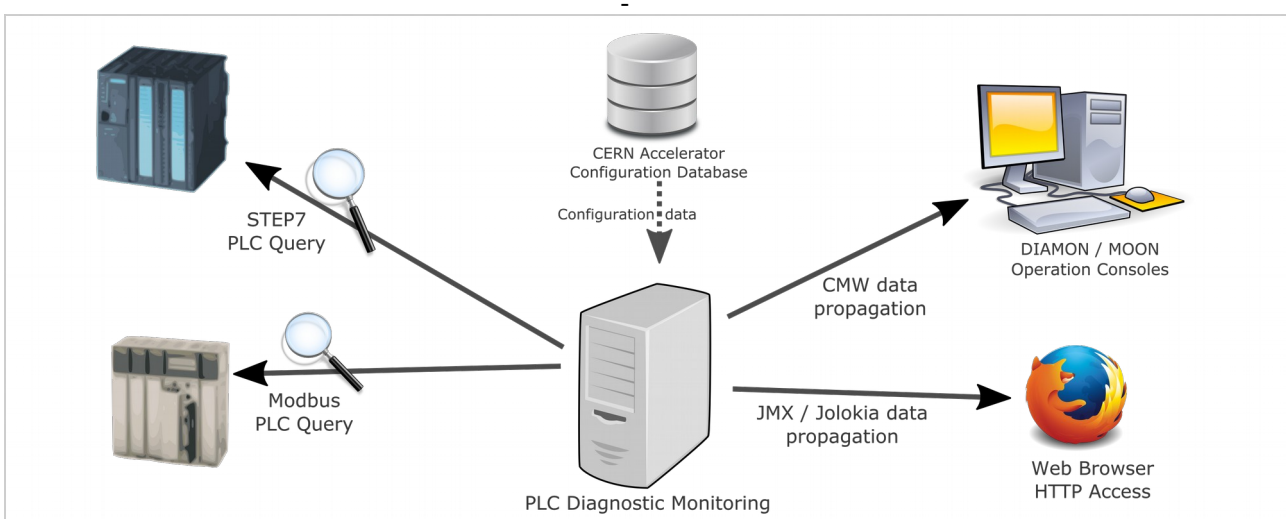


Figure 1: PLC Diagnostic Monitoring.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

QUERYING PLC DIAGNOSTICS INFORMATION

As explained in **Requirement 1**, the action of monitoring equipment must be as innocuous as possible: we must not impact control logic; we must minimize as much as possible the usage of PLC network resources; and we must perform a type of black-box monitoring, so that the PLC asset owner does not have to adapt anything to the monitoring tool.

One simple way to achieve this is to resort to the built-in, yet proprietary, diagnostics capabilities that the PLCs deployed at CERN expose. Their proprietary nature, however, implies implementing a specific approach for each brand of PLC architecture we wish to monitor.

SIEMENS, for instance, offers the programmable API known as SOFTNET [3], which is comprised of a C library and a privileged Linux daemon process that together allow one to send STEP7 network frames to a given SIEMENS PLC. Through SOFTNET, it is theoretically possible to query diagnostics memory areas that are not accessible to other STEP7 implementations. LIBNODAVE [4] is another well-known, yet non-official, open-source library allowing unprivileged access to SIEMENS PLCs and the STEP7 protocol. We will consider in the case study section of this paper the pros and cons of using each SIEMENS-compatible communication library stack.

Schneider, on the other hand, allows access to diagnostics memory areas through regular Modbus [5] calls (provided that the memory addresses and their internal structure are known in advance).

The results of these calls made to diagnose a PLC differ vastly between two vendors – the data structures resulting from a diagnostics query must therefore be unpacked and interpreted specifically, that is, in relation to the PLC's hardware configuration and device manufacturing model.

INTEGRATING WITH THE CERN ACCELERATOR INFRASTRUCTURE

The CERN accelerator infrastructure under monitoring comprises over one hundred and fifty PLC devices. Any device taking part in the accelerator operation (PLCs included) must be featured in the inventory known as the CERN Accelerator Configuration Database [6], a centralized, independent, Oracle database that acts as a device inventory.

In order to fulfil **Requirement 2**, the PLC Diagnostics Monitoring tool must therefore use this central Oracle database as a basis for configuration and ensure that the database carries all required parameters to configure network access to each device.

Furthermore, the CERN Accelerator infrastructure employs the Controls Middleware (CMW) [6] protocol as its lingua-franca. The PLC Diagnostics Monitoring therefore exposes metrics through CMW in order to integrate with the CERN DIAMoN console. Figure 1 above gives a description of how the tool integrates in the CERN accelerator infrastructure ecosystem.

The two mechanisms described above are entirely CERN-specific: neither the CERN Accelerator Configuration Database nor the CMW protocol mean anything outside the CERN environment. Thankfully, the PLC Diagnostics Monitoring tool supports two generic and standard ways of configuring the inventory of PLCs under monitoring:

- flat file configuration, through a local folder containing JSON files describing each PLC;
- and through a secure HTTP interface (based on Jolokia / JMX [7]) that allows one to add and remove PLCs from the inventory, pause the monitoring, and even restart the tool.

COMMUNICATING PLC STATUS TO A CENTRAL MONITORING FACILITY

Even if the PLC Diagnostics Monitoring tool is able to query instantaneous PLC status metrics, it would not be of any use for monitoring purposes if no historical data could be maintained and visualized. Given the tool itself has no intention to act as data storage, the information it queries must be forwarded to a central monitoring facility.

Furthermore, the monitoring tool itself broadcasts a regular “heartbeat” signal (a monotonically increasing counter) used to assert the continuous availability and responsiveness of the monitoring process.

The CERN control room employs two monitoring visualization platforms: the DIAMoN console and the ENICE MOON monitoring platform. Both platforms have been developed at CERN to provide LHC operators with a global, yet detailed, overview of the status of the entire accelerator infrastructure. Both tools can provide historical data on the status of equipment and can integrate PLC Diagnostics data through CMW.

Additionally, with the intention of being reusable outside of the CERN environment, the PLC Diagnostics Monitoring tool offers such for the Java Monitoring Extensions (JMX). JMX is an open-source community specification and reference implementation that describes both a monitoring meta-model and a remote procedure call protocol. Thanks to JMX, any Java developer can easily expose objects and their properties, to be read or modified, and functions to be called remotely. A large number of monitoring tools on the market support JMX out of the box and can incorporate JMX data in their visualizations.

One drawback of JMX is that it is initially Java-centric and thus accessible through Java-specific protocols only: the open-source Jolokia project [8] provides a trivial way to make any JMX-enabled application access through HTTP. Jolokia does not require any code changes, but only the runtime inclusion of its library (and, optionally, a

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

security configuration) to make all JMX metrics accessible through a simple REST interface, exposing data in standard JSON format.

As a result of using Jolokia, it is trivial to include PLC Diagnostics Monitoring status data into a standard monitoring solution, such as the ubiquituous ELK [8] stack (ElasticSearch, Logstash, Kibana).

CASE STUDY: USING SIEMENS SOFTNET IN THE FIELD

As mentioned above, SIEMENS SOFTNET is a Linux-compatible library that allows the interoperation of Linux with SIEMENS PLCs, such as the S7-400 series.

SOFTNET supports integration through two elements:

- a non-privileged application library written in C, on top of which any software developer can write code to interact with a SIEMENS PLC;
- and a privileged daemon process, to be installed by a root Linux user (the daemon must also be configured by a root user and run continuously; it acts as a gateway between PLCs and non-privileged applications).

SOFTNET also requires the use of text files that follow a very specific format; it is thus very sensitive to simple formatting errors (like the usage of tabs instead of spaces for parameter separation). This separation of privileges and the non-standard configuration mechanisms employed make it awkward to deploy and integrate. While the C API invites the development of third-party software, both the daemon and configuration workflow prove to be a serious hindrance.

Finally, through usage of the library, we were able to expose severe shortcomings and crashes when operating in multi-threaded mode, making it only suitable to monitor one PLC at a time in sequence.

During the course of development, SOFTNET was eventually dropped in favour of LIBNODAVE, which provides identical functionality without any of the complications imposed by SOFTNET.

CASE STUDY: IMPACT OF THE MONITORING ON A RUNNING PLC

Measuring the impact caused by the monitoring is essential to prove that it will not adversely affect the performances and control functions of the PLC.

To this end, we have carried out extensive testing against two typical SIEMENS PLCs commonly deployed at CERN. Figures 2 and 3 below summarize our measurements. We can observe that monitoring in itself affects a low-end device more significantly than a high-end one, but overall remain negligible when the PLC is under load (*i.e.* executing complex control logic and communicating with a SCADA software and its administration console).

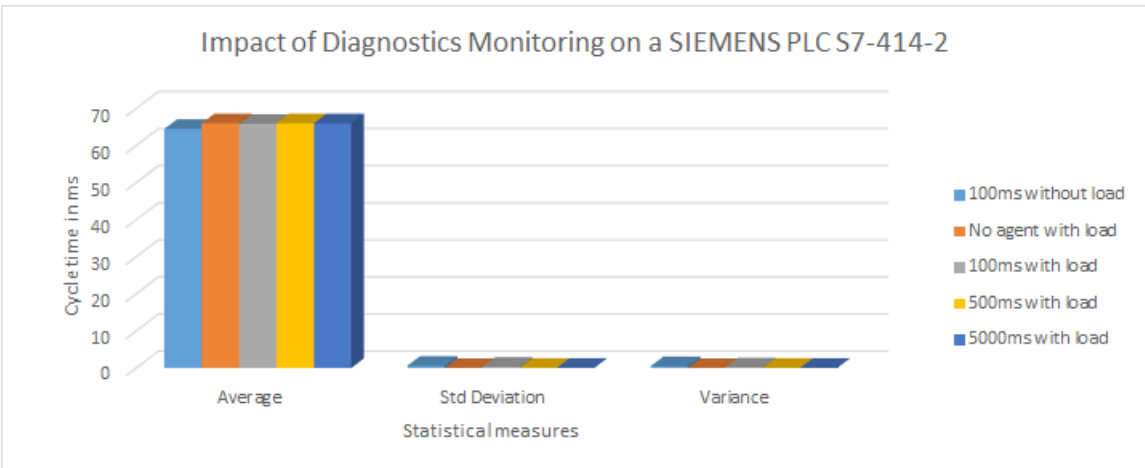


Figure 2: Impact of diagnostics monitoring on a SIEMENS PLC S7-315-2PN/DP.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

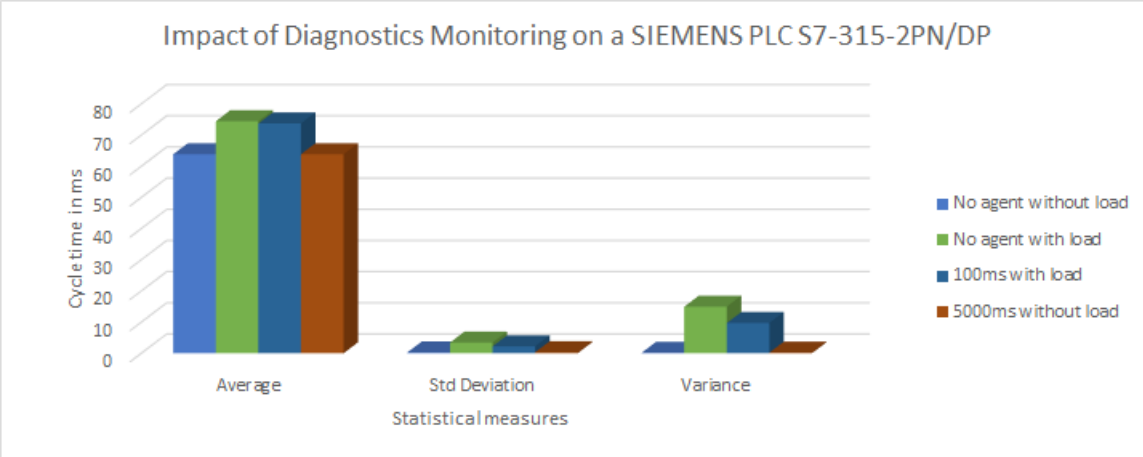


Figure 3: Impact of diagnostics monitoring on a SIEMENS PLC S7-414-2.

CONCLUSION

In conclusion, the PLC Diagnostics Monitoring tool offers a simple way to monitor the health of PLCs deployed in the field. While it integrates with the CERN infrastructure, it can easily be reused in any Linux environment and integrated with standard monitoring tools such as the Hawt.IO console [9] or the ELK monitoring stack.

REFERENCES

[1] F. Bernard et al., "Monitoring Controls Applications at CERN", ICALEPCS'2011, Grenoble, France (2011)

[2] W. Buczak et al., "DIAMON2-Improved monitoring of CERN's Accelerator control infrastructure", Oct 2013, ICALEPCS'13, San Francisco, USA

[3] SIEMENS AG, "SOFTNET S7 Linux", 25 Sept 2014, http://w3.siemens.com/mcms/human-machine-interface/en/customized-products/customized-software/portfolio/pages/softnet-linux.aspx

[4] T. Hergenhahn, "Exchange data with Siemens PLCs", 22 May 2014, http://libnodave.sourceforge.net/

[5] A. Dworak et al., "The new CERN Controls Middleware", 19th International Conference on Computing in High Energy and Nuclear Physics, (CHEP 2012), May 2012, New York, USA

[6] H. Wong et al., "Java Management Extensions", Java Specification Request (JSR-3), rev. 04 Mar 2014, https://www.jcp.org/en/jsr/detail?id=3

[7] R. Huss, et al. "Jolokia : JMX on Capsaicin", 11 July 2015, http://jolokia.org/about.html

[8] S. Banon, "ElasticSearch : you know, for search", 12 June 2012, http://thedudeabides.com/articles/you-know-for-search-inc

[9] J. Strachan et al., "Hawt.IO", retrieved 09 Sept 2015, http://hawt.io/

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

OVERVIEW OF THE MONITORING DATA ARCHIVE USED ON MeerKAT

M. Slabber*, SKA SA, Cape Town, South Africa

Abstract

MeerKAT [1], the 64-receptor radio telescope being built in the Karoo, South Africa, by Square Kilometre Array South Africa (SKA SA), comprises a large number of components. All components are interfaced to the Control and Monitoring (CAM) system via the Karoo Array Telescope Communication Protocol (KATCP). KATCP is used extensively for internal communications between CAM components and other subsystems [2]. A KATCP interface exposes requests and sensors [3]. Sampling strategies are set on sensors, ranging from several updates per second to infrequent updates. The sensor samples are of multiple types, from small integers to text fields. As the various components react to user input and sensor samples, the samples with timestamps need to be permanently stored and made available for scientists, engineers and operators to query and analyse. This paper present how the storage infrastructure (dubbed Katstore) manages the volume, velocity and variety of this data. Katstore is comprised of several stages of data collection and transportation. The stages move the data from monitoring nodes to storage node to permanent storage to offsite storage. Additional information (e.g. type, description, units) about each sensor is stored with the samples.

INTRODUCTION

On each node in the CAM system a monitoring process is responsible for collecting sensor samples for storage. The monitor process communicates with the proxy processes that, in turn, communicate directly with the devices and other systems. The rates at which the monitor processes collect these samples are configured upfront in the central configuration system.

A sample consists of the *sensor name*, *sample timestamp*, *value timestamp*, *status* and *value*.

Sample timestamp is the time at which the CAM system received the sample reading from the sensor. The *value timestamp* is the time at which the acquisition was performed on the sensor and the *value* stored. The *status* field holds the status of the sensor. Timestamps are represented as the time in seconds since the epoch of 1 January 1970 00:00:00 UTC.

The sensor sample storage system is responsible for collecting the samples from the monitor processes. The storage system transports the samples to a central storage node from where they can be queried and archived.

ARCHITECTURE

The storage system is comprised out of several elements (Fig. 1). Each element performs a specific task.

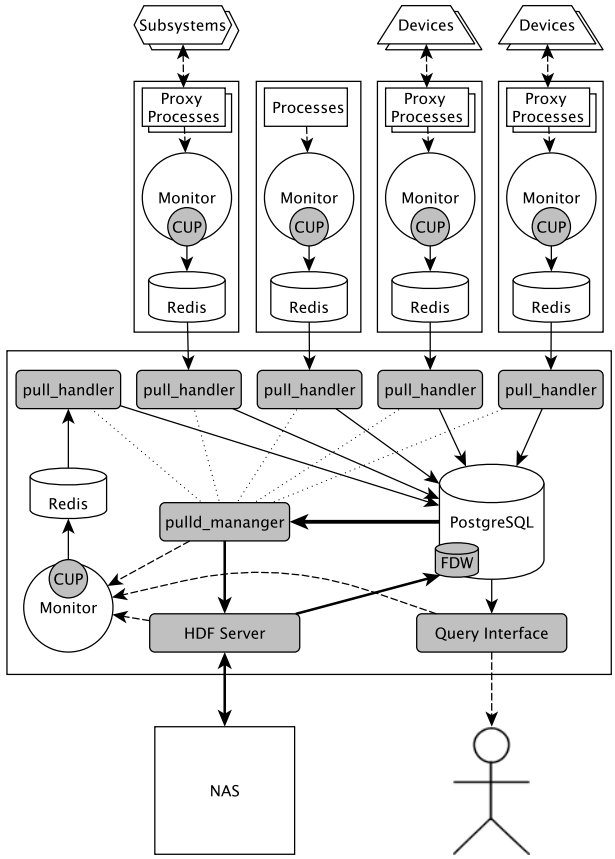


Figure 1: Connections between elements of the storage system.

Memory Buffer

Redis [4], an open source memory database server, is installed on each of the CAM nodes. Redis acts as a buffer to which the monitor process on the node writes sensor samples. On startup and at intervals, the monitor process will write metadata (e.g. type, unit of measure, description etc.) of the sensors to Redis. Internal to the monitor process, a class was developed to manage the writing to the memory database. This cache update class (CUP) runs in its own thread.

Pull Daemon - Pulld

A set of processes on the storage node pulls the samples and metadata out of Redis and stores it into the central database. These processes are collectively called Pulld. On a scheduled basis (currently daily), Pulld analyses the

* martin@ska.ac.za

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

database and samples older than a configured age (currently 2 days) are archived. Pulld starts up a separate process (pull_handler) for each Redis instance in the system. The pull_handler processes are managed by a parent process called pulld_manager. The pull_handler keeps a connection open to Redis and moves samples from there into the central database.

HDF Server

Samples to be archived, along with the relevant metadata of the sensors, are sent to the HDF Server process. HDF Server stores the samples into HDF5 formatted [5] files on the Network Attached Storage (NAS). A new file is created per day, per component of the CAM system. Files are stored in a hierarchical directory structure: a directory for each year contains directories for each month; which, in turn contains a directory for each day of the month. In this directory a file per component is created. The date is added to the file name to allow files to be copied and still maintain a unique name. For example the samples from 07 March 2015 of the *subarray1* component will be archived to *2015/03/07/2015-03-07_subarray1.h5*. Metadata for all of the sensors related to the samples in the file are also stored in the file.

Query Interfaces

Two independent interfaces were developed for applications, (e.g. Web GUI), components and other subsystems to query the storage system. All samples can be accessed through the query interfaces. The storage system is near real-time and lags the actual sensors by less than 1 minute. Archived samples (on the NAS in HDF5 files) are exposed in the database as a table called *samples_archived*. This table is a 'foreign table' in PostgreSQL parlance, and a foreign data wrapper (FDW) was developed for the HDF Server interface. This allows any archived sensor samples to be retrieved as if it is a row in the database. This is naturally not as fast as retrieving samples stored directly within the database system, but the performance is sufficient. The query interfaces use only the database API to access sensor samples, thus making it possible to develop powerful capabilities for the query interfaces in a few lines of Structured Query Language (SQL).

One of the query interfaces is developed to use KATCP as its access protocol. This interface is used by many of the internal CAM components to get historical sensor data and lists of sensor names. It provides filtering and produces the result in a format ready to be consumed by the components. This query interface runs on the storage node.

The second query interface runs on the portal node and provides an HTTP REST [6] -compliant interface and uses Javascript Object Notation (JSON) for encoding. This interface is mostly used by the web based graphical user interface (GUI). The GUI allows operators, scientists and engineers to create plots of any historical sensor data. The users of the GUI can search for sensor names using a regular expression syntax.

PULL VS. PUSH STRATEGY

One of the key differences in the architecture compared to other similar systems [7] is in the strategy used for moving samples from the nodes to the central storage system.

In a push system when a sample is available the sample is sent (pushed) to the central storage system. This is typically done with a distributed queue or via direct connections to the database.

In the architecture of the MeerKAT sensor samples storage system, a much simpler pull strategy was selected. This allows the monitoring processes and the storage systems to be completely decoupled. The storage system only needs to know the address of the memory buffer; it need not know which sensors or sampling strategies are associated with a monitoring process. This allows the important control and monitoring activities of the CAM system to evolve over time to best suit the telescope requirements, without needing to alter the storage system. For the monitor process this results in a very simple implementation that is unobtrusive and extremely fast.

The pull_handler implementation is also fairly uncomplicated. Samples are retrieved from Redis in as large a batch as possible and stored to the central database. Once the samples are stored, pull_handler removes the samples from Redis and retrieves another batch of samples. By batching samples it is possible to attain a much higher transfer rate and to balance the utilisation on the central database better. There are multiple pull_handlers all writing to the database in batches.

Another advantage of the decoupling of the components is that the storage system and monitoring processes can be restarted independently with no effect on one another. This is advantageous at startup, when developing and for fault-finding purposes.

CENTRAL DATABASE

PostgreSQL [8] is used for the central database. Many other database management systems were considered, most notably MongoDB [9] a NoSQL database system. It was found that PostgreSQL suited the needs of the sensor samples storage system best.

The pull_handlers write batches of samples to the database using the COPY FROM command rather than INSERT. This hits the SQL parser only once and thus write the samples more efficiently.

Horizontal partitioning (sharding) is employed and samples are written to different tables. A shardkey is calculated by taking the first part of a sensor name up to the separator character "_". This shardkey has no logical meaning within the CAM system and is only used within the database.

Child table creation in the partition is determined by three parameters; which, are also used to build up the name of the table.

1. A table contains only a days worth of samples.
2. A table is only associated with one shardkey.
3. A table is only associated with one pull_handler.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Separating tables per day and per shardkey has the advantage that when samples have been archived and need to be removed from the database, they are organised in such a manner that the table can be dropped. Using the DROP command is much less resource intensive and faster than using the DELETE command over the same samples.

The chosen shardkey distributes the usage of the tables so that only one pull_handler writes to a table. To guarantee this, the internal ID of the pull_handler is used as part of the table name. Thus there is no opportunity for write contention on any of the child tables.

When a child table is created, constraints are placed on the table. The shardkey and the minimum and maximum values of the sample timestamps are used as the constraints. These constraints help the database query parser to limit the tables used when processing a query. Well-formed queries where shardkey, minimum sample timestamp and maximum sample timestamp are given perform as well as if done directly against a standard database table.

CONCLUSION

We conducted research into many different database management systems of different types. It was concluded that no single system would fulfil all the requirements. A solution based on using Redis database as a buffer on the nodes and PostgreSQL as the central database was proposed, tested and used in the final implementation. It was found that the HDF5 file format was the preferred and most suitable format for archiving the data.

A complete system was developed to move sensor samples efficiently from the nodes where they were collected on to the central storage node where the samples can be archived and queried.

The system was designed in several independent components. Each component is concerned with a specific function. Thus while developing each component, it was possible to focus exactly on solving the problem at hand. The components make testing and fault finding easier. The components will also make it easier to improve the performance of the system as each component can be measured and improved independently of the others.

REFERENCES

[1] R. S. Booth, W. J. G. de Blok, J. L. Jonas, and B. Fanaroff, "MeerKAT Key Project Science, Specifications, and Proposals," *ArXiv e-prints*, pp. 1–16, 2009. [Online]. Available: http://arxiv.org/abs/0910.2935

[2] L. van den Heever, "MeerKAT Control And Monitoring - Design Concepts and Status," in *Proceedings of ICALEPC 2013*, ser. MOCOAAB06, 2013.

[3] S. Cross, R. Crida, T. Bennett, M. Welz, and T. Kusel, "Guidelines for Communication with Devices," 2012. [Online]. Available: http://pythonhosted.org/katcp/_downloads/NRF-KAT7-6.0-IFCE-002-Rev5.pdf

[4] "Redis homepage," Sep. 2015. [Online]. Available: http://redis.io

[5] M. Folk, G. Heber, and Q. Koziol, "An overview of the HDF5 technology suite and its applications," *Proceedings of the EDBT/ . . .*, pp. 36–47, 2011.

[6] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000. [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[7] T. Shen, R. Soto, P. Merino, L. Peña, A. Barrientos, M. Bartsch, A. Aguirre, J. I. Alma, and A. D. Cordova, "Exploring No-SQL alternatives for ALMA monitoring system," in *Proceedings of ICALEPC 2013*, ser. WECOBA06, 2013.

[8] "Postgresql homepage," Sep. 2015. [Online]. Available: http://www.postgresql.org

[9] "Mongodb homepage," Sep. 2015. [Online]. Available: http://www.mongodb.org

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

UPGRADES TO THE INFRASTRUCTURE AND MANAGEMENT OF THE OPERATOR WORKSTATIONS AND SERVERS FOR RUN 2 OF THE CERN ACCELERATOR COMPLEX

A. Bland, S.T. Page, CERN, Geneva, Switzerland

Abstract

The Controls Group of the CERN Beams Department provides more than 400 operator workstations in the CERN Control Centre (CCC) and technical buildings of the accelerators, plus 300 servers in the server room (CCR) of the CCC. During the long shutdown of the accelerators that started in February 2013, many upgrades were done to improve this infrastructure in view of the higher-energy LHC run. The Engineering Department improved the electrical supply with fully redundant UPS, on-site diesel generators and for the CCR, water and air cooling systems. The Information Technology Department increased network bandwidth for the servers by a factor of 10 and introduced a pilot multicast service for the video streaming of the accelerator status displays and beam cameras. The Controls Group removed dependencies on network file systems for the operator accounts they manage for the Linacs, Booster, PS, ISOLDE, AD, CTF3, SPS, LHC and cryogenics. It also moved away from system administration based on shell scripts to using modern tools like version-controlled Ansible playbooks, which are now used for installation, day-to-day re-configuration and staged updates during technical stops.

INTRODUCTION

The Controls Group of the Beams Department at CERN manage disk-based Linux workstations and servers for the operation of all of CERN's accelerators, cryogenics and technical infrastructure.

The workstations are business-class Personal Computer (PC) desktops ranging from 2006 to 2015 models running Scientific Linux CERN 6 (SLC6, based on Red Hat Enterprise Linux 6). Eighty of them are concentrated in the CERN Control Centre (CCC) on the Prévessin site in France and the rest are distributed across more than 100 technical buildings including underground areas of the Large Hadron Collider (LHC). All workstations can run the software of any part of the accelerator complex; restrictions on what can be done from where are enforced by the Role-Based Access Control system.

The servers are almost entirely in the CERN Control Centre server room (CCR) with a small number two kilometres away on the Meyrin site in Switzerland for redundancy and recovery purposes. The majority were upgraded to SLC6, only ten still run SLC5.

Although the accelerators themselves have well-defined year-end technical stops and around 2 or 3 technical stops during the year for upgrades, the technical infrastructure operation (electricity, cooling and ventilation and safety installations) and cryogenic parts of the control system must run 24 hours a day, 365 days per year.

TIMELINE

Most of the upgrades were performed between the start of Long Shutdown 1 (LS1) [1] in February 2013 and the first LHC circulating beams of Run 2 in April 2015, see Table 1. Not all of this time was suitable for infrastructure upgrades as the injectors for the LHC, but not the LHC itself, ran from April 2014 to December 2014.

Table 1: Principal Dates

Date	Activity
Q1 2006	First accelerator operation from the CCC
Q3 2008	LHC first circulating beams
Q1 2010	LHC first physics at 7 TeV
Q1 2013	Long Shutdown 1 starts
Q2 2014	Injector startup
Q2 2015	LHC Run 2: first circulating beams and physics at 13 TeV
Q1 2019	Long Shutdown 2 expected, 18 months duration [2]

Another complication for upgrades between the start of LS1 and April 2014 was that not all accelerators stopped; the Compact Linear Collider Test Facility (CTF3) and Linear Accelerator 4 (Linac4) tests had to continue, plus the LHC Magnet Test Bench (MTB) continued operation.

OPERATIONS REDUNDANCY

The Technical Infrastructure (TI) operator environment has been replicated in the fire station on the Meyrin site with four workstations, each with two screens. The main TI telephone number can be switched to this installation. One TI shift per month is run from the fire station to ensure that all activities are possible from there.

Similarly the firemen's environment in the fire station has been replicated in the CCC with five desktops, the Terrestrial Trunked Radio (TETRA) system and the hardwired fire alarm panel. One shift per week is run from the CCC to test the installation.

ELECTRICAL NETWORK

From first operation of the CCC in 2006 up to LS1, powering was already possible either from the Swiss or French national electrical network. Diesel backup if needed came from the Meyrin site and three sets of Uninterruptible Power Supplies (UPS) were locally installed on the Prévessin site providing N+1 redundancy.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

However, there was only one power line from the UPS into the CCC. The server room cooling and ventilation could not be powered from diesels or UPS which implied that when running from these sources, many of the servers would need to be switched off within half an hour to prevent overheating. Workstations only had one UPS-supplied power source which would be shed after 15 minutes. Servers had two UPS power sources, one of which would be load shed after 15 minutes and the other when the UPS batteries were empty.

A major project by the Electrical Group of the Engineering Department (EN/EL) during LS1 [3] made the following improvements:

- Installed three diesel generators on the Prévessin site providing N+1 redundancy.
- Constructed a new electrical building divided in two with a two-hour concrete firewall providing two sets of UPS (2N+1) and associated switchgear (2N).
- Two power cable routes into the CCC, separated to the greatest extent possible.
- Two completely independent sources of power to all essential equipment in the CCC and the CCR racks.
- Connected critical equipment with only one power input such as workstations for technical infrastructure, cryogenics and access to auto transfer switches which seamlessly allow operation to continue in the event of a supply path failure.
- Consequently, maintenance of the electrical system can be performed while continuing service.

No down time of the rack-mount and blade servers occurred during this upgrade as they have at least dual power supplies, so it was possible to remove one of the old power inputs then add the first new one, remove the second old input and finally add the second new one.

It is to be noted that the electrical supervision system runs on servers in the CCR.

COOLING AND VENTILATION

The pre-LS1 cooling and ventilation of the CCR used the chilled water supply coming from the neighbouring Super Proton Synchrotron (SPS) technical building, so maintenance in that building, power cuts, emergency stop tests and failures led to overheating of the servers. The amount of air that could be ventilated out of the building was very small, so even when cold outside, the system administrators would have to start a portable fan and open the doors.

A major project by the Cooling and Ventilation Group of the Engineering Department (EN/CV) during LS1 [4] made the following improvements:

- Two separate cooling and ventilation systems in CCR. The cooling and ventilation systems of the critical CCR and telecom (network) rooms have been separated from the non-critical system for CCC operators and meeting rooms.
- Cooling power supply is backed up by a diesel generator and UPS power.
- Redundant power supply for the cooling system of the CCR and the telecom room which can be powered from either of the two CCR power sources.
- Added dedicated chillers for the CCR and telecom room with redundant two-out-of-three chiller configuration where there is a chiller on stand-by.
- Water-cooled rack doors for HP ProLiant blades remove the concentrated heat in the server area.
- Dedicated cold air entry in front of server racks and hot air removal behind.
- Consequently, maintenance can be performed while continuing service.

Due to the extra depth of the water-cooled rack doors, the central row of racks in the CCR had to be moved by 30 cm and this was performed with no cut in the service. Around 20 servers not needed during LS1 were stopped to reduce the heat load on the two portable fans used while the cooling system was being upgraded. No other down time occurred.

It is to be noted that the cooling and ventilation supervision system runs on servers in the CCR.

NETWORK

The Communication Systems Group of the Information Technology Department (IT/CS) installs and manages essentially all of the Ethernet outlets, cables, fibres, switches and routers needed for accelerator operation. A minor exception is that typically each HP blade enclosure contains two 10 gigabit/second switches which are the responsibility of the Controls Group.

During LS1 IT/CS:

- Increased the number of Ethernet outlets by 25% in the CCC and the CCR.
- Re-cabled and increased the number of patch panels between the switches and the outlets.
- For the CCC, upgraded older HP ProCurve 3400 switches with a 1 gigabit/second copper uplink to the 3500 model with optional 10 gigabit/second uplink.
- Installed one HP ProCurve 8206 router in each of the three rows of server racks.
- Connected 10 gigabit/second downlinks from the new routers to each of our blade switches.
- Installed redundant routers in parallel with both the main CCC/CCR router and the IT Computer Centre router for the accelerator network. Now maintenance can be performed on one of each of these sets of routers while continuing service without the typical 10 minute down time previously experienced during firmware upgrades.
- Enabled multicast routing in the CCC/CCR accelerator network router as a pilot service. These "targeted broadcasts" allow efficient video streaming of the accelerator status displays and beam cameras to any workstation in the CCC as the network load between the router and switches of the sender and client does not increase with the number of clients - streams are only sent once. In case of problems with the pilot service, video streams will fall back to the pre-LS1 use

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

of our Video on Demand service via a central server, at the cost of higher network bandwidth.

During the CCC re-cabling, patch panel and switch upgrades, the TI operators moved to their backup location and the small amount of cryogenic operation occurring at that time was performed from the Cryogenics local control rooms at the LHC points.

The move of the servers from the main CCC/CCR router to the three new HP ProCurve 8206 routers gave only 10 seconds of network loss for each server.

The redundant router installations needed a restart of each of the switches under them, typically provoking around 90 seconds of network loss for all connected hosts.

WORKSTATIONS

From 2010 onwards, the CCC workstations were business-class HP Elite 8100 PCs (Intel Core i7-860 2.8 GHz processor, Lynnfield generation, 8 cores + threads total, 8 GB RAM, 250 GB hard disk). This had the optimal processor performance for the time, but did not have any on-board graphics processor, so all of them had either one or two Nvidia Quadro NVS 295 business graphics cards to drive the one to four monitors of a typical CCC workstation.

In 2015, a joint specification with the IT Department for office PCs for CERN chose the Optiplex 9020 Mini Tower Business-class Desktop from Dell. The specific requirement for controls use was the support of Intel Active Management Technology (AMT) [5] which permits remote restart and out-of-band BIOS management of PCs operating below the Operating System (OS) layer with no extra Ethernet connection; this saves long trips to the LHC underground areas simply to power back on a workstation for an update.

The chosen model has an Intel Core i7-4790 3.6 GHz processor, Haswell Refresh generation, 8 cores + threads total, 16 GB RAM and 256 GB Solid-State Drive (SSD). As 3 monitors can be driven with the on-board Intel graphics, 90% of the CCC workstations need no external graphics cards to be added, reducing heat load, spares requirements and cost. At the same time as the 110 CCC workstations (including 30 running Windows 7) were upgraded, the 285 monitors were also renewed. The recuperated Elite 8100 PCs, being already AMT capable, have been reused to upgrade the non-remote manageable workstations dating from 2006 in the technical buildings and LHC underground areas.

Accounts

Around 1000 user accounts have access to the accelerator control system, typically with home directories on Andrew File System (AFS) servers managed by the IT Department or on Network File System (NFS) servers managed by the Controls Group in the CCR.

For Run 2 of the accelerator complex, the operator accounts have local home directories physically on the hard drive or SSD of the workstation. This gives isolation from difficulties accessing the central NFS infrastructure and reduces network load. In rooms such as the CCC with access control, the workstation performs an auto-login to the desired account. The login starts the Java Common Console Manager which provides a per-Operations-team menu of programs that can be run, defined in the Controls Configuration Database (CCDB).

SERVERS

Most of the HP DL380 G5 rack-mounted servers installed since the startup of the CCC in 2006 have been retired. The replacements are currently HP ProLiant BL460c Gen9 half-height blade servers (two Intel Xeon E5-2630 v3 2.4 GHz processors, Haswell generation, 32 cores + threads total, 64 GB RAM, two 600 GB SAS hard disks in a RAID 1 array), 16 of which can be installed in an HP BladeSystem c7000 enclosure. Systems needing a large number of disks such as NFS servers and boot servers have been upgraded with rack-mounted DL380 Gen9 hardware (two Intel Xeon E5-2637 v3 3.5 GHz processors, 16 cores + threads total). The server usage is shown in Table 2.

Table 2: Server Usage

No.	Hardware	Usage
80	Blade	Beam control and technical infrastructure based on Java
130	Blade	Industrial controls based on Siemens WinCC OA
5	Blade	Testing and measurement using National Instruments LabVIEW
10	Rack-mount	Network File System servers
5	Rack-mount	Boot servers for the 1500 diskless front ends
8	Blade	LHC real-time orbit feedback and software interlocks
12	All types	Spares and test bench
20	Blade	Development, web, samba access from Windows, interactive login etc.
10	Desktop	Media centres for CCC wall displays
20	Desktop	Accelerator status displays

CERN General Machine Timing PCI receiver cards are needed in servers for two timing-critical areas:

- Proton Synchrotron (PS), SPS and LHC software interlock system (4 servers).
- LHC real-time orbit feedback (4 servers).

Early in LS1, these were upgraded from rack-mounted systems to HP Gen8 Blades with the PCI timing receiver cards installed in HP BladeSystem PCI Expansion Blades.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

SYSTEM ADMINISTRATION

Until LS1, a per-OS shell script was run after the base OS was installed to configure a workstation ready for an operator to use it or to run Java processes in the case of a typical server. The same per-OS shell script was also run early in the morning of every working day to keep the configuration up-to-date.

During LS1, a major investment was made by the system administrators to switch these per-OS shell scripts to the Ansible configuration management system [6].

The host name of every machine we manage is defined in a group in the Ansible inventory file. In our case, we generally map groups of machines to roles such as workstation, media centre, server, WinCC Open Architecture server, NFS server etc.

Ansible playbooks, written in YAML (a human-readable data format), code the specific idempotent (repeatable with the same result) tasks needed to configure the characteristics of a role. Ansible has built-in modules for installing software packages, configuring services, scheduling tasks and adding, removing or modifying lines in configuration files. Any functionality not implemented as a module can be coded with a line of shell script or a custom module can be developed, typically in the Python language. Tasks can have a "when" condition permitting them to be skipped if not applicable. This has allowed us to have one Ansible playbook that manages all the operating systems we use: RHEL5, SLC5, SLC6 and CentOS 7. This means that a new requirement for the control system is added by default to all operating systems, but can, if needed, be coded differently for each one. Any errors in a playbook stop execution, but they can be ignored if required. The end result of the use of Ansible is that system administration becomes more structured and less subject to the individual shell script author's coding style than before.

Ansible's more traditional usage is for a system administrator on a central management host to push out updates to potentially hundreds of machines in parallel. In two hours during our one-day technical stops, we have been able to update all 400 workstations and 100 servers from SLC6.6 to SLC6.7, which typically takes about half an hour per machine including the final reboot.

Ansible, being agentless and using OpenSSH for remote node management, was found to be the best programmable infrastructure choice for our environment.

VERSION CONTROL

Before LS1, the configuration files and scripts needed for the management of the workstations and servers were either using file-based Revision Control System (RCS) or simple dated backup copies, accessed live over NFS. If more than one system administrator was modifying configurations at the same time, there was the possibility of conflicting changes.

During LS1, the key configuration files and in particular the daily Ansible playbook, have been put into a Git [7] version control repository. These files are updated from the Git server via the network onto each machine every day before the daily Ansible run.

System administrators working on a new feature for the control system can make a "branch" in the Git repository which they can safely check on a chosen test bench machine without affecting the production environment then later “merge” into production. All configuration changes are traceable to the author and their intended action.

CONCLUSION

A massive upgrade campaign with a strong focus on redundancy, reliability and efficiency of operation and minimal downtime during implementation was completed during Long Shutdown 1 to achieve the requirements of Run 2 of the LHC.

The powering, cooling and network bandwidth upgrades performed during LS1 allow a considerable expansion of the number of servers during Run 2 and the system administration based on Ansible and Git will scale to the needs.

With the Electrical and Cooling and Ventilation Groups there has been an excellent symbiosis, helped by the fact that their supervision system is running on the servers managed by the Controls Group in the CCC server room.

ACKNOWLEDGEMENTS

Staff and contractors of the EN/EL, EN/CV and IT/CS Groups for the careful planning and implementation of the upgrades.

The Beams Department Operation Group for their patience during the disturbance.

Mikhail Grozak working as a CERN Project Associate wrote the initial Ansible implementation.

REFERENCES

[1] K. Foraz et al., “LS1 First Long Shutdown of LHC and its Injector Chains”, 5th International Particle Accelerator Conference, Dresden, Germany, pp. 1316 (2014).

[2] M. Bernardini, K. Foraz, “Long Shutdown 2 @ LHC”, LHC Performance Workshop, Chamonix, France, pp. 290-293 (2014).

[3] F. Duval, “LS1: electrical engineering upgrades and consolidation”, Chamonix 2012 Workshop on LHC Performance, Chamonix, France, pp. 248-251 (2012).

[4] M. Nonis, “EN-CV during LS1: upgrade, consolidation, maintenance, operation”, Chamonix 2012 Workshop on LHC Performance, Chamonix, France, pp. 252 (2012).

[5] AMT website: http://www.intel.com/technology/platform-technology/intel-amt/

[6] Ansible website: http://www.ansible.com/

[7] Git website: http://git-scm.com/

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

PAST, PRESENT AND FUTURE OF THE ASKAP MONITORING AND CONTROL SYSTEM

Malte Marquarding, CSIRO Astronomy and Space Science, Epping, Australia

Abstract

The Australian Square Kilometre Array Pathfinder (ASKAP) is CSIRO's latest radio interferometer located in the Mid West region of Western Australia. It is in the final phase of construction with Early Science Program starting in the middle of 2016. This fully remotely operated telescope is one of the pathfinder telescopes of the Square Kilometre Array (SKA) building knowledge about infrastructure, technologies and scalability. This paper presents another incremental status update on previous reports. It focuses on software lessons learned and modifications arising from the initial six antenna test system. ASKAP's software consists of two major components, the Telescope Operating System (TOS) and the Central Processor (CP). This paper addresses the TOS, highlighting the monitoring and control aspects of the system.

PROJECT STATUS

The full ASKAP telescope consists of 36 dishes of 12m in diameter hosting phased array feed (PAF) detectors. The hardware is located in a specially designed central building which addresses self-generated radio interference. An initial set of six PAFs, the Boolardy Engineering Test Array (BETA) has been installed and commissioned. The evaluation of this system lead to an improved design with new 36 beam PAFs and an updated digital back end including RF over fibre. This is called ASKAP design enhancement (ADE) and resulted in most of the hardware being moved from the antenna dish pedestals to the central site building connecting to the dishes through fibre optic links. 12 ADE antennas with end-to-end integration of TOS and CP will form the Early Science Program platform. 25% of the total time will be assigned to science project during the time the rest of the telescope will be commissioned. It will produce full data products with only the reduced number of antennas separating it from full ASKAP operation.

Currently the project has evolved to encompass the following active test and production platforms.

- A Parkes 12m dish hosting prototype a PAF. This system is dedicated to engineering tests and new PAF developments such as improvements to the current system as well as next-generation SKA prototyping. Several prototypes have been tested. The software to operate this facility has been stable and remained largely unchanged for several years.
- The BETA system at MRO which host Mark 1 phased array feeds. It is used for beam forming research and small science projects run through the ASKAP Commissioning and Early Science (ACES) team. It has produced several reviewed astrophysics publications and has proven to be an invaluable tool for control system integration, deployment procedures and data management.
- A Mark 2 PAF prototype system at MRO. This single antennas system is dedicated to engineering testing. It is also used for testing hardware modification to the existing Mark 2 set up. Most recently feedback from ACES has prioritised the use of on-dish calibration devices (ODC) to simplify complicated calibration procedures. One such system is being commission on this antenna.
- A four antenna array hosting Mark 2 phased array feeds. It forms the base of the full production array. Commissioning of these systems is taking place at the moment encompassing interferometer phase closure, moving to 36 electronically formed beams and visibility streaming.

For more information about ASKAP visit the project's home page [1].

PROGRESSING THE ASKAP MONITORING AND CONTROL SOFTWARE

Architecture and Framework Update

As described above ASKAP project went through three major stages. A prototype phased array feed on a non ASKAP antenna co-located with the Parkes 64m telescope which still used for research into beam weight generation and stability. Beam weights are used to create beams according to constraints such as maximising the signal-to-noise for a given configuration. This engineering test system was also used to prototype the software technologies and formed the base line of the current system. As described in our previous update [2], the ASKAP software architecture is supported by two key technologies, EPICS and ZeroC ICE. EPICS handles the control and monitoring aspects through software input/output controllers (IOC), whereas ICE defines the interface between software components. Ice interfaces provide the main service contracts between the Central Processor and the Telescope Operating System (see Fig. 1).

We have been using the EPICS framework since the beginning of the project and have now added several helper modules to abstract common usage in ASKAP. For engineering and operator displays of monitoring and control ASKAP has now fully adopted Control Systems Studio (cs-studio). We have joined the project and ASKAP has become one of the products of the upstream source. As part of the collaboration we are in the final stages of setting up cloud-based continuous integration with open access to all collaboration partners. In addition to working with the cs-studio core pack-

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

age we have chosen the operator interface BOY and created plugins for streamlining visualisation of and access to large number of repeating records. Alarms are handled through BEAST which provides a backend for alarm management and metadata storage associated with alarm handling such as acknowledgment, guidance and broadcast. It provides full cs-studio integration for client visualisation. The overall connection between EPICS integration, user interfaces and alarm handling has proven to be major benefit to operating the instrument and has justified the choice of technologies.

We have added and refined the use of following software packages since the last project update:

- ASYN [3] wrappers for EPICS driver development with customisations
- autosave with analysis tools for extracting changes in realtime values.
- Exclusive use of cs-studio [4] and BOY for GUI development for engineering
- We have moved to the Control System Studio (cs-studio) alarm system BEAST.

ASKAP Design Enhancements

The ASKAP Design Enhancements (ADE) Program introduced several enhancements to the EPICS based Telescope Monitoring and Control System. Firstly a software layer called common library has been added between the hardware and EPICS IOCS. It provides a cleaner and implicitly documented interface definition. ASKAP consists of over 4000 instances of individually addressable hardware components each with hundreds of monitoring and control points. All devices need to be efficiently configured, controlled and monitored as a single entity. To assist with this we developed Composite IOCs and Summary Record to create a scalable control and monitoring point hierarchy on top of the flat structure of EPICS.

Composite IOCs represent logical partitioning of the telescope into higher level groups that can be controlled and monitoring as a single entity. Composites exist for each antenna and a single composite for the entire telescope array (see Fig. 1). Configuration and control is fanned out from the array composite IOC through to the antenna composite and down to antenna subsystems and ultimately each hardware component (e.g. FPGA). Similarly monitoring information is aggregated up from the lower level devices using summary records. The EPICS database for the composite IOCs are largely automatically generated from a single point definition file and handles such things as control of heterogenous subsystems, sequenced control and masking out of antennas or subsystems not in use.

Summary records are automatically generated and provide aggregation of a set of monitoring points and will propagate alarm status. Hardware that is disabled, not present or not in use is masked out dynamically at runtime. Summaries exist at all levels in the monitoring hierarchy and include summaries of summaries. E.g. at the top level Array Composite IOC a single summary point can represent the alarm state of all temperatures in the system.

Table 1: ASKAP Monitoring and Control System Specifications Update

	BETA (6 Ant)	**ASKAP (36 Ant)**
Total Number of I/O Points	30,000	800,000
Total Number of Archived I/O Points	25,000	800,000
Monitoring data archival rate	130 GB/year	1.5 TB/year
Monitoring data archival rate max	1 kHz	1 kHz
Highest "soft" control loop rate	1 Hz	1 Hz
Estimated number soft-IOCs	35	230
Estimated number soft-IOC Linux computers	8	11
Science "raw" data output rate	400 MB/s	3 GB/s

With the introduction of the common library abstraction and automation has also been introduced to the generation of EPICS records for the individual IOCs. Record metadata such as monitor archiving policies, alarms and cs-studio BOY elements can be generated through the build process. This simplifies repetitive elements across a large number of antennas, subsystems, cards or points. A summary of the current control system specification is described in Table 1. This presents an incremental update to the previous specifications.

Development Tools Revisited

The complexity and interaction between the different development groups in hardware and software was not adequately supported by Redmine. We have adopted JIRA [5] as the issue management system which allows for sub-projects with issues freely movable between these. It also provides the tools for better planning of milestones and releases. This underpins the TOS move towards agile development techniques. For continuous integration a minor move has been undertaken from hudson to jenkins [6]. This tools is also used to create release deployment artefacts. For our TOS target platform debian linux jenkins automatically generates packages for code release. The CP code base has been moved into a separate repository as target platforms and developer audiences are very different. Connection is maintained via the Ice service contract interfaces.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

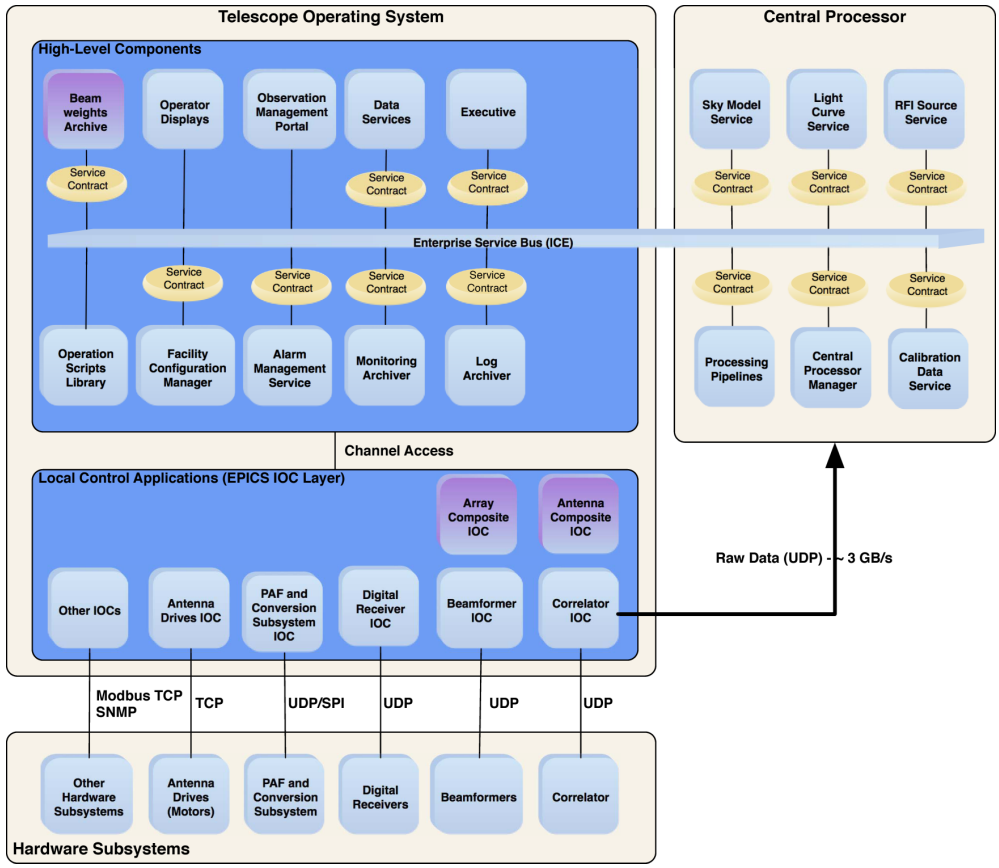


Figure 1: Logical view of the ASKAP software as implemented in latest version of TOS. Purple components are new additions.

CURRENT AND FUTURE CHALLENGES

Improvements and Upgrades

- TOS and CP integration and commissioning will be a major effort as these two software components are hosted at different physical sites with different host organisations. Another related task is to handle configuration management across all of ASKAP from firmware revisions to operational parameters.
- The current drives motor IOC was developed before the introduction of composites and lacks the benefits of aggregation and fan out over large number of similar devices. A re-design is required to scale to the full ASKAP array.
- We are in the process of moving to cs-studio version 4 which addresses issues encountered during current version. cs-studio is changing the implementation of BOY which will affect our automation and customisation.

Beam Weights

As phased array feeds are cutting edge technology in astronomy the understanding of dealing with these instruments is still a rapidly evolving process with open constraints. The TOS needs to cope with frequent changes to requirements and operations while maintaining reliability. We have updated the TOS architecture to add a beam weights component to deal with this (see Fig. 1). Data rates for capture and the beam weights processing can be comparable to the data rates for the standard (visibility) data. To make it valuable a rich set of metadata needs to be maintained as well. This can have deeper impacts on the architecture and design which still have to be evaluated. New auxiliary devices such as ODCs are coming online to assists in the beam forming methods and operations. This is the exciting new technology field in radio astronomy which will keep scientist and engineers busy for a long time.

REFERENCES

[1] ASKAP website `http://www.atnf.csiro.au/projects/askap`

[2] J.C. Guzman, Status of the ASKAP Monitoring and Control System, ICALEPCS'11, Grenoble, October 2011 `http://accelconf.web.cern.ch/AccelConf/icalepcs2011/papers/frcaust04.pdf`

[3] ASYN EPICS driver framework website `http://www.aps.anl.gov/epics/modules/soft/asyn`

[4] Control System Studio website `http://controlsystemstudio.org`

[5] JIRA software development tool website `https://www.atlassian.com/software/jira`

[6] Jenkins Continuous Integration `https://jenkins-ci.org`

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

THE LASER MEGAJOULE FACILITY: CONTROL SYSTEM STATUS REPORT

J. Nicoloso, CEA/DIF, Bruyères le Châtel, 91297, Arpajon, France

Abstract

The Laser MegaJoule (LMJ) is a 176-beam laser facility, located at the CEA CESTA Laboratory near Bordeaux (France). It is designed to deliver about 1.4 MJ of energy to targets, for high energy density physics experiments, including fusion experiments. The commissioning of the facility was achieved in October 2014. This paper gives an overview of the general control system architecture, which is designed around the industrial SCADA PANORAMA, supervising about 500 000 control points, using 250 virtual machines on the high level and hundreds of PCs and PLCs on the low level. We focus on the rules and development guidelines that allowed smooth integration for all the subsystems delivered by a dozen of different contractors. The integration platform and simulation tools designed to integrate the hardware and software outside the LMJ facility are also described. Having such tools gave us the ability of integrating the command control subsystems regardless the coactivity issues encountered on the facility itself.

LMJ FACILITY

The LMJ facility covers a total area of 40,000 m2 (300 m long x 150 m wide). It is divided into four laser bays, each one accommodating 5 to 7 bundles of 8 beams and a target bay holding the target chamber and diagnostics. The four laser bays are 128 m long, and situated in pairs on each side of the target chamber. The target bay is a cylinder of 60 m in diameter and 38 m in height. The target chamber is an aluminium sphere, 10 m in diameter, fitted with several hundred ports dedicated to laser beams injection and diagnostics introduction. Numerous diagnostic instruments are placed in the target chamber around the target to record essential measurements and observe the target behaviour during its implosion. These diagnostics are the prime tools for the physicists to determine the characteristics of the plasma under study.

LMJ CONTROL SYSTEM

LMJ Control System functions

The main functions of the control system are shots execution and machine operations: power conditioning controls, laser settings, laser diagnostics, laser alignment, vacuum control, target alignment, target diagnostics.

All these components are triggered with a high precision Timing and Triggering system [1].

The control system has also a lot of other major functions: personnel safety, shot data processing, maintenance management.

Conducting a shot is composed of two phases: first a master countdown prepares the machine and secondly an automatic sequence [3] executes the shot from the power conditioning charging to the target implosion.

The master countdown has an expected duration of about four hours and the final automatic sequence lasts a few minutes from power conditioning charging to shot execution.

The master countdown coordinates manual operations or automatic programs that prepare the machine: automatic settings computation [4] and associated downloading, laser and target alignment, diagnostics preparation. This can take 2 or 4 hours.

Then, when the laser is ready, the automatic sequence is started: the power conditioning is charged. This takes a few minutes. Then the computer system hands over to the electronic timing system that guides the laser pulses from the master oscillator sources to the target through the amplifiers and transport sections. This takes about 1 microsecond.

General Architecture

The LMJ control system has to manage over 500 000 control points, 150 000 alarms, and several gigabytes of data per shot, with a 2 years on line storage.

It is composed of a dozen of central servers supporting about two hundreds of virtual machines at the central controls level and about 450 PLC's or rack mount PC's at low levels.

Hardware Architecture

From the hardware point of view the LMJ control system is constituted of two platforms located in two different buildings:

- one for system integration (PFI), which is in operation in a dedicated building and consisting of a clone of the operational control system at the supervisory levels and a mixture of simulators and real controllers for representing low levels controls and real equipment [2].
- The operational platform, consisting of two sub-platforms: a small one for integrating the laser bundles (PI) and one for normal operations (PCI).

For each platform, two redundant Alcatel-Lucent OmniSwitch cabinets provide redundant Gigabit attachments to twelve subsystems backbones and main servers (Fig. 2).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

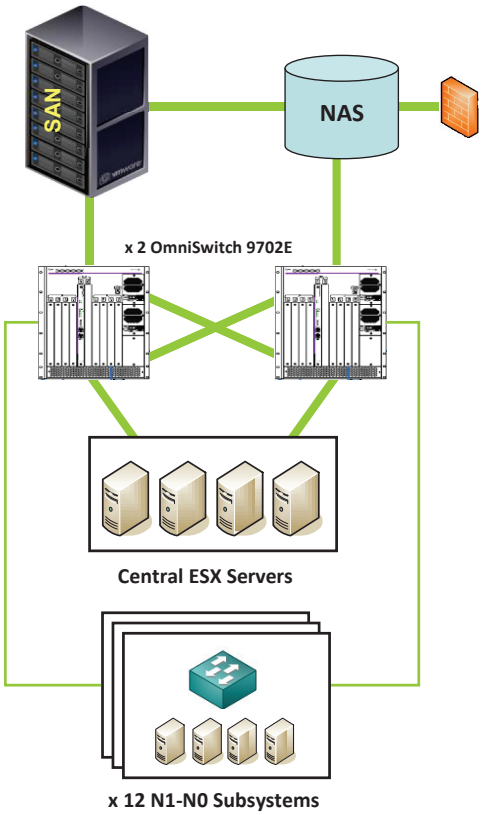


Figure 1: LMJ Network architecture.

On each platform, virtual independent contexts are configured using Virtual Routing and Forwarding technologies (VRF): on the integration platform this allows to simulate different test contexts at the same time with identical IP address spaces, and on the operational platform this allows simultaneous operation from the operational control room and the integration one.

N1, N2 and N3 layers are virtualized using VMware and DataCore solutions. Each platform consists of one virtualization infrastructures (Fig. 3) composed of:

- 2 DataCore servers, each one managing 12 To of disks,
- 6 to 8 ESX Dell PowerEdge R815 servers, with 4x12 cores and 128 Go of RAM,
- 1 VCenter Server to manage the VMware infrastructure.

Each of these infrastructures is dimensioned to execute several hundreds of virtual machines (Fig. 6).

Software Architecture

All command control software developed for the supervisory layers uses a common framework based on the industrial PANORAMA E2 SCADA from Codra.

In this framework the facility is represented as a hierarchy of objects called "Resources". Resources represent devices (motors, instruments, diagnostics…) or high level functions (alignment, laser diagnostics). Resources are linked together through different kinds of relationships (composition, dependency, and incompatibility) and the resources life-cycle is described through states-charts. Control-Points, alarms, states and functions can be attached to any resource (Fig. 1). Dedicated mechanisms manage the resource reservation and propagate properties and states changes into the tree of resources through relationships. There are about 200 000 resources in order to describe the entire LMJ.

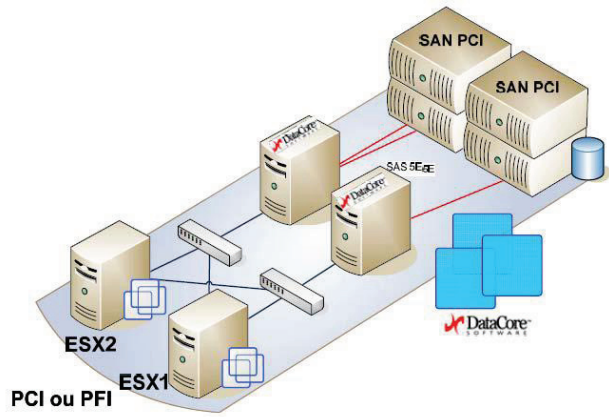


Figure 2: LMJ Virtualization architecture.

The framework implements the data model described above as .net components inside the PANORAMA E2 SCADA and adds some common services to the standard features of PANORAMA E2:

- resources management,
- alarms management,
- lifecycle states management,
- sequencing [3],
- configuration management,
- event logging

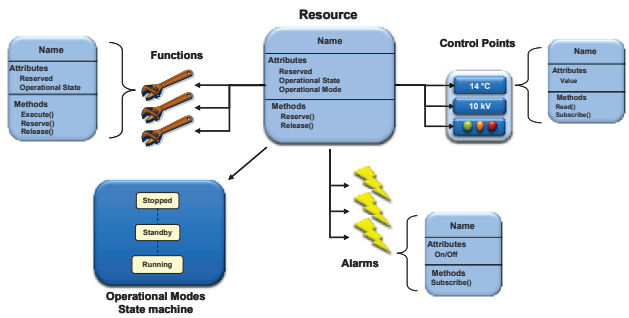


Figure 3: Data model used by the LMJ supervisory System.

INTEGRATION STRATEGY

Design of the LMJ was entrusted to a dozen of major contractors. Each one supplied all the command control associated its delivery, including the supervisory levels. CEA itself was responsible for definition design standards and subsystems integration.

To achieve that, an integration policy was clearly defined and imposed to all contractors.

This policy was based on a for steps process :

- External interface definition,
- Factory tests,

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- Integration between subsystems on an dedicated platform (PFI),
- Functional integration on the LMJ facility.

External Interface Definition

Each contractor's supervisory subsystem communicates with the other supervisory subsystems and with the central one that allows to drive the whole facility and shot sequences.

All these communication interfaces between supervisory subsystems are formally defined and managed by CEA in a centralized database and exported to contractors as xml files (Fig. 4). Dedicated tools allow to automatically generate from these xml files:

- Supervisory subsystems' skeletons.
- Supervisory subsystems' simulators called SITEX, which allow contractors to simulate the external environment they are communicated with, during software development and factory commissioning.
- The device settings configuration database that contains, for each subsystem, all the settings used by shot sequences to configure laser devices and measurement instrumentation.

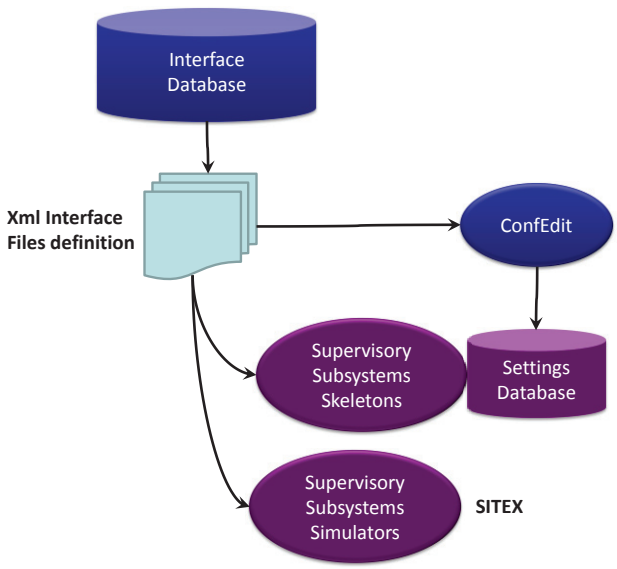


Figure 4: Software Interfaces Management.

Factory Tests

Factory tests are centered on functional requirements and have to demonstrate the ability of the supervisory subsystem delivered by each contractor to drive a full LMJ configuration (e.g. 22 bundles).

As the equipment for all bundles cannot be present at factory, this configuration must at least include a real set of equipment for one, and device simulators are used to simulate the others. These simulators also allow the simulation of abnormal behaviours that would be impossible to obtain with real equipment (safety, cost, etc.).

As it was described before, all external supervisory subsystems they communicate with, are simulated with SITEX generated and delivered by CEA from the centralized interface database.

Integration Tests on PFI

Each Subsystem Control system is delivered on the Integration Platform (PFI) and tested in three successive phases:

- Phase 1: standalone phase in which the other subsystems are simulated by SITEX (Fig. 5).
- Phase 2: integration phase in which each subsystem is connected to the high level supervisory system and the other subsystems already installed. During this phase each interface between one supervisory subsystem and the others is tested.
- Phase 3: the global commissioning phase. During this phase the global functioning is tested using all subsystems and shot sequences.

Functional Integration on the LMJ Facility

As bundle commissioning will be a long process spread over several years, a dedicated control room were designed to allow new laser bundles commissioning, while already commissioned bundles are operated for shots and fusion experiments from the main control room.

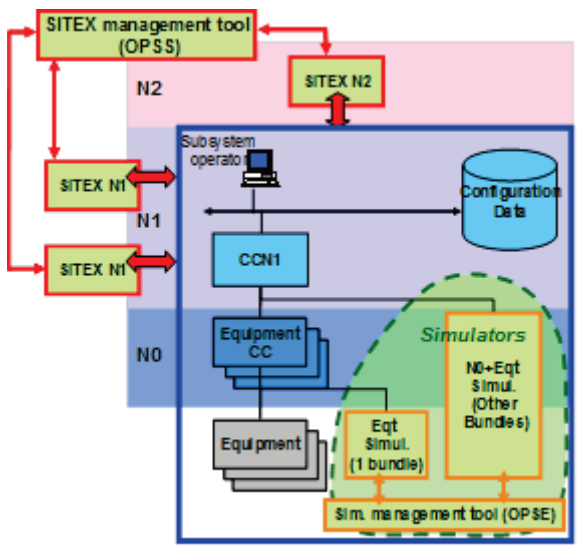


Figure 5: Different kinds of simulators used on the Integration Platform.

A second supervisory system was installed in this room and is connected to new bundles, as their commissioning is beginning. From this control room two kinds of tests are conducted:

- First, testing of each subsystem driven by the contractor which is responsible for. It is the first time that the subsystem software package controls the real equipment using the nominal wiring. In a standalone mode, with the same integration tools used at factory, the contractors check the behavior of equipment with the facility wiring.
- Then, testing of the whole bundle driven by CEA. As soon as all the subsystems are integrated on the

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

bundle, CEA makes sure that all subsystems and the high level supervisory system work well together.

When this phase is achieved, the new bundle is switched to the main control room and final testing allows to demonstrate that ability of the new bundle to be operated with the others.

LMJ PROJECT STATUS

Commissioning of the first bundle was achieved on time in October 2014. The first laser target interaction experiment was done on fall 2014 and a second experimental campaign on April and May 2015. Next one is scheduled on last trimester 2015.

From the command control point of view, the integration control room was commissioned during this summer (as first bundle's integration took place in the operational control room). Next step will be to commission in fall a new version of the supervisory software with a complete 22 bundles configuration and then deploy this version in the integration control room in order to begin the second bundle's integration. This step will end the commissioning process of the command control by itself and began its maintenance phase.

REFERENCES

[1] P. Raybaut, V. Drouet, JJ. Dupas, J. Nicoloso, "Laser Megajoule timing system", ICALEPCS 2013, San Francisco, USA, (2013).

[2] J.P. Arnoul, J. Fleury, A Mugnier, J. Nicoloso, "The Laser Megajoule control command system integration platform", ICALEPCS 2013, San Francisco, USA, (2013).

[3] Y. Tranquille, "The LMJ system sequences adaptability", ICALEPCS 2015, Melbourne, Australia, (2015).

[4] S. Vermersch, "How to computerise laser settings on the Megajoule facility", ICALEPCS 2015, Melbourne, Australia, (2015).

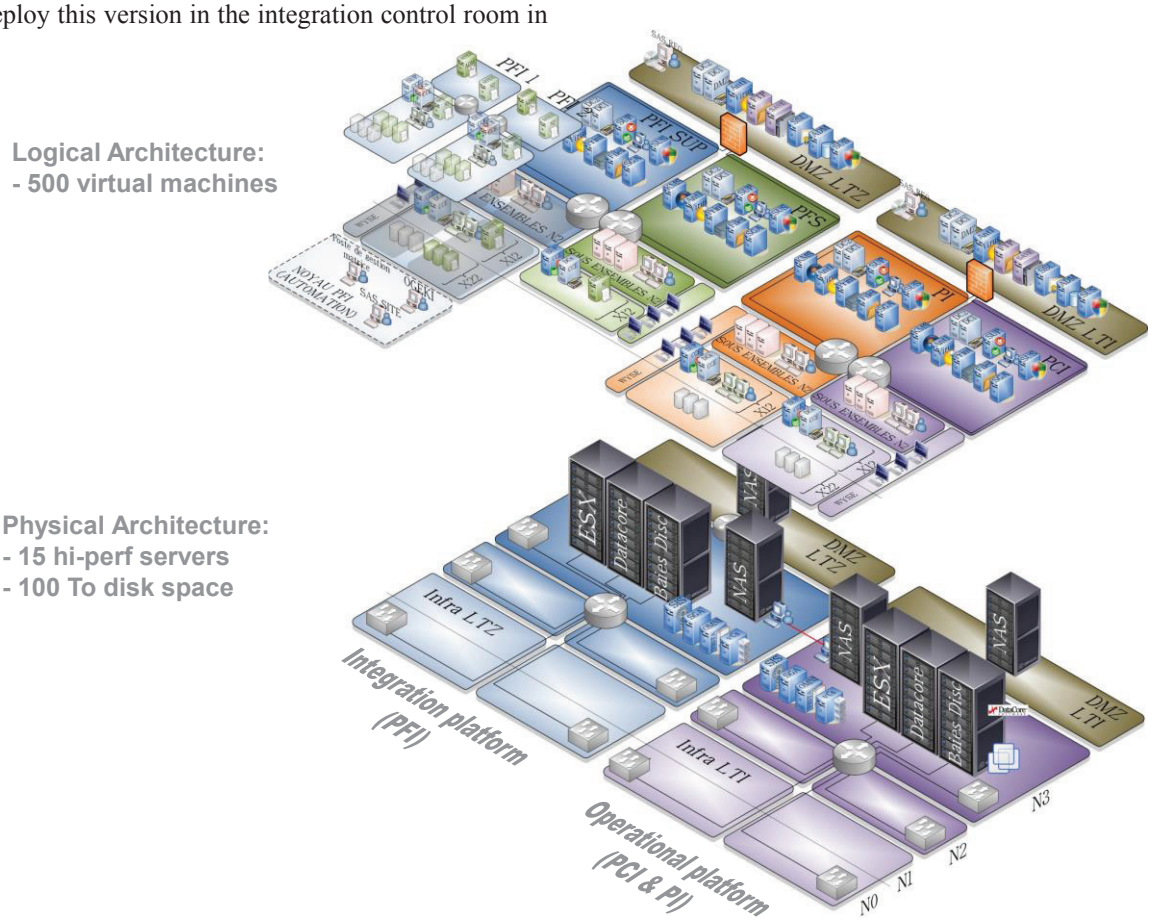


Figure 6: LMJ Central Controls Physical and Logical Architecture, showing the integration platform and the operational one.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

OVERVIEW AND STATUS OF THE SWISSFEL PROJECT AT PSI

Markus Janousch, Arturo Alarcon, Kristian Ambrosch, Damir Anicic, Alain Bertrand, Kurt Bitterli, Helge Brands, Patric Bucher, Tine Celcer, Pavel Chevtsov, Edwin Divall, Simon Ebner, Martin Gasche, Alexandre Gobbo, Colin Edward Higgs, Fabian Hämmerli, Thomas Hövel, Tadej Humar, Guido Janser, Gaudenz Jud, Babak Kalantari, Rene Kapeller, Renata Krempaska, Daniel Lauk, Michael Laznovsky, Christian Lüscher, Hubert Lutz, Dragutin Maier Manojlovic, Fabian Märki, Trivan Pal, Werner Portmann, Simon Rees, Thierry Zamofing, Christof Zellweger, Dirk Zimoch, Elke Zimoch , Paul Scherrer Institut, Switzerland

Abstract

Recently, the installation of the components for the free electron laser SwissFEL has started at the Paul Scherrer Institute (PSI). In March 2016, beginning of the injector commissioning is planned and first lasing is foreseen a year later. New hardware, like VME64x-boards (IFC 1210, an P2020 based intelligent FPGA controller from IOxOS) and -crates (Trenew), timing system (from MRF with advanced features), motion controllers (Power PMAC from Delta Tau, and MDrive from Schneider), among others, as well as modern field buses, pose great challenges to the controls team. The close interaction of machine- and experiment-components require advanced software concepts for data-acquisition, -distribution, and -archiving. An overview of the project will be presented and the different HW and SW solutions based on the experience gained from preliminary implementations at other facilities of PSI will be explained. First results of the HW commissioning at the SwissFEL will be reported.

INTRODUCTION

The Paul Scherrer Institut (PSI) is the largest research centre for natural and engineering sciences within Switzerland and is located northwest of Zurich, about an half hour's drive away. Research is performed in three main subject areas: Matter and Material; Energy and the Environment; and Human Health. For this fundamental and applied research we use three different accelerator facilities on site and support additional test facilities:

- The facilities consist of one of the highest intensity 570 MeV proton cyclotron, HIPA, that produces particles for fundamental particle physics, muons, and is the source for the neutron spallation source SINQ.
- Another low energy superconducting proton cyclotron is used for the patient treatment systems at the Center for Proton Therapy.
- The Swiss Light Source, SLS, is a 3rd generation synchrotron and houses 18 beamlines.

These accelerators are available for the international community; we host about 2400 external users per year. All these facilities are controlled from a central control room.

Until the end of October 2014 a 250 MeV electron Linac, SITF (SwissFEL Injector Test Facility), was in operation and was used as a pre-study for the free electron laser (FEL) project. Many of its components are used as the injector for the upcoming FEL. The new SwissFEL is currently set-up a few hundred meters east of the main campus of PSI.

Of course, there are several other test-facilities and laboratories related to the accelerator facilities.

This large array of different facilities and set-ups around the laboratory poses quite a challenge to be handled by one Controls group. Therefore a lot of emphasis has been put on standardization on the equipment that is supported and the tools that are being used within the Controls group. However, there are new experiments and test set-ups constantly coming up that need to be supported on the one hand, on the other one has to watch a growing heap of legacy components and code that have to be maintained. E.g. HIPA is operational since over 40 years.

Standardization in the Past

In order to minimize any overhead and maximize the efficiency of the Controls group a strong standardization effort is implemented. Among these standards are: All accelerator and most control systems are based on EPICS [1]. For fast I/O, scalers, etc. we use a VME bus architecture, controlled by a Motorola CPU board running VxWorks 5. The timing system is based on the model 230 of Micro-Research Finland [2]. The main motion controller used is MaxV from Pro-Dex.

Consoles and most servers run Scientific Linux 6.4 and the Linux software is distributed through Puppet. To deploy software for the EPICS system a mechanism developed at PSI is used. A relational database keeps an overview of our hardware inventory and helps in managing the many components at the facilities.

THE SWISSFEL PROJECT

The new SwissFEL is currently set-up a few hundred meters east of the main campus of PSI (Fig 1). The project consists of two consecutive phases.:

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- In the first phase (from 2013-2016) a compact hard x-ray free electron laser with one beamline and 2 planned endstations in the 1 Å range will be build.
- In a second phase a low energy branch will be added to produce soft x-rays.

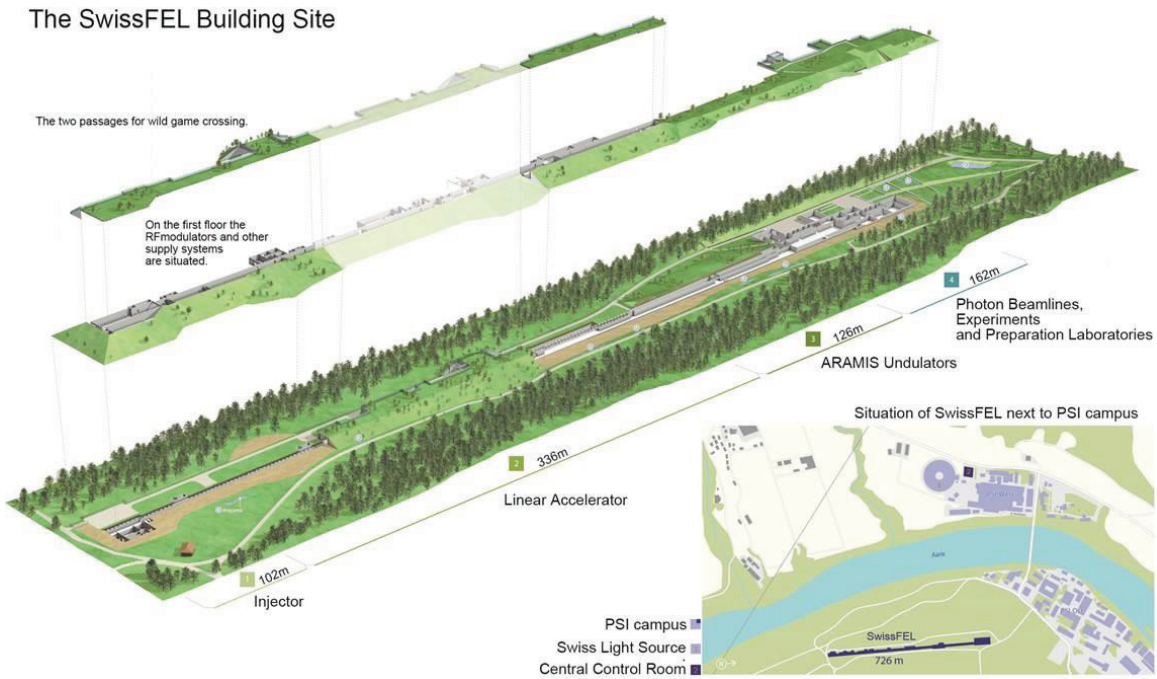


Figure 1: The SwissFEL building site.

The machine itself is housed inside a tunnel roughly at ground level. On top of it the technical gallery is located. The facility is 720 metres long and composed of four sections. The injector with a 3 GHz RF gun, a linear accelerator with 26 C-Band RF modules, an arrangement of 12 planar undulators, and the photon beam line and experimental facilities. So far financing of two user endstations is secured, but a third one is planned.

The key figures of the facility are listed in table 1. The accelerator will be operated in mainly two modes, one with a long pulse of 25 fs length and a charge of 200 pC resulting in a photon intensity of $7x10^{10}$ and a short pulse length of 6 fs with a 4 times lower photon intensity at a photon wavelength of 1 Å.

Table 1: Key parameters of the SwissFEL

Overall Length	720 m
Total electrical power	5.2 MW
Maximum electron beam energy	5.8 GeV
Number of FEL lines	2
Wavelength	1 - 7 Å, 7 - 70 Å
Repetition Rate	100 Hz
Number of Endstations	2 + (1)
Cost	278 MCHF

Installation of the machine components started in September 2015 and the installation of the control system components started at the end of October. Beam commissioning of the injector will start in April 2016, and that of the Linacs about a half year later. First lasing is expected at the beginning of 2017 with users starting to use the facility in fall of 2017.

The PSI Controls group will be responsible to enable the remote operation from the main control room for the accelerator as well as the control of the experiments from their dedicated control rooms (inside the SwissFEL building).

NEW REQUIREMENTS

The new facility poses quite a few new challenges due to new requirements. These are somewhat related:

- Going from circular to a long linear machine means a **highly distributed** system. This results in smaller VME crates and additional bus systems, like EtherCAT, that have to be supported. In addition the SwissFEL will be further away from the main control room than all other PSI facilities. This results in the wish to have remote control over even the smallest devices like power strips.
- SwissFEL is a pulsed machine and therefore a stable, and **reliable timing system** is crucial.
- Due to the pulsed nature and large extend of the machine recording of data has to be **synchronized** by tagging each datum with an event pulse number.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

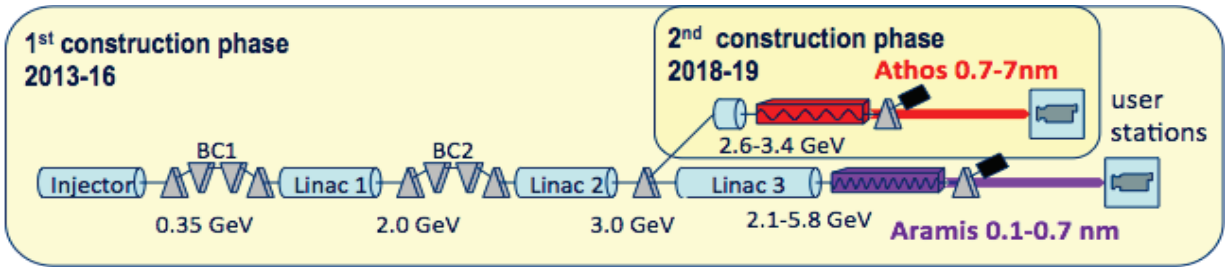


Figure 2: Overview of the SwissFEL accelerator.

- New detectors and high performing cameras produce **large amounts of data**. This calls for a high network bandwidth and new, as well as efficient ways of data-acquisition systems.

FUTURE STANDARD SOLUTIONS

Due to the new requirements and the age of some of the current standards at the other facilities there need to be new solutions for SwissFEL

Network

Throughout the facility there will be 10 Gbit Ethernet networks using copper and glass fibre. A 100 Gbit-connection is available between the main switch of the SwissFEL building, the ones of control room and the two computing centres of PSI. Two 100 Gbit lines a reserved for camera data going directly to the control room.

The usage of EPICS gateways to separate the different parts of the SwissFEL facility has proven to be problematic during tests at the SITF. Therefore we will start with a flat network structure.

VME-Crates

Much care went into the specification of the 7 slot-crates concerning reliability, low noise level on the power lines, and a remote control and monitoring capabilities. The latter is based on an I^2C protocol. The data and control are available in EPICS through the stream device driver [3]. The crates are provided by Trenew.

VME Bus and FPGA Controller

The board of choice for the VME bus controller is the IFC 1210 from IOxOS SA, Switzerland and was designed in a collaboration effort between the controls and low-level-RF groups of PSI and IOxOS. It is a 6 U VME64x single board computer containing a Freescale Power PC P2020 dual core and Xilinx Virtex-6 central and Spartan-6 IO FPGAs. Extension slots allow the insertion of 2 XMC, 1 PMC, or 2 FMC mezzanine cards. As an operating system RTLinux is used.

The IFC 1210 is used for fast D/A signal processing, timing, Power-Supply control and connection to EtherCAT-Systems. About 250 boards will be deployed at the SwissFEL.

Timing and Event System

This system is based on the new 300 model from Micro-Research, Finland [2]. Event masters come in a VME form factor and the receivers are available as VME and PCIe. The latter ones are used in the Windows systems that read-out the cameras along the facility, in the Linux systems for the detectors, and in the motion controllers that need a synchronized movement, e.g. Wiresanners. The system runs internally at 142.8 MHz which results in a resolution of 7 ns. Finer time-shifts can be obtained from dedicated delay modules. The system shows a low RMS jitter of 5 ps. The new system is capable of delay and drift compensation that is caused through the whole path of the event transmission between the master and each receiver due to different path lengths and local temperature variations. It allows the instant modification of the sequence of the event stream during each shot of the SwissFEL. The system is used to synchronize the different processes at the facility with high precision and is crucial for the data-acquisition. Instant delay shift mechanism is another new feature that is crucial for the machine protection system (MPS).

The EPICS driver has been updated recently, to take advantage of the new functionalities of the novel hardware.

Motion Control

For coordinated, as well as synchronized movements, motors are controlled through the PowerBRICK based on the PowerPMAC system of DeltaTau.

Regular movements are done with the help of MDrive system from Schneider Electric. They come with the power driver directly attached to the motor chassis and provide an Ethernet connection as a communication interface. A system without this interface is also available from Schneider Electric called MForce. For this an in-house power driver has been built and allows to control up to eight axes.

The EPICS motor record has been updated to include the new functionality of the hardware.

Camera Systems

For diagnostic, monitoring and alignment many cameras will be used at the SwissFEL. Up to five high-performing PCO cameras running at 100 Hz will be running during the operation of the accelerator. For

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

reduced needs like beam or laser alignment cameras from Basler will be used.

All of them will be connected to Windows servers that allow first image analysis, provide the camera control in EPICS, do event tagging of the images with a PCIe event receiver, and distribute the images if needed with 100 Hz (the beam repetition rate).

Serial and Low Demand Systems

For slow control and I/O like temperature, switching, etc. we provide different serial interfaces. Among these are Wago [4] and Moxa [5]. Work has also gone into interfacing EtherCat into EPICS [6], as there are needs to enable real time communication with Beckhoff PLCs [7]. This will be used for example to control the RF modulators in synchronisation with the Low Level RF to change phase and amplitude of the acceleration.

Beam Synchronous and High Volume DAQ

New detectors like Jungfrau or Gotthard will be used at the beamlines and experimental station. These 1- and 2-dimensional detectors produce large amounts of data. Furthermore, additional data that reflect basic accelerator parameters are needed to describe the current pulse for the experiments. To handle these requirements a new DAQ had to be developed. Details can be found in several contributions to this conference [8].

CONCLUSION

The SwissFEL facility introduces new requirements and challenges to the Controls group of PSI. All of them could be met with new standards in hardware or in software. Controls will be ready for beam commissioning of the SwissFEL injector that starts in March 2016. For the other PSI facilities we will gradually replace the old standards with the new ones were appropriate.

ACKNOWLEDGMENT

The contributions from Cosylab, Slovenia and Mirek Dach GmbH, Switzerland are greatly acknowledged.

REFERENCES

[1] EPICS website: http://www.aps.anl.gov/epics/
[2] Micro-Research Finnland website: http://www.mrf.fi/
[3] EPICS stream device driver website: http://epics.web.psi.ch/software/streamdevice/doc/
[4] WAGO website: http://www.wago.com
[5] MOXA website: http://www.moxa.com/
[6] D. Mayer-Manojlovic, “”,MOPGF027, these proceedings, ICALEPCS2015, Melbourne, Australia (2015).
[7] Beckhoff website: https://www.beckhoff.com/
[8] S. Ebner et al., „Data Streaming - Efficient Handling of Large and Small (Detector) Data at the Paul Scherrer Institute“,WED3O06, these proceedings, ICALEPCS2015, Melbourne, Australia (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

COMMISSIONING OF THE TPS CONTROL SYSTEM

C.Y. Liao#, J. Chen, Y.S. Cheng, P.C. Chiu, C.Y. Wu, C.H. Huang, K.H. Hu, Y.T. Chang,
D. Lee, S.Y. Hsu, C.J. Wang, C.H. Kuo, K.T. Hsu
National Synchrotron Radiation Research Center, Hsinchu 30076, Taiwan

Abstract

Control system for the Taiwan Photon Source (TPS) has been completed in 2014. Commissioning of the accelerator system is in proceeding. Electron beam were stored at the storage ring and emit first light in December 31, 2014. TPS control system adopts EPICS toolkits as its frameworks. The subsystems control interfaces include event based timing system, Ethernet based power supply control, corrector power supply control, PLC-based pulse magnet power supply control and machine protection system, insertion devices motion control system, various diagnostics related control environment, and etc. The standard hardware components had been installed and integrated, and the various IOCs (Input Output Controller) had been implemented as various subsystems control platforms. Low level and high level hardware and software are tested intensively in 2014 and final revise to prepare for routine operation is under way. Efforts will be summarized at this paper.

INTRODUCTION

The TPS [1] is a latest generation of high brightness synchrotron light source which is constructed at the National Synchrotron Radiation Research Center (NSRRC) in Taiwan. TPS consists of a 150 MeV electron linac, a booster synchrotron, a 3 GeV storage ring, and experimental beam lines. Ground breaking for civil construction was held on February 2010. The construction works were completed in April 2013. Accelerator system installation and integration was proceeding in later 2013. The control system environment was ready in half of 2014 to support subsystem integration test and commissioning without beam. After 4 months of hardware testing and improvement, the TPS initiated the commissioning of the booster ring and storage ring on December 2014. On the last day of 2014, the TPS has delivered its first synchrotron light [2, 3].

Adequate and reliable functionality of control system is one of the key to the success of TPS commissioning. Control system for the TPS is based on the EPICS framework [4]. The EPICS toolkits provide standard tools for display creation, archiving data, alarm handling and etc. The EPICS is based on the definition of a standard IOC structure with an extensive library of driver and support a wide variety of I/O cards. The EPICS toolkits have various functionalities which are employed to monitor and to control accelerator system.

The TPS control system consists of more than a hundred of EPICS IOCs. The CompactPCI (cPCI) is equipped with input/output modules to control subsystems as standard IOC. The other kinds of IOCs are also supported by the TPS control system, such as BPM IOC, PLC IOC, various soft-IOC and etc.

To achieve high availability of the control system, emphasis has been put on software engineering and relational database for system configurations. Data channels in the order of 10^5 will be serviced by the control system. Accessibility of all machine parameters through control system in a consistent and easy manner will contribute to the fast and successful commissioning of the machine. High reliability and availability of TPS control system with reasonable cost and performance are expected.

SYSTEM COMPONENTS

The system installation and integration with subsystems for the control system was done [5]. Details of the control system are summarized in the following paragraph.

General EIPCS IOC Interface

There are many different kinds of IOCs at equipment layer to satisfy various functionality requirements, convenience and cost consideration, shown in Table 1. Most of the devices and equipments are directly connected to cPCI IOCs with EPICS. The cPCI EPICS IOC is equipped with the cPCI-6510 CPU board. The cPCI-7452 128 bits DI/DO module is used for BI, BO solution. ADC and DAC modules in IP (Industry pack) module form-factor are used for smaller channel count application, such as insertion devices control. Event system modules are in 6U cPCI form-factor. Private Ethernet will be heavily used as field-bus to connect many devices. Power supplies of all magnets except for correctors are equipped with Ethernet to the EPICS IOC. Multi-axis motion controller with Ethernet interface is the standard for the control system.

Ethernet attached devices are connected to the EPICS IOC via private Ethernet. Devices support VXI-11, LXI, Raw ASCII and Modbus/TCP protocol are connect to EPICS IOC directly by TCP/IP interface. Devices of this category include power supply, temperature acquisition (RTD or thermocouple), digital multi-meters, oscilloscopes, signal generator, and other instruments.

All corrector power supplies are driven by the corrector power supply controller (CPSC) module [6]. The CPSC equips with 20 bits DAC and 24 bits ADC. Two SFP ports supported by the on board FPGA (Spatan 6), these SFP ports are receive correction setting (Aurora and Gigabit Ethernet by using UDP/IP protocol) from fast orbit feedback FPGAs to slow orbit feedback PC, feed-

#liao.cy@nsrrc.org.tw

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

forward correction computer and IOC. Setting command sent to these SFP ports will be added with the slow setting from EPICS CA client.

Table 1: Type of EPICS IOCs

Type	Quantity	Applications
6U CompactPCI	~40	CIA IOCs, Timing, RF, ID, etc.
COM Express Embedded	72 (Atom) + 13 (xScale)	BPMs
EPICS IOC Embedded PLC	13	Pulse Power Supply, MPS
Embedded w PCI/PCIe	10~20	Gateways and Beam-line Timing
Embedded w POE	~5	Image IOCs
Embedded IOP	~122	Corrector PSs
Miscellaneous	~20	Software IOCs, Bunch-by-Bunch Feedbacks, LabVIEW IOCs, etc.

Power Supply Control

TPS power supplies control interface are divided into three categories due to it provides by three different vendors. The small power supplies for corrector magnets, skew quadrupoles are in the range of ±10 Amp categories. This category power supply will be in module form-factor. Each power supply sub-rack can accommodate up to 8 power supply modules. A custom designed CPSC module was installed at control slot of the sub-rack. The CPSC will be embedded with EPICS IOC and provide fast setting SFP ports to support orbit feedback functionality. Power supply modules installed at the same sub-rack will interface to this CPSC module.

The intermediate power supply with current rating 250 Amp is equipped with Ethernet interface. Power supplies are expected to have internal data buffer with post-mortem capability. There are two versions of power supply in this category, sextupole power supply with 16 bits resolution and quadrupole power supply with 18 bits resolution DAC.

Storage ring dipole DC power supply and power supplies for the dipoles and quadrupoles of the booster synchrotron are contracted to Eaton. Each power supply equips with RS-485 serial interface. MOXA serial to Ethernet adapters enable directly interface with the EPICS IOCs. The storage ring dipole will be control via this link. Booster dipole and quadruple power supplies will interface by precision analogue interface as shown in Fig. 1. The DACs and ADCs operated synchronize by the same clock and trigger to achieve better reproducibility. Waveform generate form the DAC on IOC will drive these booster power supplies. This functionality is essential for energy ramping of the booster synchrotron [7]. Control resolution of these power supplies has 18 effective bits.

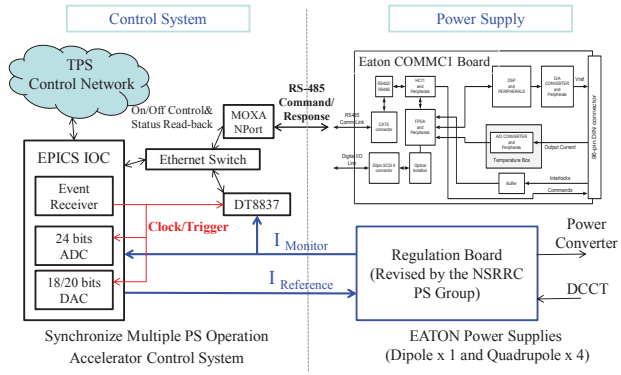


Figure 1: Booster synchrotron AC power supply control interface.

Networking and Timing System

Mixed of 1/10 Gbps switched Ethernet are deployed for the TPS control system [8]. The Gigabit Ethernet connection was delivered at edge switches installed at control and instruments area (CIA). The control network backbone is a 10 Gigabit link to the control system computer room. Private Ethernet is used for Ethernet based devices access which support fast Ethernet and GbE. Adequate isolation and routing topology will balance between network security and needed flexibility. The file and database servers are connected to the control and intranet network, allowing the exchange of data among them. Availability, reliability and cyber security, and network management will continue to strengthen.

The event system consists of event generator (EVG), event receivers (EVRs) and a timing distribution fiber network [9, 10]. The EVG and EVRs can be installed with various universal I/O mezzanine modules to meet different input/output requirements. The mechanical form-factor of EVG and EVRs is in 6U cPCI module. The 125 MHz event rate will deliver 8 nsec coarse timing resolution. Fine delay is supported by the EVRTG which generates gun trigger signal. Its high resolution and low timing jitter provide accurate synchronization of hardware and software across the TPS control system.

Insertion Devices and Front-end

Insertion devices (ID) control for the phase I project include one set of EPU46, two sets of EPU48 [11] and seven sets of in-vacuum insertion devices (two sets of 2 meter long IU22-2m, three sets of 3 meter long IU22-3m, and one set of 3 meter long IUT22-3m with taper functionality). Motion control was done by the Galil DMC-40x0 motion controller. In-house EPICS device support for this motion controller was developed. A cPCI EPICS IOC equips with AI/AO/BI/BO I/O modules were used. All parameters of motion controller will be created as EPICS PVs. Update rate can be up to 200 Hz. This would be useful for feed-forward compensation process. The user interface of insertion device with front-end layout is shown in Fig. 2.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

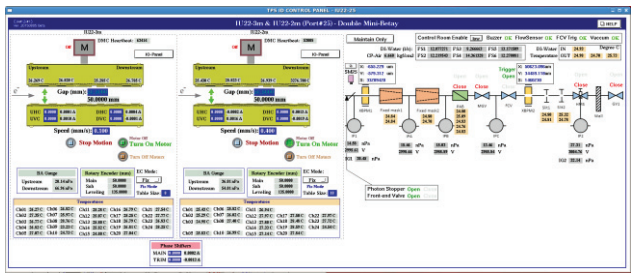


Figure 2: Graph user interface of two insertion devices (IU22-3m + IU22-2m) with front-end.

Diagnostic System

New generation digital BPM electronics is equipped with Ethernet interface for configuration and served as EPICS CA server with 10 Hz data rate. Another multi-gigabit interface will deliver beam position for fast orbit feedback purpose at rate up to 10 kHz. The BPM electronics will also provide post-mortem buffer for orbit analysis during specific event happened like beam loss.

High precision beam current reading and lifetime calculation was done at a dedicated IOC. This IOC will install EVR to received booster cycle timing signals and high resolution IP ADC modules to digitize the DCCT signal and perform beam lifetime calculation.

The GigE Vision digital cameras support for screen monitor [12], synchrotron radiation monitor, X-ray pinhole camera [13] and other applications. Counting type and integrating type beam loss monitors was connected to the control system by counter or ADC modules installed at IOCs.

PLC and Interlock

The PLC was used for most of control system related interlock system [14]. FM3R with embedded EPICS IOC is also used for some applications, such as pulse magnet power supply control and machine protection system (MPS). The MPS collects various interlock signals from local interlock subsystem of orbit, vacuums, front-ends, and etc. The beam disable commands to trip beam or inhibit injection can be distributed to the specific devices or subsystem by the global machine interlock system or uplink functionality of the event system. The summarize control page is shown in Fig. 3.

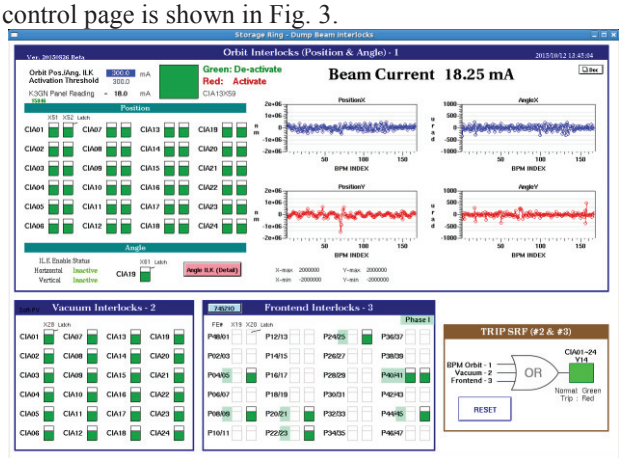


Figure 3: The machine protection system control page.

CONTROL APPLICATIONS

Generic applications provided by the EPICS toolkit will be used for all kinds of applications [15]. Standard tools such as the archive system, alarm handler and save/restore tools are supported. Channel Access (CA) is used as an interface for process variables (PVs) access. Simple tasks such as monitoring, alarm handling, display and setting of PVs are performed using EDM panels and strip tools. Cold start, warm up and shutdown process are done by MATLAB scripts.

Operator Interface

The operator interface level consists of Linux PCs for consoles and servers for various purposes. Operation user interface (OPI) is implemented by EDM, MATLAB and CSS (Control System Studio). The top-layer control page is built by the EDM toolkit shown as the Fig. 4. All control pages can be launched from this page. All control components are located at the foreground of the TPS accelerator illustration.

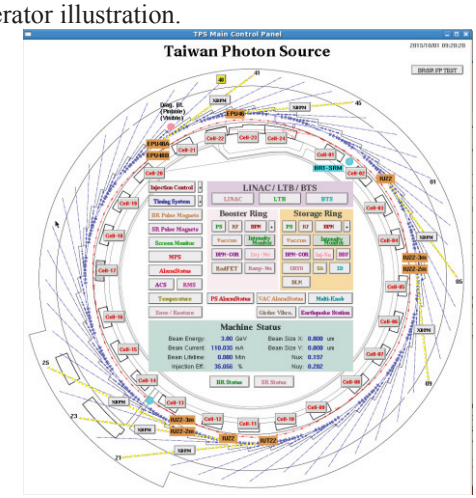


Figure 4: The TPS main control page made by EDM tool.

Save and Restore

To readily restore a set of the machine parameters for subsystems during operation as well as to optimize and record working points for different machine conditions, the mechanism of save and restore is developed. The save and restore function is established by using the MATLAB with the labCA. The various files of grouped PVs (Process Variables) list are created for saving the respective parameter values of each subsystem. The file with PVs and saved parameters is also selectable for resume the settings.

Archive and Logbook

The archive system of CSS (Control System Studio) named BEAUTY (Best Ever Archive Toolset, yet) was built to be used as the TPS data archive system [16]. An archive engine takes PV data from EPICS IOCs via channel access, and stores them in the data storage. The PostgreSQL RDB (Relational Database) was adopted as the data storage for the BEAUTY. Both the historic PVs data and the archive engine configuration are saved into

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

the same RDB. The archived data can be retrieved in a form of graphical representation using the CSS-based data browser. Taking the performance and redundancy into considerations, the storage servers and RDB table structures are tuned relative.

The "Olog" [17] solution is selected for the TPS electronic logbook. The TPS logbook is recorded the progress about commissioning information by the commissioning team and operators, and supports print function to copy data into the logbook for logging.

Alarm Handler

The "BEAST" (Best Ever Alarm System Toolkit) of CS-Studio with the MySQL RDB is adopted as the alarm handler for the TPS, as shown in Fig. 5. A distributed alarm system monitors the alarms in a control system and helps operators to make right decisions and actions in the shortest time. In the CS-Studio alarm system, each alarm is supposed to be meaningful, requiring an operator action. An alarm is no status display that operators may ignore. Each alarm requires an operator to react because the control system cannot automatically resolve an issue.

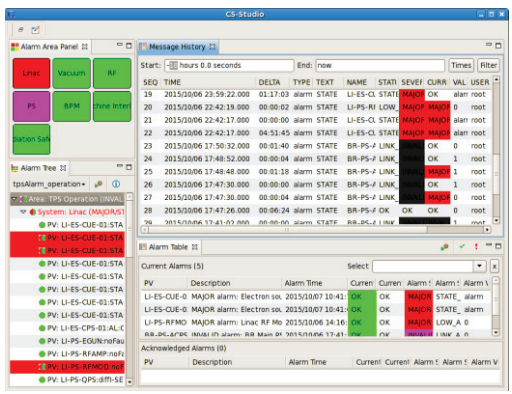


Figure 5: The CS-Studio alarm handler interface.

SUMMARY

The TPS take advantages of the latest hardware and software developments to deliver high performance, rich functionality, and economically control system. TPS control system revised continually during the commissioning and help successful completion the Phase I commissioning. The Phase II commissioning is starting with two superconducting RF cavities and insertion devices. Rich control related applications are developed on going and ready for routine operation.

REFERENCES

[1] TPS Design Book, v16, September 30, 2009.
[2] C.C. Kuo et al., "Commissioning of the Taiwan Photon Source", TUXC3, IPAC2015, Richmond, USA (2015).
[3] P.C. Chiu et al., "The role of Beam Diagnostics in the Rapid Commissioning of the TPS Booster and Storage Ring", MOBLA01, IBIC2015, Melbourne, Australia (2015).
[4] EPICS, http://www.aps.anl.gov/epics/index.php
[5] Y.S. Cheng et al., "Status of Control System for the TPS Commissioning", WPO033, PCaPAC2014, Karlsruhe, Germany (2014).
[6] D-TACQ, http://www.d-tacq.com
[7] P.C. Chiu et al., "Control Supports for TPS Booster Synchrotron", THPRO121, IPAC2014, Dresden, Germany (2014).
[8] C.S. Huang et al., "Network Architecture at Taiwan Photon Source on NSRRC", WPO034, PCaPAC2014, Karlsruhe, Germany (2014).
[9] Micro Research Finland, http://www.mrf.fi
[10] C.Y. Wu et al., "Integration of the Timing System for TPS", THPRO121, IPAC2014, Dresden, Germany (2014).
[11] C.Y. Wu et al., "Control System of EPU48 in TPS", THPRO123, IPAC2014, Dresden, Germany (2014).
[12] C.Y. Liao et al., "TPS Screen Monitor User Control Interface", FPO032, PCaPAC2014, Karlsruhe, Germany (2014).
[13] C. Y. Liao et al., "Synchrotron Radiation Measurement at Taiwan Photon Source", TUPB067, IBIC2015, Melbourne, Australia (2015).
[14] C.Y. Liao et al., "Implementation of Machine Protection System for the Taiwan Photon Source", THPRO126, IPAC2014, Dresden, Germany (2014).
[15] Y.S. Cheng et al., "Control System Software Environment and Integration for the TPS", FPO030, PCaPAC2014, Karlsruhe, Germany (2014).
[16] Y.S. Cheng, et al., "Implementation of the EPICS Data Archive System for the TPS Project", THPEA049, IPAC2013, Shanghai, China (2013).
[17] Olog, http://olog.sourceforge.net/olog

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

STATUS OF THE EUROPEAN SPALLATION SOURCE CONTROL SYSTEM

T.Korhonen, R. Andersson, F.Bellorini, S.Birch, H.Carling, J. Cereijo García, L. Fernandez, R. Fernandez, B.Gallese, S. Gysin, E. Laface, N.Levchenko, M.Mansouri, A. Monera-Martinez, R. Mudingay, A. Nordt, D. Paulic, D. Piso, K. Rathsman, M. Rescic, G. Trahern, M. Zaera-Sanz
European Spallation Source, Lund, Sweden
K. Strnisa, U.Rojec, N.Claesson, A.Söderqvist, Cosylab, Ljubljana, Slovenia

Abstract

The European Spallation Source (ESS) is a collaboration of 17 European countries to build the world's most powerful neutron source for research [1]. ESS has entered the construction phase and the plan is to produce first neutrons by the year 2019 and to complete the construction with 22 neutron instruments and reach the target 5 MW beam power by 2025. The Integrated Control System Division (ICS) is responsible to provide control systems for the whole facility.
The unprecedented beam power and the construction of the facility as a collaboration of the member countries, with many components delivered in-kind presents a number of challenges to the control system. The Integrated Control System Division has to prepare systems and specifications so that the work can be effectively shared between the contributors and staff on-site. The systems need to provide sufficient performance for successful operation of the facility, be standardized to a level that allows easy integration into the facility during a short installation period and have to be maintainable by the in-house staff after the construction has finished.

INTRODUCTION

The European Spallation Source (Fig. 1) is an accelerator-based facility currently in construction in Lund, Sweden. The facility will provide neutron beams for various fields of research, using a variety of instruments in neutron beam lines. Different components of the ESS will be delivered in-kind by several institutions in the member countries. This applies to all parts of the project, including the control system.

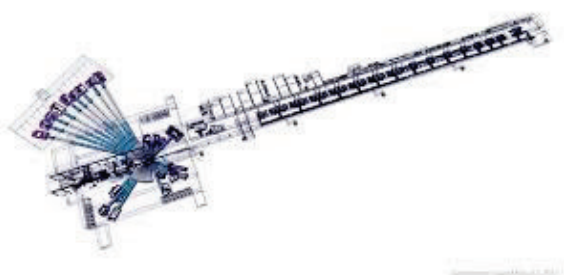


Figure 1: A schematic view of the ESS facility.

In ESS, a high power proton beam hits a spallation target to produce neutrons. The proton beam is generated in a proton linac (Fig. 2) which operates at 14 Hz repetition rate and 5 megawatt average (125 MW peak) beam power. The about 600 meters long linac consists of a proton source, normal conducting ("warm") section, a superconducting section and finally a high energy beam transport section towards the target [2].

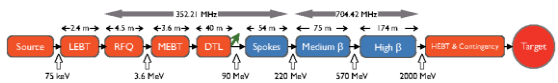


Figure 2: The ESS linear accelerator.

The ESS accelerator produces long pulses of 2.86 milliseconds in normal operation. This pulse lenght allows a great flexibility for designing the neutron experiments.

The accelerated proton beam hits the neutron target (Fig. 3), a rotating tungsten wheel divided into several sectors. The produced spallation neutrons go through moderators and neutron guides to the neutron instruments. The accelerator operation has to be synchronised with the rotation of the wheel so that the beam pulses hit the middle of each sector. A rastering magnet system expands the beam before it hits the target so that the heat load is distributed to a wider area.

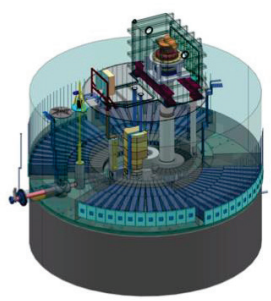


Figure 3: The neutron target of ESS.

Neutron beam lines carry the produced neutrons to the instruments for detection. The neutron beam lines have among other components several choppers on the way to select the neutron energy, or wavelength. The choppers have also to be synchronised with the beam operation.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

THE INTEGRATED CONTROL SYSTEM

The Integrated Control System Division (ICS) is responsible of building control systems for all parts of the facility, from integration of the site infrastructure, the accelerator, target and providing EPICS control of the instrument components. The integration of the instruments and handling the scientific data is the responsibility of the Data Management and Software Center (DMSC), located in Copenhagen. This obviously requires a close collaboration between ICS and DMSC.

ICS will provide the integration of all systems up to (but excluding) the instruments so that the facility can be operated as a single unit from a central control room. This includes a global timing system to synchronise the operation of the whole facility, as well as the machine protection system to protect the facility from beam-induced damage. In addition, ICS is responsible of building the Personnel Safety System (PSS) to protect the personnel from hazards when working in the facility.

ICS will also provide the computer networks for connecting the control systems and a local data centre with servers to run the services required by the machine operation [3].

CONTROL SYSTEM ARCHITECTURE

The control system will be based on the EPICS [4] software toolbox. EPICS will be applied to all parts of the facility, from control and monitoring of the site infrastructure systems up to control of the neutron instruments. All components that require control or monitoring will be integrated into EPICS. At this stage it is difficult to precisely estimate the number of process variables but in any case it will be in the range of a million and probably even higher.

Our goal is to build the system based on the EPICS Version 4 [5]. All our low-level controllers as well as software applications use the version 4 structured data and normative types at the application level and communicate using pvAccess. This enables us to simplify and improve the integration of different systems beyond the pure control capabilities of EPICS version 3. The handling of scientific data in a direct connection to the EPICS systems and services will enable many applications that would otherwise be cumbersome or even impossible to implement. The development of EPICS 4 has progressed to a state where the reliability and performance has been proven in real-life applications [6]. A number of components like process variable gateways need still to be developed to enable running a large facility but they are within reach so the goal is realistic. As a side benefit we hope that this will also help to push the EPICS community forward.

The control system will also provide a number of supporting services. A part of the services is directly related to EPICS, like the archiving system to store time-series data of the process variables. Other services like the Controls Configuration Database (CCDB) provide the support for building, operating and maintaining the control system.

GLOBAL SYSTEMS

Timing and synchronisation of the facility will be based on a global event system, developed by Micro-Research Finland [7] that is used in a large number of accelerator and other similar facilities. The system has gone though several developments since it was first deployed [8]. The ESS will use the latest generation of the event system. The timing system provides a global distribution of RF-synchronised triggers and beam parameter data to the facility. Actions like acquiring data at the device level are synchronised by this system and synchronised timestamps are provided so that all measured parameters can be correlated system-wide.

The timing system consists of a central timing master node that steers the whole operation of the ESS. The timing master sends out a sequence of timing events that trigger the machine operations at proper times. Several parameters for defining the operation cycles are also distributed by the timing master. The master node generates an event stream signal that is distributed to the receivers through a fan-out system. The latest generation of the event system hardware can compensate the propagation delay through the fan-out system to the receivers so that the propagation delay can be kept constant, thus relaxing the requirements for cable routing and temperature stability in the buildings.

Machine Protection is a system to prevent beam-induced damage to the facility and also to minimise unnecessary activation of the components. The part that makes the decisions to allow or stop the beam, the Beam Interlock System (BIS) is connected to a number of local protection and beam monitoring devices and to systems to switch off the beam and thus mitigate the beam loss. The MPS (including BIS) has to be able to switch off the beam in the middle of a beam pulse, in the range of 5 μS, depending on the location where beam losses or system malfunction is detected [9].

Personnel Safety System protects the personnel from hazards caused by radiation or oxygen depletion in the accelerator tunnel or in the target and instrument areas. PSS is a safety-credited system and will be built according to state of the art standards like IEC 61508[10].

HARDWARE STRATEGY

The control system will have a very large number of I/O channels to interface to. To do that in a cost-efficient way, we have decided to apply a hardware architecture consisting of three layers. One layer is used for systems that require very fast signal acquisition and online signal processing, typically implemented in FPGAs or similar, and a capability to transport large amounts of data. These systems also require very short latencies and operate in hard real time. This layer of systems will be built on the MTCA.4 standard [11].

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

For MTCA.4-based applications we plan to provide a standard digital processing platform that can handle most of our fast real-time applications. This will happen in collaboration with the Paul Scherrer Institute in Switzerland and be based on an upgrade of an earlier development [12] which is being used in several applications at PSI. The common platform enables us to share several applications that have already been implemented, as well as co-developing new ones together with other ESS partner institutes. It also simplifies our hard-, firm- and software management.

Many systems require distributed I/O and real-time responses, but at more moderate data rates. Implementing these on MTCA.4 would not be cost-effective and require the development of several components that are not readily available on that platform. For these systems the EtherCAT [13] standard provides a cost-efficient and flexible solution and a wide variety of available I/O modules off the shelf. We plan to use the EtherCAT in two flavours: one is for direct I/O using a regular computer (e.g., a MTCA.4 CPU) as an EtherCAT master and connect it to a number of slave modules. For the cases that require more complicated handling of the bus modules, for instance control of complex motion it seems better to use a dedicated EtherCAT Master from Beckhoff GmbH that provides a lot of software support in its TwinCAT suite that would be hard to implement otherwise.

In addition to the tasks that require real-time, beam synchronised operations there is a large part of more process-type control. For that, industrial PLCs are the most appropriate solution and have also been widely adopted for fields like control of vacuum and cryogenic systems. A standardisation process of the PLC hardware hast recently come to a conclusion. Most systems in the target seem to naturally fit into this category.

SOFTWARE

The ICS will provide a set of services for people who develop or contribute data to the control system.

The most fundamental support for integration and management of the I/O devices and signals is a naming convention that is available for all the system contributors. That has been developed early on and is supplemented by naming service to record, manage and check the syntax of the entered device names[14].

ESS has been participating in the DISCS [15] collaboration that develops software tools for use in accelerator and other EPICS-based laboratories. One of the products from that collaboration that we have adopted and are further developing to fulfil our requirements is the Controls Configuration Database (CCDB). CCDB will be used to store data about control system components and their hierarchies, plus several types of metadata. The CCDB has two foremost goals: to provide means to track an EPICS PV from a control screen down to the physical hardware, through paths of powering, housing and control. The other goal is to store the component and hierarchy data so that it can be used to configure and manage IOCs.

For software development, a unified development environment (Figure 4.) will be provided. The development environment has to take into account that system development happens not only on the Lund site but also in the collaborating institutes. That software has to be integrated into the ICS system when the components are delivered and need to be operated as a part of the system. To satisfy these requirements ICS has developed a system that enables full management of the development process from the Lund site. Users can set up computers (virtual or real) using Vagrant [16] and Ansible [17] playbooks that have been set up by the ICS team in Lund for different profiles, containing the required software tools, libraries and configuration for the particular task. For developing EPICS applications on the I/O Controllers (IOC), the system includes the ESS EPICS Environment (EEE). This system is based on the concept of dynamical loading of EPICS libraries [18]. The EPICS application is composed by dynamically loading, linking and configuring pre-compiled modules at start-up. The system also tracks dependencies between modules so that the user just needs to define what modules are used by the application. The EEE consists of these modules plus a build system that can be used to create new modules.

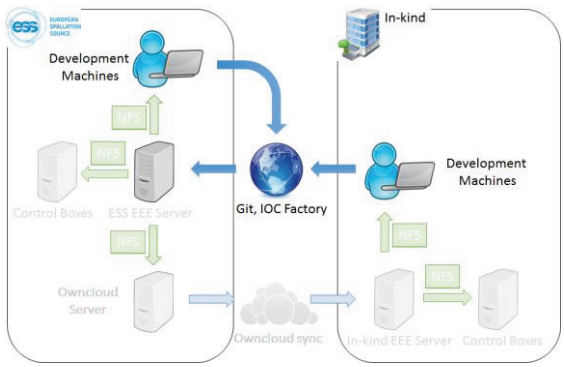


Figure 4: The ESS development environment.

The pre-compiled modules and configuration scripts for the EPICS I/O controllers reside on a network (NFS) drive, which is mounted by all IOCs. Thus each site needs its own NFS server. When this is set up, the ownCloud file sharing system [19] synchronises the remote servers with the central system.

The sources for the EEE software are kept in our git-based BitBucket [20] repository. A system that allows us to do systematic quality control of the software, consisting among others from a continuous build server (Jenkins) [21] etc., has been set up.

For setup and management of the EPICS IOCs we are developing a tool called IOC Factory. This is a web-based tool that enables setting up an EPICS IOC using information from the CCDB database and EPICS modules in the EEE. This way, an IOC setup consists only of configuration metadata; all logic will be contained in the EPICS modules. The first version of IOC Factory is

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

going to be released for production soon, however it will probably still need several refinements before it reaches the maturity for large scale deployment of systems.

All the services required for this development environment are web- or network-based. This makes it possible for us to keep the configuration data and the software modules under control at the Lund site without the need of replicating the data sources but still enable all developers to work using exactly the same set of tools.

In the developer toolset we also include tools from the EPICS community that have been selected based on their suitability to our purpose. For instance we are using the Archiver Appliance [22] for time series archiving of EPICS data. Control System Studio [23] has been selected as the toolbox to produce applications, first and foremost configuring GUI screens using BOY, BEAST for alarm management and so on. We also plan to invest in further development of tools in that framework.

We have been collaborating with the Spallation Neutron source SNS to build an accelerator online physics model based on OpenXAL [24]. In collaboration with SLAC we also have installed SLAC's EPICS 4-based set of services and applications and intend to use that as a basis for developing the accelerator physics applications for accelerator commissioning. For easier programmer access, interfaces to the Python language are provided for application writing. Python provides a powerful set of mathematical and utility libraries for scientific applications.

MACHINE OPERATION

The ultimate goal of the control system is to allow the ESS facility to be operated in a simple and efficient way. The characteristics of the machine set some constraints on how the machine is to be operated. For instance, one cannot start immediately with the highest available beam power but has to be able to smoothly go up from a low power mode to the full power operation. This has to happen so that machine protection is guaranteed at each step of the process. Details of this are still under discussion but the basic idea is to define a set of beam modes, namely beam power "envelopes" and conditions for operating within each envelope. A beam mode envelope is a combination of beam repetition rate, proton current and proton pulse length. Any of these parameters may be changed within the envelope, so that operation is still safe. Several conditions for machine safety have to be met before the beam mode can be changed. As one example, wire scanners can be used to measure beam profile only in modes where the beam power is low enough that it will not damage the wires. In higher power modes, operation of the wire scanners is not allowed.

The initial ideas for controlling the operation in these modes include distributing the mode and beam parameter information through the timing system and a system for redundantly cross-checking that the mode has been correctly propagated to the facility before allowing switching to a different beam mode. This requires a lot of coordination between the control system, machine protection and the accelerator components.

CONCLUSION

Although operation of the ESS is still some years ahead, we already need to support the development of the early systems in our in-kind partner laboratories. There is still a lot to do but a number of standards and structures have been set up to enable development.

Our strategy is to build a system using a combination of best practices and technologies from the community, new emerging technologies where they fit plus our own developments where required but as much in collaboration with the community and partners as possible.

ACKNOWLEDGMENT

We are grateful for the support of Greg White, SLAC, who helped us to set up the machine physics application framework based on EPICS V4. We also want to acknowledge the efforts of all our in-kind contributors, especially the team at CEA for their patience and help during the (still on-going) development of our services. This has given us an excellent chance to see how our ideas work in practice.

REFERENCES

[1] European Spallation Source ERIC website: http://europeanspallationsource.se
[2] European Spallation Source Technical Design Report, ESS 2013.
[3] R.Mudingay, "Preliminary Design of the Control Network and Related Infrastructure for ESS", THHD3O02, these proceedings, ICALEPCS 2015, Melbourne, Australia (2015).
[4] EPICS collaboration: http://www.aps.anl.gov/epics/
[5] EPICS 4 website: http://epics-pvdata.sourceforge.org
[6] G.White et al., "Recent Advancements and Deployments of EPICS Version 4", WEA3O02, these proceedings. ICALEPCS 2015, Melbourne, Australia (2015).
[7] Micro-Research Finland website: http://www.mrf.fi
[8] Timing System of the Swiss Light Source
[9] A. Nordt, "Development and Realisation of the ESS Machine Protection Concept", TUC3O03, these proceedings, ICALEPCS 2015, Melbourne, Australia (2015).
[10] Functional Safety of Electrical/ Electronic/ Programmable Electronic Safety-related Systems, IEC Standard 61508. International Electrotechnical Committee, Geneva, Switzerland.
[11] MicroTCA Enhancements for Rear I/O and Precision Timing (MTCA.4), PICMG 2011, http://www.picmg.org
[12] T.Korhonen et.al., "Modern Technology in Disguise, WECOCB05, ICALEPCS 2013, San Francisco, USA (2013).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

[13] EtherCAT Technology Group: https://www.ethercat.org/en/technology.html
[14] K.Rathsman, "ESS Naming Convention", THPEA038, Proc. IPAC 2013, Shanghai, China (2013).
[15] DISCS Collaboration: http://openepics.sourceforge.net/about-discs/
[16] Vagrant: https://www.vagrantup.com/
[17] Ansible: http://www.ansible.com/
[18] Dirk Zimoch: "Loadable Driver Modules", EPICS Meeting 2005, Archamps, France: http://epics.desy.de/epicsgroup/epics/content/e2/e3/e50/e81/FR-3-1-LoadableDriverModules.pdf
[19] ownCloud file synchronization and sharing: https://owncloud.org/
[20] Bitbucket (git) version management server: https://bitbucket.org/
[21] Jenkins continuous integration server: https://jenkins-ci.org/
[22] M.Shankar: New Archiver Appliance, EPICS Meeting 2013, San Francisco, USA (2013).
[23] Control System Studio software toolbox: http://controlsystemstudio.org/
[24] T. Pelaia, C.K. Allen, A. Shishlo, A. Zhukov, Y.C. Chao, et al.. "OPEN XAL Status Report 2015". 6th International Particle Accelerator Conference (IPAC2015), May 2015, Richmond, United States. Proceedings of IPAC2015, pp.1270-1272, (2015).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

List of Authors

Bold papercodes indicate primary authors

— A —

Abalo Miron, D. MOPGF115, WEB3O02
Abbott, M.G. **MOPGF097**, WEPGF089
Abeghyan, S. MOA3O02
Abeillé, G. **MOD3O02**, MOPGF150, WEA3O01
Abiven, Y.-M. MOPGF047, MOPGF098
Ackermann, S. MOC3O07
Acres, R. MOK03K01
Ács, P. **MOPGF050**
Adzic, P. MOPGF016
Afshar, N. **MOM312**
Agari, K. MOM309, **MOPGF123**
Aghababyan, A. MOA3O02
Aguilera-Padilla, C. WEPGF046, WEPGF047
Aiba, M. WEB3O04
Aishima, J. **WEPGF053**
Akeroyd, F.A. **MOPGF048**
Akiyama, A. MOB3O04, MOPGF141, WEPGF085
Al-Dmour, E. WEPGF148
Alarcon, A.D. FRA3O03
Alberts, M. **THHC3O01**
Alessio, F. MOPGF052
Allan, D.B. WED3O02, WEPGF043
Allison, S. WEPGF122
Althaus, A. MOPGF036
Altinbas, Z. MOPGF155
Alverson, S.C. **MOPGF032**, MOPGF037
Alves, D. MOPGF024
Amand, F. **WEPGF146**
Ambrosch, A. FRA3O03
Ameil, F. **MOPGF125**
An, S. **MOPGF001**, WEPGF034
Andersson, R. **MOPGF126**, MOPGF138, TUC3O03, FRB3O02
Andre, J.M. MOPGF025, TUA3O01, WEPGF013
Andreassen, O.Ø. **MOPGF115**, **WEPGF041**, WEPGF042, WEPGF129
Andreev, V. MOPGF149
Andronidis, A. MOPGF025, TUA3O01, WEPGF013
Androulakis, S. WEPGF059
Angelsen, C. MOPGF070
Anicic, D. FRA3O03
Antoine, A. MOPGF135
Antoniotti, F. MOPGF102, MOPGF103, MOPGF104, MOPGF112
Aoyama, T. MOB3O04, WEPGF085
Apollonio, A. TUC3O03
Aragao, D. WEPGF059, THHC3O04
Aragon, F. MOPGF104
Arkilic, A. WEA3O02, **WED3O02**, **WEPGF043**, **WEPGF044**
Arnold, N.D. THHC2O02
Arpaia, P. MOPGF171
Arruat, M. WEPGF127
Asano, K. MOB3O04
Ashton, A. WEB3O01, WEPGF053, THHB3O01
Augrandjean, F. MOPGF115
Augustinus, A. WEPGF018, WEPGF147, THHA3O01
Avila-Abellan, J.A. WEPGF081
Ayvazyan, V. **MOC3O07**, WEPGF029

— B —

Babel, S. **MOPGF002**
Bacchetti, M. MOPGF158
Bacher, R. **MOA3O02**, **WEM308**
Baek, S.Y. MOM306
Bär, R. WEPGF119
Bahnsen, F.H. MOPGF036
Bai, J.N. **WEPGF119**
Bailleux, S. TUB3O04
Baker, K.V.L. MOPGF048
Bakule, P. TUD3O02
Baldini, V. MOPGF163, **WEPGF001**
Balzer, A.F. **THHA3O02**
Bambery, K. WED3O01
Banerjee, A.S. WEA3O03, WEPGF025
Barbosa, J. MOPGF052
Bargalló, E. MOPGF126
Barillère, R. WEPGF094
Bark, A.P. MOPGF056
Baros, D. MOPGF161
Barros Marin, M. THHB2O02
Bartkiewicz, P.K. MOA3O02, WEPGF006
Bartolini, M. MOPGF110
Basham, M. THHB3O01
Batrakov, A.M. MOPGF080
Battam, N.W. WEPGF137
Baud, G. MOPGF099
Baudrenghien, P. WEPGF062
Bayer, M. MOPGF070
Bayramian, A.J. MOB3O01
Becciani, U. MOPGF163
Becheri, F. MOD3O04
Beck, D. WEPGF119
Beckmann, A. MOA3O02
Bednarek, M. MOPGF063
Behrens, U. MOPGF025, TUA3O01, WEPGF013
Bell, M. MOPGF048
Bell, P.J. **MOPGF049**
Bellato, M.A. MOPGF042, WEM307
Bellorini, F. MOPGF104, FRB3O02
Berryman, E.T. WEPGF113
Bertling, P.J. WEPGF059
Bertocco, M. MOPGF042
Bertrand, A.G. FRA3O03

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Bes, M. WEPGF094
Betz, C. MOPGF125
Beutner, B. MOPGF101
Billè, F. **WEPGF037**
Billich, H.R. WED3O06, WEPGF063
Binello, S. **WEPGF036**
Birch, S.L. FRB3O02
Birke, T. **MOC3O01**, **MOPGF077**
Bisou, J. MOPGF098, WEPGF056
Bitterli, K. FRA3O03
Björklund, E. MOPGF161, **THHC2O03**, **THD3O03**
Blache, F. MOPGF047, WEPGF056
Blanchard, S. **MOPGF102**, MOPGF103, MOPGF104
Blanco Alonso, J. WEPGF129
Blanco Vinuela, E. **MOC3O02**, MOPGF115, WEPGF091, WEPGF092, WEPGF094
Bland, A. **THHD3O08**
Bobnar, J. WEPGF133
Boccardi, A. **THHB2O02**
Bocchetta, C.J. MOPGF179
Boeckmann, T. MOA3O02
Bogani, A.I. WEPGF112
Boissinot, G. MOD3O02
Boivin, J-P. MOPGF104, MOPGF112
Bolkhovityanov, D. **WEPGF093**
Bond, C.S. WEPGF059
Bond, E.J. THHC3O05
Bond, P.M. WEPGF018, WEPGF147, THHA3O01
Bongers, R. THHB3O03
Booth, W. **WEPGF094**
Borghes, R. WEPGF037, **WEPGF038**
Bortolato, D. WEM307
Boukhelef, D. MOA3O02
Bourtembourg, R. WED3O04, **WEM310**
Bowers, G.A. MOD3O03
Boychenko, S. **WEPGF046**, WEPGF047
Boyes, M. FRB3O03
Bradu, B. MOC3O02, WEPGF094
Bräger, M. WEPGF141
Bräuning, H. MOPGF022, MOPGF153
Brands, H. WED3O06, FRA3O03
Branlard, J. MOC3O07, MOPGF079, WEPGF029
Branson, J. MOPGF025, TUA3O01, WEPGF013
Brederode, L.R. **MOA3O01**
Bree, M. WEM304
Briegel, C.I. **WEPGF002**
Brischetto, Y. MOPGF091
Brito, J.L.N. **WEPGF128**
Brockhauser, S. MOPGF050, MOPGF051, WEA3O01
Brodrick, D.P. FRB3O02
Broseta, M. MOPGF140, WEPGF081
Brown, G.W. MOPGF032
Brown, K.A. MOD3I01, MOM310, MOPGF129, WEPGF036, **WEPGF134**, **WEPGF135**
Brun, F. WEPGF037
Bruns, B. MOA3O02
Brunton, G.K. **MOD3O03**
Bucher, P. FRA3O03
Buckle, A.M. WEPGF059
Buffat, X. WEPGF066
Bulira, P. MOPGF179
Burdzanowski, L. **MOPGF006**
Bush, I.A. MOPGF048
Bussmann, K. MOPGF063
Buteau, A. MOD3O02, **MOPGF150**
Butkowski, Ł. MOPGF093, WEPGF029, **WEPGF074**
Butterworth, A.C. WEPGF062
Buttner, M. WEPGF155
Buttu, M. MOPGF110
Buyya, R. **FRK3K01**

— C —

Caillaud, T. TUD3I01
Calderone, G. WEPGF001
Calvo, J. **MOPGF045**
Camino, P. **WEC3O05**
Campell, M.L. MOPGF015
Caradoc-Davies, T. WEPGF059, THHC3O04
Carcassi, G. MOB3O02, **THHC3O02**
Cardoso, L.G. **MOPGF052**
Cardoso, M.B. THHB3O03
Carlier, E. MOPGF122, MOPGF135, WEPGF127
Carling, H. FRB3O02
Carmichael, L.R. **MOPGF145**, WEPGF061
Carrone, E. TUC3O07
Carwardine, J. THHC2O02
Casey, A.D. **THHC3O05**
Castro Carballo, M.E. MOM308
Caswell, T.A. WEPGF043, WEPGF044
Cautero, G. MOPGF070
Cecillon, F. THHB3O02
Celcer, T. FRA3O03
Cereijo García, J. FRB3O02
Cerenius, Y. MOB3O03
Cernaianu, M.O. WEA3O01
Cerniglia, P. THHB2O01
Chabot, D. WED3O02, WEPGF044
Chaize, J.M. WEA3O01
Chanavat, C. **WEPGF127**
Chang, P. C. Y. THHB3O01
Chang, Y.-T. WEPGF019, FRB3O01
Chapron, N. TUB3O04
Chareyre, V. MOPGF135
Charron, K. MOB3O01
Charrondière, C. WEPGF041
Charrue, P. **WEPGF155**
Chatzigeorgiou, N. MOPGF104, MOPGF112
Chaze, O. MOPGF025, TUA3O01, WEPGF013
Cheblakov, P.B. WEPGF093, **WEPGF095**
Chen, J. WEPGF019, FRB3O01
Chen, Z.C. WEPGF072
Chenda, V. WEPGF037, WEPGF038

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Cheng, W.X. MOC3O05, TUC3O05
Cheng, Y.-S. **MOPGF008**, MOPGF087, MOPGF088, **WEPGF019**, FRB3O01
Cherney, M.G. WEPGF136
Chevtsov, P. FRA3O03
Chilingaryan, S.A. WEPGF049
Chin, E.P. MOPGF029
Chitnis, P. WEPGF134, **MOD3I01**, **MOM310**, **MOPGF129**
Chiu, P.C. **MOPGF087**, MOPGF088, WEPGF019, FRB3O01
Cho, Y.-S. TUC3O06
Chochula, P.Ch. WEPGF018, WEPGF147, **THHA3O01**
Choi, H. J. **WEPGF115**
Chrin, J.T.M. **WEB3O04**, **WEPGF132**
Cichalewski, W. **MOPGF079**, WEPGF029
Cirami, R. MOPGF163, WEPGF001
Cirkovic, P. MOPGF016
Cittolin, S. MOPGF025, TUA3O01, WEPGF013
Claesson, N. FRB3O02
Clarke, M.J. MOPGF048
Clausen, M.R. MOA3O02
Clift, M. THHC3O04
Cobb, T.M. MOPGF056, MOPGF098, WEPGF137, THHB3O01
Cocho, C. THHB3O02
Coelho, E.P. MOPGF158
Conan, R. MOPGF071
Condamoor, S. **MOPGF014**
Conder, A.D. MOD3O03
Conrad, B.A. THHC3O05
Coppola, N. MOA3O02
Copy, B. **WEPGF042**, **THHD3O05**
Coretti, I. WEPGF001
Correa, J. MOPGF070
Corruble, D. MOPGF047
Corvetti, L. **MOO3O01**, **FRX3O02**, **FRX3O03**
Costa Lopez, X.B. WEPGF012
Costa, A. MOPGF163
Costanzo, M.R. MOPGF155, THHB2O01
Cousins, A.M. MOPGF098
Coutinho, T.M. WEA3O01, THHC3O03
Cowieson, N. THHC3O04
Crawford, D.J. MOPGF145
Crisanti, M. WEPGF083
Cristiani, S. WEPGF001
Cui, W. WEPGF021, THHD3O01
Cuk, G. MOPGF125
Cuní, G. **MOD3O04**, MOPGF140, MOPGF172, TUB3O02, WEPGF081, WEPGF148, THHC3O03
Curcio, R.F. MOPGF158
Curri, A. WEPGF037
Cusick, M.L. THHA3O03
Cyterski, C. **MOPGF029**, MOPGF037

— D —

D'Ewart, J.M. **MOPGF015**
D'Ottavio, T. MOPGF155, WEPGF036, WEPGF134, WEPGF135
da Graca, S. THHB3O01
Dabain, Y.S. MOPGF033
Dai, X.L. **WEC3O03**
Dale, D. MOPGF160
Dalesio, L.R. **MOB3O02**, MOC3O05, TUC3O05, WEA3O02, WED3O02, WEPGF043, WEPGF044
Daligault, F. MOPGF104
Damerau, H. MOPGF091
Daniluk, G. WEC3O01
Darlea, G.L. MOPGF025, TUA3O01, WEPGF013
Darvas, D. **WEPGF091**, **WEPGF092**
Das, D. MOPGF070
Davids, D. MOPGF120
Davidsaver, M.A. MOB3O02, WEA3O02, WEPGF030
de Almeida, H.D. MOPGF057
De Dios Fuente, A. WEPGF041
de Frutos, L. MOC3O02
De Long, J.H. MOB3O02, MOC3O05, TUC3O05, WEPGF080
De Maio, N. **MOPGF056**
de Prada, C. MOC3O02
de Villiers, C.C.A. **WEPGF118**
Dean, D. THHA3O03
Decker, F.-J. MOPGF032
Decker, G. THHC2O02
Decking, W. MOPGF101, TUD3O04
Deconinck, G. WEB3O05
Deghaye, S. WEPGF102
Delamare, Ch. MOPGF136
Deldicque, C. MOPGF025, TUA3O01, WEPGF013
Delfs, T. MOA3O02
Dent, A.J. THHB3O01
Di Calafiori, D.R.S. **MOPGF016**
Di Carlo, M. **WEPGF097**
Di Cosmo, M. **WEPGF005**
Di Giulio, L. MOPGF024
Di Marcantonio, P. WEPGF001
Di Nicola, J.-M.G. MOD3O03
Di Nicola, P. MOD3O03
Diamond, J.S. **WEPGF096**
Dickerson, C. MOPGF092
Ding, Y. MOPGF014
Dissertori, G. MOPGF016
Divall, E.J. FRA3O03
Djambazov, L. MOPGF016
Dobson, M. MOPGF025, TUA3O01, WEPGF013
Dolci, M. WEPGF097
Dolin'ek, I. WEPGF145
Donadio, M.P. **MOPGF057**
Dragu, M. WEPGF046, WEPGF047
Drochner, M. **MOPGF058**, MOPGF063

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Drouin, M.A. MOB3O01, TUD3O02, WEC3O05
Du, Q. WEPGF126
Du, Y.Y. MOPGF146
Duic, V. WEPGF037
Dupont, A.D. MOPGF025, TUA3O01, WEPGF013
Duval, P-Y. MOPGF052
Duval, P. MOA3O02, **WEPGF133**

— E —

Ebner, S.G. TUA3O02, **WED3O06**, FRA3O03
Ehm, F. WEPGF155
Eichin, M. TUC3O04
Elaazzouzi, A. THHB3O02
Elattaoui, X. MOD3O02
ElKamchi, N. **MOPGF131**
Emanov, F.A. WEPGF093
Ercolani, A. **TUK3K01**
Erhan, S. MOPGF025, TUA3O01, WEPGF013
Esenov, S.G. MOA3O02
Ezawa, K. MOPGF114, MOPGF160

— F —

Faatz, B. MOC3O07
Falaleev, V. MOPGF020
Falcón Torres, C.M. MOPGF172, TUB3O02, THHC3O03
Falkenstern, F. MOC3O01, MOPGF077
Fallejo, R.N. MOB3O01
Fantechi, R. MOPGF020
Fara, A. MOPGF110
Farkas, S. MOPGF051
Farnham, B. WEB3O02
Farrell, T.J. **WEPGF100**
Fatkin, G.A. **MOPGF080**
Faucett, J.A. MOPGF161
Favretto, D. WEPGF037
Fedorov, M.A. MOD3O03
Felzmann, C.U. WED3O01, **WEM303**, THHC3O04
Fernandes, B. MOA3O02
Fernandes, R.N. FRB3O02
Fernández Adiego, B. WEPGF092
Fernandez Carmona, P. **TUC3O04**
Fernandez, L. WEPGF146, FRB3O02
Fernández-Carreiras, D. MOD3O04, MOPGF140, MOPGF172, TUB3O02, WEPGF081, WEPGF148, THHC3O03, **FRX3O01**
Ferrand, T. WEPGF119
Ferreira, R. MOPGF102, MOPGF103, MOPGF104
Filik, J. THHB3O01
Filimonov, V. WEB3O02
Fisher, S. WEPGF053
Fishler, B.T. MOD3O03
Fitzek, J. MOPGF147
Fleischhauer-Fuss, L. MOPGF058, MOPGF063
Fleming, R. MOD3O03
Fleury, J. TUB3O04
Follath, R. THHA3O02
Fong, K. MOPGF160
Forsberg, J. MOPGF065, **WEM309**, WEPGF120
Fouquet, J.C. MOPGF150
Fraga, J. MOPGF104
Franco, J.G.R.S. MOPGF158
Friedrich, T. MOPGF019
Fröhlich, L. MOA3O02, **MOPGF101**, TUD3O04, **WEPGF006**
Fu, W. WEPGF135
Fuchsberger, K. MOPGF024
Fuica, S.A. WEPGF031
Fujii, Y. MOPGF030
Fujita, J. **WEPGF136**
Fujita, M. MOB3O04
Fukui, T. MOM305, WEPGF014
Fukunishi, N. WEPGF032
Fulkerson, E.S. MOB3O01
Fülöp, L.J. MOPGF050, **MOPGF051**, WEA3O01
Furseman, M.J. **WEPGF137**
Furukawa, K. MOB3O04, MOPGF035, WEC3O04, **THHC2O01**
Furukawa, Y. WEM305, WEPGF107

— G —

Gabourin, S. MOPGF135
Gabriel, M. **WEB3O03**
Gagey, B. MOPGF150
Gahl, T. **TUB3O03**
Gaio, G. **MOC3O03**
Galilée, M.A. WEPGF046, WEPGF047
Galindo, J. WEPGF062
Gallerani, L. **WEPGF045**
Gallese, B. FRB3O02
Gama, J. MOPGF104, MOPGF112
Garcia, F. MOPGF092
Garnett, R.W. MOPGF162
Garnier, J.C. WEPGF046, **WEPGF047**
Gasche, M. FRA3O03
Gąsior, M. MOPGF099
Gaspar, C. MOPGF052
Gatsi, T. WEPGF023
Gauron, P. MOPGF131
Gayadeen, S. **MOPGF175**, **MOPGF177**, **MOPGF178**
Gayet, Ph. MOPGF039, WEPGF042
Gelain, F. WEM307
Gende, J. TUB3O04
Gerhardt, W. MOA3O02
Gerring, M.W. **WEB3O01**, THHB3O01
Gessler, P. MOA3O02
Giacchini, M.G. WEM307
Giacuzzo, F. MOPGF113
Giambartolomei, G. MOA3O02
Gigi, D. MOPGF025, TUA3O01, WEPGF013
Gil, K.H. WEPGF115
Gilevich, S. MOPGF032
Gillingham, I.J. **MOPGF019**, WEPGF137

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Gindler, C. MOA3O02
Gioia, J.G. MOPGF161
Girone, M. MOPGF171
Giuressi, D. MOPGF070
Glege, F. MOPGF025, **TUA3O01**, WEPGF013
Glover, C. MOM312
Gnadt, P. MOPGF070
Gobbo, A. FRA3O03
Godineau, G. MOPGF136
Göttlicher, P. MOPGF070
Golonka, P. **MOPGF020**, **MOPGF021**, MOPGF039, **WEPGF010**, WEPGF069
Gomes, P. MOPGF102, MOPGF103, **MOPGF104**, MOPGF112
Gomez De La Cruz, M.F. MOPGF115
Gomez-Ceballos, G. MOPGF025, TUA3O01, WEPGF013
Gong, G.H. MOPGF134, WEPGF090, WEPGF126, **THHB2O03**
Gong, H. THHB2O03
Gonzalez-Berges, M. MOPGF020, MOPGF021, MOPGF039, WEPGF068, WEPGF069
Goralczyk, L. MOPGF039
Gorbachev, E.V. **MOPGF149**
Gorbonosov, R. WEB3O03
Goryl, P.P. **MOPGF179**, WEPGF145
Goscinski, W.J. WEPGF059, **WED3O01**
Götz, A. **WEA3O01**, WEPGF063
Gou, S.Z. WEPGF034
Graafsma, H. MOPGF070
Graillot, H. TUB3O04
Grecki, M.K. MOC3O07
Greer, A. TUD3O02
Groenewegen, D. WEPGF059
Grossi, D.R. MOC3O04
Grossmann, M. TUC3O04
Grzegrzólka, M. MOPGF093
Gu, K. MOPGF001, WEPGF034
Gu, Q. MOPGF070
Gu, X. MOPGF155
Guerrini, N. MOPGF070
Guo, Y.H. **WEPGF011**, WEPGF021
Gupta, Y. **WEK3K01**
Gutierrez, A. MOPGF102, MOPGF103, MOPGF104
Guyotte, G.S. WEPGF105
Guzman, J.C. **THD3O05**
Gysin, S.R. FRB3O02

— H —

Ha, K. MOC3O05, **TUC3O05**, WEPGF080
Ha, K.M. MOB3O02
Haas, D. **WEPGF049**
Haefner, C. MOB3O01, WEC3O05
Haemmerli, F. FRA3O03
Hager, M. **WEPGF101**
Hakulinen, T. **MOPGF132**, MOPGF143, **WEPGF012**
Hall, C.J. MOK03K01, WED3O01
Hall-Wilton, R.J. TUB3O03
Hamada, Y. WEM305
Hammouti, L. MOPGF143
Hanyecz, V. MOPGF050, MOPGF051
Hardion, V.H. **MOB3O03**, MOPGF049, MOPGF179, WEM309, WEPGF120
Hartman, S.M. **MOM303**, WEA3O02
Harvey, M. MOPGF155
Hasegawa, T. WEPGF014
Haseitl, R. **MOPGF022**, MOPGF153
Hatje, J. MOA3O02
Hauf, S. MOA3O02, WEPGF050
Hauser, N. TUB3O01, WEPGF071
Hausermann, D. MOK03K01
Havart, F. MOPGF132
He, J. MOPGF146
He, Y. WEPGF021
He, Z. WEPGF011, **WEPGF021**, **THHD3O01**
He, Z.Q. WEPGF113
Hegeman, J. MOPGF025, TUA3O01, WEPGF013
Heisen, B.C. MOA3O02, WEPGF050, WEPGF153
Hennies, F. H. MOB3O03
Hensler, O. MOA3O02, MOC3O07, MOPGF101, TUD3O04, WEPGF029
Heron, M.T. MOPGF098, MOPGF175, **MOPGF176**, MOPGF177, MOPGF178, WEPGF137
Heuer, M. WEPGF015
Hickin, D.G. WEA3O02
Hierholzer, M. TUD3O05, WEPGF015
Higgs, C.E. FRA3O03
Hilbes, C. MOPGF138, TUC3O03
Hill, J.O. MOPGF161, **THHA2O02**
Himmel, B. TUD3O02
Hines, C. WED3O01, WEPGF059
Hirose, M. MOB3O04
Hisazumi, K. MOPGF035
Hobbs, N. WEM303
Höfle, W. WEPGF062
Hofer, J. MOPGF021
Hoffmann, T. MOPGF022, **MOPGF153**
Hoguin, F. **WEPGF102**
Holme, O. TUA3O01, MOPGF016, MOPGF025, MOPGF120, WEPGF013
Holzner, A. MOPGF025, TUA3O01, WEPGF013
Hoobler, S. L. **MOPGF037**, **MOPGF038**
Horáček, J. TUD3O02
Horáček, M. TUD3O02
Hosoda, N. **MOM305**
Hovel, T. FRA3O03
Howells, G.D. MOPGF048
Hoyle, S.A. **WEM301**
Hsu, K.T. MOPGF008, MOPGF087, MOPGF088, WEPGF019, FRB3O01
Hsu, S.Y. FRB3O01
Hu, J. MOPGF134
Hu, K.H. MOPGF087, MOPGF088, FRB3O01

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Hu, L.B. **MOPGF154**
Hu, Y. TUC3O05
Huang, C. H. MOPGF008, **MOPGF088**, WEPGF019, FRB3O01
Huang, Z. **WEPGF020**
Hubatka, M. MOPGF171
Hüther, H.C. **MOPGF147**
Hui, H. MOPGF160
Hulsart, R.L. THHB2O01
Humar, T. WEPGF145, FRA3O03
Hutton, M.S. THHC3O05
Hyun, H.J. MOPGF070

— I —

Ibsen, J.P.A. MOM311, WEPGF031
Ichikawa, T. MOPGF035
Igarashi, S. MOPGF117
Iida, N. THHC2O01
Iitsuka, Y. MOB3O04, WEC3O04
Ikeda, H. WEPGF052
Ikegami, M. WEPGF113
Ilinski, P. TUC3O05
Ilyin, I.V. MOPGF080
Inagaki, T. WEPGF014
Ingallinera, A. MOPGF163
Iranmanesh, R. MOPGF160
Ishii, M. MOM305, **WEPGF014**
Ismail, A. MOPGF033
Isocrate, R. MOPGF042, WEM307
Iwasaki, M. **MOB3O04**, MOPGF141, WEC3O04, WEPGF085
Iwasaki, Y. **MOPGF023**

— J —

Jackson, S. **MOPGF024**, MOPGF099
Jäger, J.M. MOA3O02, MOM308
Jamilkowski, J.P. MOM302, MOPGF066, **MOPGF155**, THHB2O01
Jamróz, J.J. MOPGF179, **WEPGF120**
Janaschke, S. MOPGF063
Jang, H. MOPGF119, WEPGF124
Janousch, M. **FRA3O03**
Janser, G. FRA3O03
Janulis, M. MOPGF025, TUA3O01, WEPGF013
Januschek, F. WEPGF050
Jarosz, D.P. THHC2O02
Jenninger, B. MOPGF102, MOPGF103
Jensen, L.K. MOPGF099
Jiang, X.S. MOPGF134
Jiménez Estupiñán, R.J. **MOPGF025**, TUA3O01, WEPGF013
Jiménez-Rey, D. MOPGF045
Jin, D.P. **TUC3O02**
Johansen, E. TUC3O04
Johnson, A.N. WEA3O02
Johnson, G.W. MOB3O01, WEC3O05
Jong, L.M. **THHC3O04**
Joubert, F. THHC3O01
Jovanovic, D. MOPGF016
Jover-Mañas, G. MOPGF172
Jud, G. FRA3O03
Junkes, H. **MOPGF026**
Jurcso, P. WEPGF155
Juvan, V. **WEPGF145**

— K —

Kadyrov, R.A. **WEPGF080**
Kago, M. WED3O03, WEPGF014
Kaji, H. MOB3O04, **WEC3O04**, WEPGF085
Kalantar, D.H. MOD3O03
Kalantari, B. FRA3O03
Kamarainen, H.T.T. WEPGF010
Kamikubota, N. MOPGF117, **WEPGF121**
Kamitani, T. THHC2O01
Kammering, R. MOA3O02, MOPGF101, **TUD3O04**, WEPGF150
Kaneyasu, T. MOPGF023
Kang, H.-S. MOM306, WEPGF115
Kankiya, P. K. **MOM302**, **MOPGF066**, MOPGF155, **WEC3O06**
Kannan, A. WEPGF059
Kapeller, R. FRA3O03
Kappen, P. MOM312
Karabekyan, S. MOA3O02, **MOD3O05**
Karnaev, S.E. WEPGF095
Karstensen, S. MOA3O02, **MOM308**
Kasemir, K.-U. **TUA3O04**, WEA3O02, **WEPGF105**
Kasl, K. MOB3O01, TUD3O02
Kato, Y. WEPGF052
Katz, R.A. WEPGF036
Kavarskas, J. MOPGF160
Kawane, Y. MOPGF137
Kazama, S. THHC2O01
Keitel, R. MOPGF114
Keller, H. MOA3O02
Kester, O.K. WEPGF119
Keuler, S. MOPGF063
Keymer, D.P. MOPGF048
Kheffafa, C.K. MOPGF047
Khokhriakov, I.A. WEA3O01
Khromova, A. MOPGF070
Khudayberdieva, O.A. WEPGF095
Kikuzawa, N. MOPGF137, **WEPGF052**, WEPGF121
Killenberg, M. **TUD3O05**, **WEPGF015**, WEPGF029
Kim, C. **MOM306**
Kim, D. MOB3O01
Kim, D.I. TUC3O06
Kim, H.S. TUC3O06
Kim, J.H. MOM306
Kim, K.H. **WEPGF122**
Kim, K.S. MOPGF070
Kim, K.W. MOM306
Kim, S.H. WEPGF115

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

King, Q. MOPGF090, WEC3O02, **WEPGF106**
Kirchner, D. WEPGF066
Kirichenko, A. MOPGF149
Kirstein, O. TUB3O03
Kiss, M. MOPGF050, MOPGF051
Kjellsson, L. MOPGF065
Kleines, H. MOPGF058, **MOPGF063**
Knap, G. WEPGF137
Knowles, K.J. MOPGF048
Ko, I.S. MOM306
Kobayashi, T. WEC3O04
Kocharyan, V. MOA3O02
Koda, S. MOPGF023
Koh, E.S. MOB3O01
Kolad, B. MOPGF024
Komel, M. MOPGF105
Komiyama, M. WEPGF032
Koncz, Cs. MOPGF050
Koncz, M.T. MOPGF051
Konrad, M.G. WEPGF030
Kopmann, A. WEPGF049
Kopylov, L. MOPGF102, MOPGF104
Korepanov, A.A. MOPGF111
Korhonen, T. TUB3O03, TUC3O03, WEA3O02, **FRB3O02**
Korth, O. MOA3O02
Kourousias, G. WEPGF037, WEPGF038
Koza, M. WEPGF046, WEPGF047
Kozak, T. WEPGF074, WEPGF015
Koziol, Q. WEPGF063
Kraimer, M.R. WEA3O02
Krakówski, P. MOPGF104, MOPGF112
Kreider, M. THHA2O03
Krejcik, P. MOPGF014, MOPGF015
Krempaská, R.A. FRA3O03
Krol, K.H. WEPGF046, WEPGF047
Kudou, T. MOPGF035, WEC3O04
Kumar, M. MOA3O02
Kuntzsch, M. MOPGF093
Kuo, C.H. FRB3O01
Kurepin, A.N. WEPGF018, THHA3O01, WEPGF147
Kusano, S. MOPGF035, WEC3O04
Kuster, M. WEPGF050
Kwon, H.-J. TUC3O06

— L —

Labrenz, M. WEPGF042
Labudda, A. MOA3O02
Laface, E. FRB3O02
Lai, L.W. WEPGF072
Lam, B. MOPGF002
Lang, K. MOPGF022
Lang, P.M. WEPGF050
Lange, R. WEA3O02
Lange, S. MOPGF070
Langlois, F. WEPGF056
Lanning, R.K. MOB3O01
Larrieu, T. L. MOPGF072
Larsen, R.S. MOPGF038
Larsson, K. MOB3O03
Laster, J.S. WEPGF036
Lau, G. MOD3O03
Lauk, D.J. FRA3O03
Lay, S.C. MOPGF019
Laznovsky, M.P. FRA3O03
Le Goc, Y. THHB3O02
le Roux, G.M. MOD3O06, WEPGF097
Lê, S. MOD3O02, WEPGF056
Lebioda, K.T. WEPGF106
Lechman, M. **WEPGF018**, WEPGF147, THHA3O01
Leclercq, N. **WEPGF056**
Ledeul, A. **TUD3O03**
Lee, D. MOPGF008, FRB3O01
Lee, S. MOPGF119, **WEPGF124**
Leng, Y.B. WEPGF072
Levchenko, N. FRB3O02
Levens, T.E. WEPGF062, THHB2O02
Levik, K. WEPGF053
Levine, W.S. MOC3O05
Lewis, W.K. MOB3O02, WED3O02, WEPGF043, WEPGF044
Li, C. WEPGF020, WEPGF117
Li, H. WEPGF090, **WEPGF126**, THHB2O03
Li, J.M. WEPGF090
Li, J.Y. WEPGF117
Li, L.F. WEPGF070, WEPGF030
Li, P. WEPGF034
Li, W. WEPGF117
Liao, C.Y. MOPGF008, MOPGF087, MOPGF088, WEPGF019, **FRB3O01**
Lidón-Simon, J. MOB3O03, MOPGF179
Limberg, T. MOA3O02, MOPGF101, TUD3O04
Lin, L. MOD3O05
Lindberg, M. **MOPGF065**, MOPGF179
Lipinski, M.M. WEC3O01
Lisboa, R.P. MOPGF158
Liu, D. WEPGF113
Liu, F. **MOPGF134**, MOPGF146
Liu, G. WEPGF020, WEPGF117
Liu, H.T. WEPGF011
Liu, M. WEC3O03, WEC3O04
Liu, N. MOPGF145
Liu, T. WEPGF011
Liu, X.J. MOPGF001
Liu, Y.T. MOD3O05
Livingstone, J. MOK03K01
Locatelli, J. THHB3O02
Lomperski, M. WEPGF133
Lonsing, R. MOPGF153
Lonza, M. MOC3O03, WEPGF037, WEPGF038, WEPGF152
Loos, H. MOPGF014, MOPGF015, MOPGF038

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Luchini, K. MOPGF002, MOPGF015
Lüders, S. **THD3O06**
Luo, J.B. WEPGF011
Luo, Y. WEPGF021, THHD3O01
Lurkin, N. MOPGF020
Luscher, C. FRA3O03
Lustermann, W. MOPGF016
Lutz, H. FRA3O03

— M —

Mader, J.A. MOPGF040
Märki, F. FRA3O03
Maesaka, H. WEPGF014
Magnin, N. **MOPGF135**, WEPGF127
Magrans de Abril, M. MOPGF090, **WEC3O02**, WEPGF106
Maia, L.G. MOA3O02
Maier-Manojlovic, D. **MOPGF027**, FRA3O03
Majzik, I. WEPGF091
Makowski, F. WEPGF029
Maksimenko, A. **MOK03K01**, WED3O01
Malitsky, N. MOB3O02, WEA3O02
Mannicke, D. **TUB3O01**, WEPGF071
Mansouri Sharifabad, M. FRB3O02
Marais, N. **MOPGF067**
Marassi, A. MOA3O01, MOD3O06, MOPGF163
Marcato, D. WEM307
Marchal, J. MOPGF070
Marquarding, M. **FRA3O01**
Marques, S.R. WEPGF128
Marras, A. MOPGF070
Marroquin, P.S. MOPGF161
Marsching, S. TUD3O05, WEPGF015
Marsh, B. MOPGF070
Marshall, C.D. **MOB3O01**
Martel, P. **MOPGF136**
Marti Martinez, R. MOC3O02
Martin, K.S. WEPGF096
Martin, P. MOM312
Martini, R. **WEPGF141**
Martino, M. WEPGF106
Martins Ribeiro, T. WEPGF047
Martins, B.S. WEA3O02
Martins, J.P.S. **MOPGF158**
Martins, L.A. WEPGF128
Martos, V. WEPGF120
Maruta, T. MOPGF137
Maruyama, T. WEPGF014
Masetti, L. MOPGF025, TUA3O01, **WEPGF013**
Maslov, P.A. **MOPGF105**
Mateo, F. MOPGF102, **MOPGF103**
Mathisen, D.G. MOD3O03
Matilla, O. **WEPGF081**
Matsumoto, T. **WEM305**, **WEPGF107**
Matsushita, T. WEM305
Maxwell, D.G. WEPGF113
Maxwell, T.J. MOPGF014
Mayer, C. TUD3O02
Mayor, A. TUC3O04
Mazaeda, R. MOC3O02
Mazanec, T. MOB3O01, TUD3O02
McGowan, S. WEPGF059
McIntosh, P. WED3O01
Mead, J. TUC3O05
Medjoubi, K. WEPGF056
Mégevand, D. WEPGF001
Mehle, M. WEPGF015
Meijers, F. MOPGF025, TUA3O01, WEPGF013
Menk, R.H. MOPGF070
Mercado, R. MOPGF019
Merker, S. MOPGF102, MOPGF104
Mertens, K.-H. MOPGF063
Meschi, E. MOPGF025, TUA3O01, WEPGF013
Mexner, W. WEPGF049
Meyer, G.R. **WEPGF059**
Meykopff, S.M. MOA3O02, MOPGF101, TUD3O04, **WEPGF142**
Michel, V. MOPGF049
Michnoff, R.J. MOPGF155, **THHB2O01**
Miedzik, P.B. MOPGF153
Migoni, C. MOPGF110
Mikawa, K. MOPGF035
Mikheev, M.S. MOPGF104
Milan, A.M. MOPGF065
Milcent, H. WEPGF042, THHD3O05
Miller Kamm, V.J. MOD3O03
Miller, T.A. MOPGF155, WEC3O06
Milosic, T. MOPGF022, MOPGF153
Minolli, S.M. MOPGF047
Mirtschin, P.L. WEM301
Mita, C. THHB3O01
Miura, A. **MOPGF137**
Miura, T. THHC2O01
Miyahara, F. MOPGF035, WEC3O04, THHC2O01
Miyao, T. MOPGF137
Mizukawa, Y. MOPGF035
Mocuta, C. WEPGF056
Möller, M. MOA3O02
Mohácsi, Á. MOPGF051
Mokone, O.J. **WEPGF023**
Molendijk, J.C. WEPGF062
Molina Marinas, E. MOPGF045
Mollá, J. MOPGF045
Mommsen, R.K. MOPGF025, TUA3O01, WEPGF013
Monakhov, D.V. MOPGF149
Monard, H. MOPGF131
Monera Martinez, A. MOPGF126, **MOPGF138**, TUC3O03, FRB3O02
Monnier-Bourdin, D. WEC3O05
Monteiro, P. MOPGF098
Montis, M. WEM307
Moreno, G.B.Z.L. **THHB3O03**
Moreton-Smith, C. MOPGF048

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Morino, Y. **MOM309**, MOPGF123
Moriyama, K. **WEPGF060**
Morovic, S. MOPGF025, TUA3O01, WEPGF013
Morris, D.B. MOPGF160
Morris, J. WEPGF036, WEPGF134
Mudie, N. WEPGF059, THHC3O04
Mudingay, R. FRB3O02
Müller, R. MOC3O01, MOPGF077
Müller, R. MOPGF147
Münnich, A. **WEPGF050**
Mun, G. MOM306
Muralikrishna, G. WEA3O03
Murillo-Garcia, R. **MOPGF090**, WEC3O02, WEPGF106
Murphy, J.M. TUC3O07
Mutti, P. **WEPGF083**, **WEPGF084**, **THHB3O02**

— N —

Naito, T. MOB3O04, MOPGF141
Nakamura, T. MOB3O04, WEPGF085
Nakamura, T.T. MOB3O04, MOPGF141, WEC3O04, **WEPGF085**
Nakanishi, Y. MOPGF142
Nakatani, T. WEPGF060
Nakayoshi, K. **MOPGF030**
Nallin, P.H. MOPGF158
Napieralski, A. MOPGF079
Naylon, J. MOB3O01, **TUD3O02**, WEC3O05
Nelissen, K. MOPGF051
Nelson, R.F. MOPGF048
Nemesure, S. MOPGF155, WEPGF134, WEPGF135
Neswold, R. MOPGF145, WEPGF002, WEPGF061
Nicholls, T.C. MOPGF070
Nicklaus, D.J. **WEPGF061**
Nicoletti, A. WEPGF106
Nicoloso, J. TUB3O04, **FRA3O02**
Nicotra, G. MOPGF163
Niedziela, J. WEPGF147
Nikiel, P.P. WEB3O02
Ninin, P. MOPGF132, MOPGF143, WEPGF012
Nissen, J.D. MOB3O01
Nocita, C. MOPGF163
Nordt, A. MOPGF126, MOPGF138, **TUC3O03**, FRB3O02
Noulibos, R. MOPGF121
Nunes, R. MOPGF136
Nunez-Barranco-Fernandez, C. MOPGF025, TUA3O01, WEPGF013
Nussbaumer, R.B. MOPGF114, **MOPGF160**
Nutter, B.J. **THD3O02**

— O —

O'Dell, V. MOPGF025, TUA3O01, WEPGF013
O'Dowd, F.P. THHB3O03
Oberson, D. MOPGF091
Ockards, M.T. WEPGF065
Odagiri, J.-I. MOB3O04, WEPGF085
Ohnishi, Y. THHC2O01
Ohshima, T. MOM305, WEPGF014
Ojeda Sandonís, M. **WEPGF062**
Okada, K. WEPGF107
Olexa, J. MOPGF099
Olsen, J.J. MOPGF002, MOPGF014, MOPGF038
Omet, M. WEPGF029
Ondreka, D. WEPGF119
Oram, D.E. MOPGF048
Orlandi, R. WEPGF046, WEPGF047
Orlati, A. **MOPGF110**
Orsini, L. MOPGF025, TUA3O01, WEPGF013
Ortiz, H. THHB3O02
Oser, P. WEPGF012
Otake, Y. WEPGF014
Ovando, A. WEPGF031

— P —

Pache, J. MOPGF039
Pacheu, V. MOD3O03
Padrazo, D. TUC3O05
Page, R.F. MOPGF020
Page, S.T. THHD3O08
Pal, T. FRA3O03
Palaha, A.S. **MOPGF070**
Palmer, C.M. THHB3O01
Panepucci, E.H. **TUA3O02**, WED3O06
Panjikar, S. WED3O01
Panov, A. **MOPGF111**
Parenti, A. MOA3O02
Park, B.R. MOM306
Pascual-Izarra, C. MOPGF172, TUB3O02, WEPGF148, **THHC3O03**
Passuello, R. WEPGF112
Pastor Ortiz, R. MOPGF172
Paterson, D. WED3O01
Patsouli, A. MOPGF135
Patwari, P. **WEA3O03**, WEPGF025
Paul, J.D. MOPGF161
Paul, M. MOD3O03
Paulic, D. FRB3O02
Paus, C. MOPGF025, TUA3O01, WEPGF013
Pavinato, S. WEM307
Pavleski, M. MOPGF105, WEPGF145, WEPGF146
Pearson, J. MOK03K01
Pearson, M.R. TUA3O04, WEPGF105
Pedersen, E. MOPGF024
Pedersen, U.K. MOPGF070, THHB3O01
Pedretti, D. MOPGF042, **WEM307**
Peele, A. **MOO3O02**
Pekrul, W.E. THHB2O01
Pelliccia, D. MOK03K01
Penning, J. MOA3O02
Pereira, H.F. MOPGF102, MOPGF103, MOPGF104, MOPGF112
Perez, S. **TUD3I01**
Perez, S. WEPGF134

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Perrelet, D. **MOPGF091**
Persson, A.G. MOPGF179
Pessemier, W. **WEB3O05**
Peters, C.E. **MOPGF092**
Petersen, D.E. MOB3O01
Petit, A. MOPGF153
Petrosyan, A. MOA3O02
Petrosyan, G. MOA3O02
Petrosyan, L.P. MOA3O02, WEPGF015
Petrosyan, V. MOA3O02, WEPGF028
Petrucci, A. MOPGF025, TUA3O01, WEPGF013
Pezzetti, M. **MOPGF171**
Pfeiffer, S. MOC3O07, WEPGF029
Pflüger, J. MOD3O05
Piacentino, J. WEPGF036
Piccinelli, F. WEPGF042
Pieck, M. **MOPGF161**, **MOPGF162**
Pieri, M. MOPGF025, TUA3O01, WEPGF013
Pigny, G. MOPGF104, **MOPGF112**
Pinaroli, G. MOPGF070
Pinazza, O. WEPGF018, THHA3O01, **WEPGF147**
Piotrowski, A. MOC3O07, WEPGF015, WEPGF029
Piso, D.P. FRB3O02
Piton, J.R. MOPGF057
Pivetta, L. **MOPGF113**, **WED3O04**, **WEPGF112**, WEPGF152
Plaz, M. WEPGF083, WEPGF084
Pleško, M. WEPGF146
Podadera, I. MOPGF045
Poeschl, M.C. WEPGF046, WEPGF047
Poirier, S. WEPGF056
Pon, J.J. **MOPGF114**, MOPGF160
Ponchia, R. MOPGF042
Pons, J.L. WEA3O01, WED3O04, WEM310
Poppi, S. MOPGF110
Portmann, W. FRA3O03
Pototzki, P. MOA3O02
Pourmal, E. WEPGF063
Power, M.A. MOPGF092
Prados, C. WEPGF119
Prędki, P. WEPGF015, WEPGF074
Prica, M. WEPGF037, WEPGF038
Price, I.A. **MOPGF071**
Prieto, P.P. MOPGF104
Przygoda, K.P. WEPGF029
Pugliese, R. WEPGF037

— Q —

Quenette, S.M. WEPGF059
Quilichini, M. WEPGF094
Quinn, P.D. THHB3O01

— R —

Raab, N. WEPGF050
Racz, A. MOPGF025, TUA3O01, WEPGF013
Rah, S.Y. MOPGF070
Rapaz, S. MOPGF160
Raskin, G. WEB3O05
Rathsman, K. FRB3O02
Rawat, A. WEB3O04
Redaelli, S. MOPGF099, WEPGF066
Reed, R.K. MOD3O03
Rees, N.P. MOPGF070, **WEPGF063**, THHB3O01, **THD3O01**
Rees, S.G. FRA3O03
Regele, H.A. TUC3O04
Regodic, M. WEPGF101
Rehlich, K.R. MOA3O02, TUD3O04, WEPGF150
Rehm, G. MOPGF097, MOPGF175, MOPGF177, MOPGF178
Reisdorf, P.D. THHC3O05
Reisdorf, S.M. THHC3O05
Reiter, A. MOPGF153
Renaud, G. MOPGF098
Reščič, M. FRB3O02
Ressmann, D. WEPGF049
Reszela, Z. MOD3O04, **MOPGF172**, **TUB3O02**, THHC3O03
Rettig-Labusga, S. MOA3O02
Řeža, S. MOPGF070
Ribeiro, T.M. WEPGF046
Richards, J.E. MOPGF114, MOPGF160
Richter, T.S. TUB3O03
Rickens, H.R. MOA3O02
Rigby, J. WEPGF059
Riggi, S. MOA3O01, **MOPGF163**
Righini, S. MOPGF110
Rijllart, A. MOPGF115, **WEPGF129**
Rio, B. MOPGF104
Rissi, M. WEPGF063
Ritzert, M. **MOPGF164**, **WEPGF024**
Robertazzi, T.G. MOPGF129
Roberts, P. MOPGF025, TUA3O01, WEPGF013
Roderick, C. MOPGF006
Rodrigues, A.R.D. MOPGF158
Rogers, W.A.H. WEPGF137
Rogind, D. **FRB3O03**
Rojec, U. FRB3O02
Roldán, D. WEPGF148
Rollins, J.G. **MOA3O03**
Romanov, S. MOPGF149
Rosanes Siscart, M. MOPGF172, TUB3O02, THHC3O03
Rose, A.J. MOPGF176
Roshchin, M. WEPGF068
Rouse, J. MOD3O03
Rowe, M. MOPGF114, MOPGF160
Roy Chaudhuri, S. MOD3O06, WEA3O03, **WEPGF025**
Rubio, A. MOPGF140
Rubio-Manrique, S. MOD3O04, **MOPGF140**, MOPGF172, **WEPGF148**
Rüter, T. WEPGF050
Ruiz-Martinez, E. WEPGF083, WEPGF084

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Rukoyatkina, T.V. MOPGF149
Rus, B. MOB3O01, TUD3O02, WEC3O05
Rutkowski, I. **MOPGF093**
Ruz, A. WEPGF081
Ryan, C.G. WED3O01
Rybaniec, R. WEPGF074
Rybnikov, V. TUD3O04, **WEPGF028**

— S —

Sachinelli, L.D.S. MOPGF158
Saey, P. WEB3O05
Saez, N. MOM311, WEPGF031
Sakashita, K. MOPGF030
Sakulin, H. MOPGF025, TUA3O01, WEPGF013
Sakurai, T. MOM305
Sala, L. WED3O06
Salabert, J. WEPGF081
Saleh, I. **MOPGF033**
Salmon, J.L. WEPGF141
Samms, T. MOPGF066
Sanchez, R.J. MOD3O03
Sandström, A. TUB3O03
Santin, P. WEPGF001
Saotome, H.S. MOPGF035
Sasaki, S. MOB3O04, **MOPGF141**, WEPGF085
Sass, R.C. MOPGF037
Såthe, C. MOPGF065
Sato, K.C. **MOPGF117**
Sato, Y. MOM309, MOPGF123
Satoh, M. **MOPGF035**, WEC3O04, THHC2O01
Scafuri, C. WED3O04
Scalamera, G. MOPGF113, WED3O04
Scarcia, M. WEPGF037, WEPGF038
Scarisoreanu, A.M. MOPGF174
Scarlat, F. **MOPGF174**
Scarlat, Fl. MOPGF174
Schälicke, A. MOC3O01, MOPGF077
Schaller, A. MOPGF147
Schierle, E. THHA3O02
Schillirò, F. MOPGF163
Schirmer, D. **MOPGF036**
Schlarb, H. MOC3O07, WEPGF029
Schlenker, S. **WEB3O02**
Schlesselmann, G. MOA3O02
Schmidt, Ch. MOC3O07, MOPGF079, MOPGF093, TUD3O05, WEPGF015, **WEPGF029**
Schmidt, R. **TUC3I01**, TUC3O03
Schöllkopf, W. MOPGF026
Schoeneburg, B. MOA3O02
Scholz, M. MOC3O07
Schreiber, S. MOC3O07
Schrettner, L. MOPGF050, MOPGF051
Schwarz, N. THHC2O02
Schwemmer, R. MOPGF052
Schwick, C. MOPGF025, TUA3O01, WEPGF013
Sedgwick, I. MOPGF070
Sedillo, J.D. MOPGF161
Sedykh, G.S. MOPGF149
Segura, G. TUD3O03
Seimiya, Y. MOPGF035
Sekoranja, M. WEA3O02
Seletskiy, S. TUC3O05
Seo, K.W. WEPGF115
Seol, K.T. TUC3O06
Serra-Gallifa, X. WEPGF081
Serrano, J. WEC3O01, THHA2I01, **THD3O07**
Shankar, M.V. WEA3O02, **WEPGF030**
Shaw, M.J. MOD3O03
Sheehy, B. WEC3O06
Shehzad, N. WEPGF015
Shelley, F.E. MOPGF161, MOPGF162
Shen, G. MOB3O02, TUC3O05, WEA3O02, **WEPGF113**
Shen, T.C. MOM311, **WEPGF031**
Shepherd, E.L. MOM312
Shevyakov, I. MOPGF070
Shoaee, H. FRB3O03
Shortridge, K. WEPGF100
Shrey, T.C. MOPGF155
Shroff, K. MOB3O02, THHC3O02
Silenzi, A. MOA3O02
Silva, M. E. MOPGF158
Silvola, R.P.I. TUD3O03
Singh, O. TUC3O05
Sjöblom, P. MOPGF065
Sjöström, M. MOB3O03
Škoda, P. TUD3O02
Slabber, M.J. **WEPGF065**, **THHD3O06**
Slepicka, H.H. THHB3O03
Sliczniak, M.Z. WEPGF002
Slominski, R.J. **MOPGF072**
Smareglia, R. MOD3O06, WEA3O01, WEPGF097
Smith, B.G. MOPGF162
Smith, R.M. TUC3O05
Smith, S.R. MOPGF038
Smoljanin, S. MOPGF070
So, S. WEPGF080
Soare, C.-V. WEB3O02
Söderqvist, A.A. FRB3O02
Sollander, P. MOPGF039
Sombrowski, E. MOA3O02, **WEPGF150**
Son, C.W. MOPGF119, WEPGF124
Son, H.J. **MOPGF119**
Song, Y. THHD3O03
Song, Y. WEPGF020
Song, Y.G. **TUC3O06**
Sorrell, Z. THHB2O01
Soto, R. **MOM311**, WEPGF031
Sotoudi Namin, H. MOA3O02
Spinka, T.M. MOB3O01
Splawa-Neyman, P. WEPGF059
Spruce, D.P. MOB3O03, MOPGF179, WEM309,

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

WEPGF120
Staack, M. MOA3O02, MOM308
Staig, T.I. WEPGF031
Stamos, K.S. WEPGF046
Stankiewicz, M.J. MOPGF179
Starritt, A. C. WEM303
Stavitski, E. WEPGF080
Stebel, L. MOPGF070
Stechmann, C. MOA3O02
Stephenson, M. MOM312
Stevenson, A.W. MOK03K01
Stieger, B. MOPGF025, TUA3O01, WEPGF013
Stout, E.A. MOD3O03
Strangolino, G. WED3O04, **WEPGF152**
Straumann, T. MOPGF002, MOPGF038, WEPGF122
Strniša, K. FRB3O02
Styczen, B. TUD3O03
Su, S. MOPGF063
Suetake, M. WEC3O04
Suh, Y.J. WEPGF115
Suljoti, E. THHA3O02
Sumorok, K. MOPGF025, WEPGF013
Sundal, M.V. MOPGF091
Sušnik, T. WEPGF015
Sutton, I. TUB3O03
Suwada, T. MOPGF035, THHC2O01
Suwalska, A. WEPGF141
Suxdorf, F. MOPGF063
Swart, P.S. **MOD3O06**, WEPGF097
Szalai, B. MOPGF051
Szász, P. MOPGF051
Szczesny, J. MOA3O02
Sztuk, J. WEPGF050
Szuba, J. MOA3O02
Szuk, B. THHB2O02
Szymocha, T. MOPGF179

— T —

Takabayashi, Y. MOPGF023
Takagi, M. MOPGF035
Takebe, H. MOM305
Tamura, F. WEPGF121
Tanigaki, M. **MOPGF142**
Tao, F. **TUC3O07**
Tartoni, N. MOPGF070
Tateyama, T.M. MOPGF160
Taurel, E.T. WEA3O01
Tavares, D.O. **MOC3O04**, WEPGF128
Taylor, J.W. TUB3O03
Teichmann, M. MOA3O02, **WEPGF153**
Telford, S.J. MOB3O01, WEC3O05
Terpstra, W.W. MOPGF147, WEPGF119, **THHA2O03**
Theisen, C. THHB2O01
Thomas, D.J. WEPGF134
Thomas, G. **MOPGF120**
Thompson, D.A. WED3O01
Thompson, J.A. MOPGF056
Tian, Y. MOB3O02, **MOC3O05**, TUC3O05
Tikhomolov, E. MOPGF160, **WEPGF154**
Tilaro, F.M. WEPGF068
Ting, X. THHD3O01
Tobin, M. WED3O01
Tobiyama, M. WEC3O04
Todd, B. WEPGF005
Tolkiehn, J. MOA3O02
Toyoda, A. MOM309, MOPGF123
Trahern, G. FRB3O02
Tranquille-Marques, Y. **TUB3O04**
Trigilio, C. MOPGF163
Tsubota, K.T. **MOPGF040**
Turcato, M. WEPGF050
Turchetta, R. MOPGF070
Turcinovich, M. WEPGF037
Turner, C. MOPGF051
Turner, K.T. TUC3O07

— U —

Uchiyama, A. **WEPGF032**
Urpelainen, S. MOPGF065
Uyttenhove, S. MOPGF122
Uzun, I.S. MOPGF097, **MOPGF098**, **WEPGF089**

— V —

Vaga, F. WEC3O01, WEPGF062, THHA2I01
Valentini, F. MOPGF132, **MOPGF143**
Valentino, G. **MOPGF099**, **WEPGF066**
Valuch, D. WEPGF062
Van den Heever, L. WEPGF025
Van Esch, P. WEPGF083, WEPGF084
Van Trappen, P. **MOPGF121**, **MOPGF122**
van Waasen, S. MOPGF058, MOPGF063
Van Winckel, H. WEB3O05
Varela, F. MOPGF020, **MOPGF039**
Varghese, G. WEPGF015
Vasilyev, M.Yu. MOPGF080
Vasques Ribeira, D. TUD3O03
Vásquez, J.A. **MOPGF042**, WEM307
Vázquez-Otero, A. WEA3O01
Velez, G. WEPGF031
Verdier, P.V. WEA3O01, WED3O04, WEM310
Verga, N. MOPGF174
Vermersch, S. **MOC3O06**
Verstovšek, I. MOPGF125
Veseli, S. WEA3O02, **WEPGF116**, **THHC2O02**
Vestergard, H. MOPGF104
Vetter, S. MOPGF032
Veverka, J. MOPGF025, TUA3O01, WEPGF013
Viganò, W. THHB2O02
Vigder, M.R. **THHA3O03**
Villanueva, A.V. MOPGF091
Villanueva, J. MOPGF140
Viti, M. WEPGF015

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Vittor, D. MOPGF113
Voitier, A. MOPGF021, **WEPGF068**, **WEPGF069**
Volkov, V. MOPGF149
Voumard, N. MOPGF135
Vrcic, S. MOD3O06
Vuppala, V. WEPGF113

— W —

Wagener, M. MOPGF058
Walla, M. WEPGF006
Walton, R.D. WEB3O01, **THHB3O01**
Wan, K. WEPGF020
Wang, C.-J. FRB3O01
Wang, C.H. **WEPGF070**
Wang, J. WEPGF011
Wang, J.G. WEPGF117
Wang, J.Y. THHD3O03
Wang, L. WEPGF117
Wang, L.W. THHD3O03
Wang, W. MOD3O05
Wang, W.T. **THHD3O03**
Wang, Y.P. WEPGF034
Wang, Y.P. WEPGF011
Wang, Z. WEB3O04
Ward, K. MOPGF048
Warner, A. MOPGF145
Waters, G. MOPGF160
Watkins, H.A. MOPGF161
Wawrzyniak, A.I. MOPGF179
Wawrzyniak, K. MOPGF179
Weaver, S. MOD3O03
Webb, G.I. **THK3K01**
Weger, K. MOA3O02, WEPGF153
Wenninger, J. MOPGF099
Wesemann, M. MOPGF026
Weterings, W.J.M. MOPGF121
Wettenhall, J.M. WEPGF059
White, G.R. **WEA3O02**
Wiggins, J. MOA3O02, WEPGF153
Wilgen, J. MOA3O02, MOPGF101, TUD3O04
Wilksen, T. MOA3O02, TUD3O04
Williams, E. MOPGF002, WEPGF122
Willingham, D.S. **WED3O05**
Wilson, E.F. MOD3O03
Winter, G. WEPGF053
Wintersberger, E. WEPGF063
Witt, M. THHA3O02
Włostowski, T. **WEC3O01**, **THHA2I01**
Wojdyla, J.A. TUA3O02
Woods, K. MOPGF048
Wright, G. **WEM304**
Wrona, K. MOA3O02
Wu, C.Y. FRB3O01
Wu, H. MOA3O02
Wu, J.Q. MOPGF001
Wu, J.Y. MOPGF145
Wunderer, C. MOPGF070
Wychowaniak, J. TUD3O05, WEPGF015

— X —

Xaia, B. WEPGF118
Xia, Q. MOPGF070
Xiong, N. TUB3O01, **WEPGF071**
Xu, C. MOPGF038
Xu, H.S. THHD3O01
Xuan, K. **WEPGF117**
Xue, T. **WEPGF090**, THHB2O03

— Y —

Yakopov, M. MOA3O02
Yamamoto, N. MOPGF117, WEPGF121
Yamashita, A. **WED3O03**
Yan, Y.B. **WEPGF072**
Yang, B.Y. WEPGF074
Yang, L. MOB3O02
Yang, L. THHD3O01
Ye, Q. MOPGF134, MOPGF146
Yee, C. MOPGF002
Yin, C.X. WEC3O03, WEC3O04
Yogendran, P.J. MOPGF114, MOPGF160
Yokoyama, K. THHC2O01
Yoshida, S.Y. MOPGF117
Yoshifuji, N. MOB3O04, WEPGF085
Yoshii, A. WEPGF052
Yoshii, K. MOB3O04, WEPGF085
Yoshioka, M. WEPGF014
Young, A. MOPGF038
Youngman, C. MOA3O02
Yousef, H. MOPGF070
Yu, L. MOC3O05
Yuan, Z.Y. WEPGF061
Yue, M. WEPGF034
Yun, S.P. TUC3O06

— Z —

Zaera-Sanz, M. MOPGF138, TUC3O03, FRB3O02
Žagar, K. MOPGF105, WEPGF015
Zamantzas, C. THHB2O02
Zambon, L. WED3O04
Zamofing, T. FRA3O03
Zaza, S. MOPGF025, TUA3O01, WEPGF013
Zejdl, P. MOPGF025, TUA3O01, WEPGF013
Zelepoukine, S. MOPGF016
Zellweger, C. FRA3O03
Zenha-Rela, M. WEPGF046
Zerbi, F. WEPGF001
Zerlauth, M. WEPGF046, WEPGF047
Zhang, S.Z. **MOPGF047**
Zhang, W. MOPGF001, **WEPGF034**
Zhang, X. THHD3O01
Zhang, Y.L. TUC3O02

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

Zhao, L.Y.	WEC3O03	Zimmer, M.	MOPGF070
Zhao, Q.	WEPGF021, THHD3O01	Zimny, M.Z.	WEPGF094, THHD3O05
Zhao, Y.	**MOPGF146**	Zimoch, D.	FRA3O03
Zhou, Z.W.	MOPGF154	Zimoch, E.	FRA3O03
Zhu, P.	TUC3O02	Å»ytniak, Ł.	MOPGF179
Zhuang, M.	MOPGF154		

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Institutes List

AAO
North Ryde, Australia
- Farrell, T.J.
- Shortridge, K.

AGH University of Science and Technology
Kraków, Poland
- Goralczyk, L.

ALBA-CELLS Synchrotron
Cerdanyola del Vallès, Spain
- Avila-Abellan, J.A.
- Becheri, F.
- Broseta, M.
- Cuní, G.
- Falcón Torres, C.M.
- Fernández-Carreiras, D.
- Jover-Mañas, G.
- Matilla, O.
- Pascual-Izarra, C.
- Pastor Ortiz, R.
- Reszela, Z.
- Roldán, D.
- Rosanes Siscart, M.
- Rubio, A.
- Rubio-Manrique, S.
- Ruz, A.
- Salabert, J.
- Serra-Gallifa, X.
- Villanueva, J.

ALMA Observatory
Santiago, Chile
- Fuica, S.A.
- Ovando, A.
- Saez, N.
- Shen, T.C.
- Soto, R.
- Staig, T.I.
- Velez, G.

ALTEN CEA /CESTA
Merignac, France
- Gende, J.

ANL
Argonne, Ilinois, USA
- Arnold, N.D.
- Carwardine, J.
- Decker, G.
- Dickerson, C.
- Garcia, F.
- Jarosz, D.P.
- Johnson, A.N.
- Peters, C.E.
- Power, M.A.
- Schwarz, N.
- Veseli, S.

ANSTO
Menai, New South Wales, Australia
- Hauser, N.
- Mannicke, D.
- Stephenson, M.
- Xiong, N.

Aquenos GmbH
Baden-Baden, Germany
- Marsching, S.

AREVA TA Site LMJ CEA /CESTA
LE BARP cedex, France
- Bailleux, S.
- Chapron, N.

ASCo
Clayton, Victoria, Australia
- Acres, R.
- Afshar, N.
- Bambery, K.
- Hall, C.J.
- Hausermann, D.
- Livingstone, J.
- Maksimenko, A.
- Panjikar, S.
- Paterson, D.
- Pearson, J.
- Pelliccia, D.
- Ryan, C.G.
- Stevenson, A.W.
- Tobin, M.
- Willingham, D.S.

ASIPP
Hefei, People's Republic of China
- Hu, L.B.
- Zhou, Z.W.
- Zhuang, M.

BINP SB RAS
Novosibirsk, Russia
- Batrakov, A.M.
- Bolkhovityanov, D.
- Cheblakov, P.B.
- Emanov, F.A.
- Fatkin, G.A.
- Ilyin, I.V.
- Karnaev, S.E.
- Khudayberdieva, O.A.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- Korepanov, A.A.
- Panov, A.
- Vasilyev, M.Yu.

Birmingham University
Birmingham, United Kingdom
- Lurkin, N.

Bit Solutions
Bucharest, Romania
- Scarlat, Fl.

BNL
Upton, New York, USA
- Allan, D.B.
- Altinbas, Z.
- Arkilic, A.
- Binello, S.
- Brown, K.A.
- Carcassi, G.
- Caswell, T.A.
- Cerniglia, P.
- Chabot, D.
- Cheng, W.X.
- Chitnis, P.
- Costanzo, M.R.
- D'Ottavio, T.
- Dalesio, L.R.
- Davidsaver, M.A.
- De Long, J.H.
- Fu, W.
- Gu, X.
- Ha, K.M.
- Ha, K.
- Harvey, M.
- Hu, Y.
- Hulsart, R.L.
- Ilinski, P.
- Jamilkowski, J.P.
- Kadyrov, R.A.
- Kankiya, P. K.
- Katz, R.A.
- Kraimer, M.R.
- Laster, J.S.
- Lewis, W.K.
- Malitsky, N.
- Martins, B.S.
- Mead, J.
- Michnoff, R.J.
- Miller, T.A.
- Morris, J.
- Nemesure, S.
- Padrazo, D.
- Pekrul, W.E.
- Perez, S.
- Piacentino, J.
- Samms, T.
- Seletskiy, S.
- Sheehy, B.
- Shrey, T.C.
- Shroff, K.
- Singh, O.
- Smith, R.M.
- So, S.
- Sorrell, Z.
- Stavitski, E.
- Theisen, C.
- Thomas, D.J.
- Tian, Y.
- Yang, L.
- Yu, L.

BUTE
Budapest, Hungary
- Majzik, I.

CALTECH
Pasadena, California, USA
- Rollins, J.G.

CASS
Epping, Australia
- Hoyle, S.A.
- Marquarding, M.

CEA/DAM/DIF
Arpajon, France
- Nicoloso, J.

CEA
LE BARP cedex, France
- Caillaud, T.
- Fleury, J.
- Graillot, H.
- Nicoloso, J.
- Perez, S.
- Tranquille-Marques, Y.
- Vermersch, S.

CERN
Geneva, Switzerland
- Aguilera-Padilla, C.
- Alessio, F.
- Alves, D.
- Andreassen, O.Ø.
- Andronidis, A.
- Antoine, A.
- Antoniotti, F.
- Apollonio, A.
- Aragon, F.
- Arruat, M.
- Augrandjean, F.
- Augustinus, A.
- Barbosa, J.
- Barillère, R.
- Barros Marin, M.
- Baud, G.
- Baudrenghien, P.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- Bellorini, F.
- Bes, M.
- Blanchard, S.
- Blanco Alonso, J.
- Blanco Vinuela, E.
- Bland, A.
- Boccardi, A.
- Boivin, J-P.
- Bond, P.M.
- Booth, W.
- Boychenko, S.
- Bradu, B.
- Brischetto, Y.
- Bräger, M.
- Buffat, X.
- Burdzanowski, L.
- Butterworth, A.C.
- Buttner, M.
- Cardoso, L.G.
- Carlier, E.
- Chanavat, C.
- Chareyre, V.
- Charrondière, C.
- Charrue, P.
- Chatzigeorgiou, N.
- Chaze, O.
- Chochula, P.Ch.
- Copy, B.
- Costa Lopez, X.B.
- Daligault, F.
- Damerau, H.
- Daniluk, G.
- Darvas, D.
- Davids, D.
- De Dios Fuente, A.
- Deghaye, S.
- Delamare, Ch.
- Deldicque, C.
- Di Cosmo, M.
- Di Giulio, L.
- Dobson, M.
- Dragu, M.
- Dupont, A.D.
- Ehm, F.
- Fantechi, R.
- Farnham, B.
- Fernández Adiego, B.
- Ferreira, R.
- Fraga, J.
- Fuchsberger, K.
- Gabourin, S.
- Gabriel, M.
- Galilée, M.A.
- Galindo, J.
- Gallerani, L.
- Gama, J.
- Garnier, J.C.
- Gąsior, M.
- Gaspar, C.
- Gayet, Ph.
- Gigi, D.
- Glege, F.
- Godineau, G.
- Golonka, P.
- Gomes, P.
- Gomez De La Cruz, M.F.
- Gonzalez-Berges, M.
- Gorbonosov, R.
- Gutierrez, A.
- Hakulinen, T.
- Hammouti, L.
- Havart, F.
- Hegeman, J.
- Hofer, J.
- Hoguin, F.
- Holme, O.
- Höfle, W.
- Jackson, S.
- Janulis, M.
- Jenninger, B.
- Jensen, L.K.
- Jiménez Estupiñán, R.J.
- Jurcso, P.
- King, Q.
- Kirchner, D.
- Kolad, B.
- Koza, M.
- Krakówski, P.
- Krol, K.H.
- Kurepin, A.N.
- Labrenz, M.
- Lebioda, K.T.
- Lechman, M.
- Ledeul, A.
- Levens, T.E.
- Lipinski, M.M.
- Lüders, S.
- Magnin, N.
- Magrans de Abril, M.
- Martel, P.
- Marti Martinez, R.
- Martini, R.
- Martino, M.
- Martins Ribeiro, T.
- Masetti, L.
- Mateo, F.
- Meijers, F.
- Meschi, E.
- Milcent, H.
- Molendijk, J.C.
- Morovic, S.
- Murillo-Garcia, R.
- Niedziela, J.
- Nikiel, P.P.
- Ninin, P.
- Noulibos, R.
- Nunes, R.
- Nunez-Barranco-Fernandez, C.
- Ojeda Sandonís, M.
- Olexa, J.
- Orlandi, R.
- Orsini, L.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- Oser, P.
- Pache, J.
- Page, S.T.
- Patsouli, A.
- Pedersen, E.
- Pereira, H.F.
- Perrelet, D.
- Petrucci, A.
- Pezzetti, M.
- Piccinelli, F.
- Pigny, G.
- Pinazza, O.
- Poeschl, M.C.
- Prieto, P.P.
- Quilichini, M.
- Racz, A.
- Redaelli, S.
- Ribeiro, T.M.
- Rijllart, A.
- Rio, B.
- Roberts, P.
- Roderick, C.
- Sakulin, H.
- Salmon, J.L.
- Schlenker, S.
- Schmidt, R.
- Schwemmer, R.
- Schwick, C.
- Segura, G.
- Serrano, J.
- Silvola, R.P.I.
- Soare, C.-V.
- Sollander, P.
- Stamos, K.S.
- Stieger, B.
- Styczen, B.
- Suwalska, A.
- Szuk, B.
- Thomas, G.
- Tilaro, F.M.
- Todd, B.
- Uyttenhove, S.
- Valentini, F.
- Valentino, G.
- Valuch, D.
- Van Trappen, P.
- Varela, F.
- Vasques Ribeira, D.
- Vestergard, H.
- Viganò, W.
- Villanueva, A.V.
- Voitier, A.
- Voumard, N.
- Wenninger, J.
- Weterings, W.J.M.
- Włostowski, T.
- Zamantzas, C.
- Zaza, S.
- Zejdl, P.
- Zerlauth, M.
- Zimny, M.Z.

CIEMAT
Madrid, Spain
- Calvo, J.
- Jiménez-Rey, D.
- Molina Marinas, E.
- Mollá, J.
- Podadera, I.

CLS
Saskatoon, Saskatchewan, Canada
- Bree, M.
- Wright, G.

CNL
Ontario, Canada
- Cusick, M.L.
- Dean, D.
- Vigder, M.R.

Cosylab
Ljubljana, Slovenia
- Amand, F.
- Bobnar, J.
- Claesson, N.
- Cuk, G.
- Dolin'ek, I.
- Humar, T.
- Juvan, V.
- Komel, M.
- Maslov, P.A.
- Mehle, M.
- Pavleski, M.
- Pleško, M.
- Rojec, U.
- Sekoranja, M.
- Strniša, K.
- Sušnik, T.
- Söderqvist, A.A.
- Verstovšek, I.
- Žagar, K.

CPPM
Marseille, France
- Duval, P-Y.

Creighton University
Omaha, NE, USA
- Cherney, M.G.
- Fujita, J.

CSIRO ATNF
Epping, Australia
- Guzman, J.C.
- Mirtschin, P.L.
- Thompson, D.A.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Cyfronet
Kraków, Poland
- Szymocha, T.

Czech Republic Academy of Sciences, Institute of Physics
Prague, Czech Republic
- Rus, B.

DECTRIS Ltd.
Baden, Switzerland
- Rissi, M.

DELTA
Dortmund, Germany
- Althaus, A.
- Bahnsen, F.H.
- Schirmer, D.

DESY
Hamburg, Germany
- Ackermann, S.
- Aghababyan, A.
- Ayvazyan, V.
- Bacher, R.
- Bartkiewicz, P.K.
- Bayer, M.
- Behrens, U.
- Beutner, B.
- Boeckmann, T.
- Branlard, J.
- Bruns, B.
- Butkowski, Ł.
- Castro Carballo, M.E.
- Clausen, M.R.
- Correa, J.
- Decking, W.
- Delfs, T.
- Duval, P.
- Faatz, B.
- Fröhlich, L.
- Gerhardt, W.
- Gindler, C.
- Gnadt, P.
- Graafsma, H.
- Grecki, M.K.
- Göttlicher, P.
- Hatje, J.
- Hensler, O.
- Heuer, M.
- Hierholzer, M.
- Jäger, J.M.
- Kammering, R.
- Karstensen, S.
- Keller, H.
- Killenberg, M.
- Kocharyan, V.
- Korth, O.
- Kozak, T.
- Labudda, A.
- Lange, S.
- Limberg, T.
- Lomperski, M.
- Marras, A.
- Meykopff, S.M.
- Möller, M.
- Omet, M.
- Penning, J.
- Petrosyan, A.
- Petrosyan, G.
- Petrosyan, L.P.
- Petrosyan, V.
- Pfeiffer, S.
- Pototzki, P.
- Przygoda, K.P.
- Rehlich, K.R.
- Rettig-Labusga, S.
- Řeža, S.
- Rickens, H.R.
- Rybnikov, V.
- Schlarb, H.
- Schlesselmann, G.
- Schmidt, Ch.
- Schoeneburg, B.
- Scholz, M.
- Schreiber, S.
- Shehzad, N.
- Shevyakov, I.
- Smoljanin, S.
- Sombrowski, E.
- Staack, M.
- Stebel, L.
- Stechmann, C.
- Szczesny, J.
- Varghese, G.
- Viti, M.
- Walla, M.
- Wilgen, J.
- Wilksen, T.
- Wintersberger, E.
- Wu, H.
- Wunderer, C.
- Xia, Q.
- Yang, B.Y.
- Zimmer, M.

DLS
Oxfordshire, United Kingdom
- Abbott, M.G.
- Angelsen, C.
- Ashton, A.
- Bark, A.P.
- Basham, M.
- Battam, N.W.
- Chang, P. C. Y.
- Cobb, T.M.
- Cousins, A.M.
- da Graca, S.
- De Maio, N.
- Dent, A.J.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- Filik, J.
- Fisher, S.
- Friedrich, T.
- Furseman, M.J.
- Gayadeen, S.
- Gerring, M.W.
- Gillingham, I.J.
- Gu, Q.
- Heron, M.T.
- Hickin, D.G.
- Knap, G.
- Lay, S.C.
- Levik, K.
- Marchal, J.
- Mercado, R.
- Mita, C.
- Nutter, B.J.
- Palaha, A.S.
- Palmer, C.M.
- Pedersen, U.K.
- Quinn, P.D.
- Rees, N.P.
- Rehm, G.
- Rogers, W.A.H.
- Rose, A.J.
- Tartoni, N.
- Thompson, J.A.
- Uzun, I.S.
- Walton, R.D.
- Winter, G.
- Yousef, H.

EBG MedAustron
Wr. Neustadt, Austria
- Hager, M.
- Regodic, M.

EJIT
Hitachi, Ibaraki, Japan
- Iitsuka, Y.
- Yoshifuji, N.

Elettra-Sincrotrone Trieste S.C.p.A.
Basovizza, Italy
- Billè, F.
- Bogani, A.I.
- Borghes, R.
- Brun, F.
- Cautero, G.
- Chenda, V.
- Curri, A.
- Duic, V.
- Favretto, D.
- Follath, R.
- Gaio, G.
- Giacuzzo, F.
- Giuressi, D.
- Khromova, A.
- Kourousias, G.
- Lonza, M.
- Menk, R.H.
- Passuello, R.
- Pinaroli, G.
- Pivetta, L.
- Prica, M.
- Pugliese, R.
- Scafuri, C.
- Scalamera, G.
- Scarcia, M.
- Strangolino, G.
- Turcinovich, M.
- Vittor, D.
- Zambon, L.

ELI-ALPS
Szeged, Hungary
- Ács, P.
- Brockhauser, S.
- Farkas, S.
- Fülöp, L.J.
- Hanyecz, V.
- Kiss, M.
- Koncz, Cs.
- Koncz, M.T.
- Mohácsi, Á.
- Nelissen, K.
- Schrettner, L.
- Szalai, B.
- Szász, P.
- Turner, C.

ELI-BEAMS
Prague, Czech Republic
- Bakule, P.
- Drouin, M.A.
- Himmel, B.
- Horáček, J.
- Horáček, M.
- Kasl, K.
- Mazanec, T.
- Naylon, J.
- Škoda, P.
- Vázquez-Otero, A.

EPFL
Lausanne, Switzerland
- Nicoletti, A.

ESO
Santiago, Chile
- Ibsen, J.P.A.

ESRF
Grenoble, France
- Bourtembourg, R.
- Chaize, J.M.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- Coutinho, T.M.
- Götz, A.
- Pons, J.L.
- Taurel, E.T.
- Verdier, P.V.

ESS
Lund, Sweden
- Andersson, R.
- Bargalló, E.
- Bellorini, F.
- Birch, S.L.
- Brodrick, D.P.
- Carling, H.
- Cereijo García, J.
- Fernandes, R.N.
- Fernandez, L.
- Gahl, T.
- Gallese, B.
- Gysin, S.R.
- Hall-Wilton, R.J.
- Kirstein, O.
- Korhonen, T.
- Laface, E.
- Levchenko, N.
- Mansouri Sharifabad, M.
- Monera Martinez, A.
- Mudingay, R.
- Nordt, A.
- Paulic, D.
- Piso, D.P.
- Rathsman, K.
- Reščič, M.
- Richter, T.S.
- Sandström, A.
- Sutton, I.
- Taylor, J.W.
- Trahern, G.
- Zaera-Sanz, M.

ETH
Zurich, Switzerland
- Di Calafiori, D.R.S.
- Dissertori, G.
- Djambazov, L.
- Holme, O.
- Lustermann, W.

FastLogic Sp. z o.o.
Łódź, Poland
- Piotrowski, A.

Fermilab
Batavia, Illinois, USA
- Andre, J.M.
- Briegel, C.I.
- Carmichael, L.R.
- Crawford, D.J.
- Diamond, J.S.
- Liu, N.
- Martin, K.S.
- Mommsen, R.K.
- Neswold, R.
- Nicklaus, D.J.
- O'Dell, V.
- Sliczniak, M.Z.
- Warner, A.
- Wu, J.Y.
- Yuan, Z.Y.
- Zejdl, P.

FHI
Berlin, Germany
- Junkes, H.
- Schöllkopf, W.
- Wesemann, M.

FRIB
East Lansing, Michigan, USA
- He, Z.Q.
- Ikegami, M.
- Konrad, M.G.
- Liu, D.
- Maxwell, D.G.
- Shen, G.
- Vuppala, V.

FZJ
Jülich, Germany
- Bednarek, M.
- Bussmann, K.
- Drochner, M.
- Fleischhauer-Fuss, L.
- Janaschke, S.
- Keuler, S.
- Kleines, H.
- Mertens, K.-H.
- Su, S.
- Suxdorf, F.
- van Waasen, S.
- Wagener, M.

GMTO Corporation
Pasadena, USA
- Conan, R.

Greenfield Technology
Massy, France
- Camino, P.
- Monnier-Bourdin, D.

GSI
Darmstadt, Germany
- Ameil, F.
- Bär, R.
- Beck, D.
- Betz, C.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- Bräuning, H.
- Fitzek, J.
- Haseitl, R.
- Hoffmann, T.
- Hüther, H.C.
- Kester, O.K.
- Kreider, M.
- Lang, K.
- Lonsing, R.
- Miedzik, P.B.
- Milosic, T.
- Müller, R.
- Ondreka, D.
- Petit, A.
- Prados, C.
- Reiter, A.
- Schaller, A.
- Terpstra, W.W.

HEIA-FR
Fribourg, Switzerland
- Oberson, D.

Heidelberg University
Heidelberg, Germany
- Ritzert, M.

Hisense Co. Ltd.
Qingdao, People's Republic of China
- Wang, W.

HZB
Berlin, Germany
- Balzer, A.F.
- Birke, T.
- Falkenstern, F.
- Müller, R.
- Schälicke, A.
- Schierle, E.
- Suljoti, E.
- Witt, M.

HZDR
Dresden, Germany
- Kuntzsch, M.

HZG
Geesthacht, Germany
- Khokhriakov, I.A.

IAP
Frankfurt am Main, Germany
- Bai, J.N.

IBS
Daejeon, Republic of Korea
- Jang, H.
- Lee, S.
- Son, C.W.
- Son, H.J.

IFIN-HH
Bucharest - Magurele, Romania
- Cernaianu, M.O.

IHEP
Beijing, People's Republic of China
- Du, Y.Y.
- He, J.
- Hu, J.
- Jiang, X.S.
- Jin, D.P.
- Kopylov, L.
- Li, L.F.
- Liu, F.
- Merker, S.
- Mikheev, M.S.
- Wang, C.H.
- Ye, Q.
- Zhang, Y.L.
- Zhao, Y.
- Zhu, P.

ILL
Grenoble, France
- Cecillon, F.
- Cocho, C.
- Elaazzouzi, A.
- Le Goc, Y.
- Locatelli, J.
- Mutti, P.
- Ortiz, H.
- Plaz, M.
- Ruiz-Martinez, E.
- Van Esch, P.

IMP/CAS
Lanzhou, People's Republic of China
- An, S.
- Cui, W.
- Gou, S.Z.
- Gu, K.
- Guo, Y.H.
- He, Y.
- He, Z.
- Li, P.
- Liu, H.T.
- Liu, T.
- Liu, X.J.
- Luo, J.B.
- Luo, Y.
- Ting, X.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- Wang, J.
- Wang, Y.P.
- Wu, J.Q.
- Xu, H.S.
- Yang, L.
- Yue, M.
- Zhang, W.
- Zhang, X.
- Zhao, Q.

INAF - IRA
Bologna, Italy
- Bartolini, M.
- Orlati, A.
- Righini, S.

INAF - OA Teramo
Teramo, Italy
- Di Carlo, M.
- Dolci, M.

INAF - OAC
Selargius (CA), Italy
- Buttu, M.
- Fara, A.
- Migoni, C.
- Poppi, S.

INAF IRA
Bologna, Italy
- Nicotra, G.
- Nocita, C.

INAF-OACT
Catania, Italy
- Becciani, U.
- Costa, A.
- Ingallinera, A.
- Riggi, S.
- Schillirò, F.
- Trigilio, C.

INAF-OAT
Trieste, Italy
- Baldini, V.
- Calderone, G.
- Cirami, R.
- Coretti, I.
- Cristiani, S.
- Di Marcantonio, P.
- Marassi, A.
- Santin, P.
- Smareglia, R.

INAF-Osservatorio Astronomico di Brera
Merate, Italy
- Zerbi, F.

INEST
Hefei, People's Republic of China
- Song, Y.
- Wang, J.Y.
- Wang, L.W.

INFLPR
Bucharest - Magurele, Romania
- Scarisoreanu, A.M.
- Scarlat, F.

INFN- Sez. di Padova
Padova, Italy
- Bellato, M.A.
- Isocrate, R.

INFN-Bologna
Bologna, Italy
- Pinazza, O.

INFN/LNL
Legnaro (PD), Italy
- Bortolato, D.
- Gelain, F.
- Giacchini, M.G.
- Marcato, D.
- Montis, M.
- Pavinato, S.
- Pedretti, D.
- Ponchia, R.
- Vásquez, J.A.

IP SAS
Bratislava, Slovak Republic
- Lechman, M.

IST
Lisboa, Portugal
- Sundal, M.V.

ITER Organization
St. Paul lez Durance, France
- Lange, R.

J-PARC, KEK & JAEA
Ibaraki-ken, Japan
- Kamikubota, N.
- Sato, K.C.
- Yamamoto, N.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

JAEA/J-PARC
Tokai-mura, Japan
- Kawane, Y.
- Kikuzawa, N.
- Maruta, T.
- Miura, A.
- Moriyama, K.
- Nakatani, T.
- Tamura, F.
- Yoshii, A.

JAEA
Ibaraki-ken, Japan
- Ikeda, H.
- Kato, Y.

JAMK
Jyväskylä, Finland
- Kamarainen, H.T.T.

JASRI/SPring-8
Hyogo-ken, Japan
- Furukawa, Y.
- Hamada, Y.
- Ishii, M.
- Kago, M.
- Matsumoto, T.
- Matsushita, T.
- Okada, K.
- Yamashita, A.

JCF
PARIS, France
- Fouquet, J.C.

JINR/VBLHEP
Dubna, Moscow region, Russia
- Sedykh, G.S.
- Volkov, V.

JINR
Dubna, Moscow Region, Russia
- Andreev, V.
- Falaleev, V.
- Gorbachev, E.V.
- Kirichenko, A.
- Monakhov, D.V.
- Romanov, S.
- Rukoyatkina, T.V.

JLab
Newport News, Virginia, USA
- Larrieu, T. L.
- Slominski, R.J.

KAERI
Daejon, Republic of Korea
- Cho, Y.-S.
- Kim, D.I.
- Kim, H.S.
- Kwon, H.-J.
- Seol, K.T.
- Song, Y.G.
- Yun, S.P.

Kanto Information Service (KIS), Accelerator Group
Ibaraki, Japan
- Saotome, H.S.
- Takagi, M.
- Yoshida, S.Y.

KEK
Tsukuba, Japan
- Agari, K.
- Akiyama, A.
- Fujii, Y.
- Furukawa, K.
- Igarashi, S.
- Iida, N.
- Iwasaki, M.
- Kaji, H.
- Kamitani, T.
- Kazama, S.
- Kobayashi, T.
- Mikawa, K.
- Miura, T.
- Miyahara, F.
- Miyao, T.
- Morino, Y.
- Naito, T.
- Nakamura, T.T.
- Nakayoshi, K.
- Odagiri, J.-I.
- Ohnishi, Y.
- Sakashita, K.
- Sasaki, S.
- Sato, Y.
- Satoh, M.
- Seimiya, Y.
- Suetake, M.
- Suwada, T.
- Tobiyama, M.
- Toyoda, A.
- Yokoyama, K.

KIS
Ibaraki, Japan
- Asano, K.
- Hirose, M.

KIT
Eggenstein-Leopoldshafen, Germany
- Chilingaryan, S.A.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

- Haas, D.
- Kopmann, A.
- Mexner, W.
- Ressmann, D.

KU Leuven
Heverlee (Leuven), Belgium
- Deconinck, G.
- Pessemier, W.
- Raskin, G.
- Saey, P.
- Van Winckel, H.

Kyoto University, Research Reactor Institute
Osaka, Japan
- Tanigaki, M.

LAL
Orsay, France
- ElKamchi, N.
- Gauron, P.
- Monard, H.

LANL
Los Alamos, New Mexico, USA
- Baros, D.
- Björklund, E.
- Faucett, J.A.
- Garnett, R.W.
- Gioia, J.G.
- Hill, J.O.
- Marroquin, P.S.
- Paul, J.D.
- Pieck, M.
- Sedillo, J.D.
- Shelley, F.E.
- Smith, B.G.
- Watkins, H.A.

LBNL
Berkeley, California, USA
- Du, Q.

LLNL
Livermore, California, USA
- Bayramian, A.J.
- Bond, E.J.
- Bowers, G.A.
- Brunton, G.K.
- Casey, A.D.
- Charron, K.
- Conder, A.D.
- Conrad, B.A.
- Di Nicola, J.-M.G.
- Di Nicola, P.
- Fallejo, R.N.
- Fedorov, M.A.
- Fishler, B.T.
- Fleming, R.
- Fulkerson, E.S.
- Haefner, C.
- Hutton, M.S.
- Johnson, G.W.
- Kalantar, D.H.
- Kim, D.
- Koh, E.S.
- Lanning, R.K.
- Lau, G.
- Marshall, C.D.
- Mathisen, D.G.
- Miller Kamm, V.J.
- Nissen, J.D.
- Pacheu, V.
- Paul, M.
- Petersen, D.E.
- Reed, R.K.
- Reisdorf, P.D.
- Reisdorf, S.M.
- Rouse, J.
- Sanchez, R.J.
- Shaw, M.J.
- Spinka, T.M.
- Stout, E.A.
- Telford, S.J.
- Weaver, S.
- Wilson, E.F.

LNLS
Campinas, Brazil
- Bacchetti, M.
- Bongers, R.
- Brito, J.L.N.
- Cardoso, M.B.
- Coelho, E.P.
- Curcio, R.F.
- de Almeida, H.D.
- Donadio, M.P.
- Franco, J.G.R.S.
- Lisboa, R.P.
- Marques, S.R.
- Martins, J.P.S.
- Martins, L.A.
- Moreno, G.B.Z.L.
- Nallin, P.H.
- O'Dowd, F.P.
- Piton, J.R.
- Rodrigues, A.R.D.
- Sachinelli, L.D.S.
- Silva, M. E.
- Slepicka, H.H.
- Tavares, D.O.

MAX-lab
Lund, Sweden
- Al-Dmour, E.
- Bell, P.J.
- Cerenius, Y.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- Forsberg, J.
- Hardion, V.H.
- Hennies, F. H.
- Jamróz, J.J.
- Kjellsson, L.
- Larsson, K.
- Lidón-Simon, J.
- Lindberg, M.
- Martos, V.
- Michel, V.
- Milan, A.M.
- Persson, A.G.
- Såthe, C.
- Sjöblom, P.
- Sjöström, M.
- Spruce, D.P.
- Urpelainen, S.

MECOS AG
Winterthur, Switzerland
- Hubatka, M.

Mitsubishi Electric System & Service Co., Ltd
Tsukuba, Japan
- Aoyama, T.
- Fujita, M.
- Hisazumi, K.
- Ichikawa, T.
- Kudou, T.
- Kusano, S.
- Mizukawa, Y.
- Nakamura, T.
- Yoshii, K.

MIT
Cambridge, Massachusetts, USA
- Darlea, G.L.
- Gomez-Ceballos, G.
- Paus, C.
- Sumorok, K.
- Veverka, J.

Monash University, Faculty of Science
Clayton, Victoria, Australia
- Goscinski, W.J.

Monash University
Clayton, Australia
- Androulakis, S.
- Bertling, P.J.
- Buckle, A.M.
- Goscinski, W.J.
- Groenewegen, D.
- Hines, C.
- Kannan, A.
- McGowan, S.
- McIntosh, P.
- Meyer, G.R.
- Quenette, S.M.
- Rigby, J.
- Splawa-Neyman, P.
- Webb, G.I.
- Wettenhall, J.M.

Naples University Federico II, Science and Technology Pole
Napoli, Italy
- Arpaia, P.

National Centre for Radio Astrophysics, Tata Institute of Fundamental Research
Pune, India
- Gupta, Y.

NEXEYA SYSTEMS
LA COURONNE, France
- Minolli, S.M.

NRC-Herzberg
Penticton, BC, Canada
- Vrcic, S.

NSCL
East Lansing, Michigan, USA
- Berryman, E.T.

NSRRC
Hsinchu, Taiwan
- Chang, Y.-T.
- Chen, J.
- Cheng, Y.-S.
- Chiu, P.C.
- Hsu, K.T.
- Hsu, S.Y.
- Hu, K.H.
- Huang, C. H.
- Kuo, C.H.
- Lee, D.
- Liao, C.Y.
- Wang, C.-J.
- Wu, C.Y.

NSU
Novosibirsk, Russia
- Fatkin, G.A.

ORNL
Oak Ridge, Tennessee, USA
- Guyotte, G.S.
- Hartman, S.M.
- Kasemir, K.-U.
- Pearson, M.R.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

OSL
Cambridge, United Kingdom
- Greer, A.
- Mayer, C.

PAL
Pohang, Kyungbuk, Republic of Korea
- Baek, S.Y.
- Choi, H. J.
- Gil, K.H.
- Hyun, H.J.
- Kang, H.-S.
- Kim, C.
- Kim, J.H.
- Kim, K.S.
- Kim, K.W.
- Kim, S.H.
- Ko, I.S.
- Mun, G.
- Park, B.R.
- Rah, S.Y.
- Seo, K.W.
- Suh, Y.J.

PNPI
Gatchina, Leningrad District, Russia
- Filimonov, V.

PSI
Villigen PSI, Switzerland
- Aiba, M.
- Alarcon, A.D.
- Ambrosch, A.
- Anicic, D.
- Bertrand, A.G.
- Billich, H.R.
- Bitterli, K.
- Brands, H.
- Bucher, P.
- Celcer, T.
- Chevtsov, P.
- Chrin, J.T.M.
- Divall, E.J.
- Ebner, S.G.
- Eichin, M.
- Fernandez Carmona, P.
- Gasche, M.
- Gobbo, A.
- Grossmann, M.
- Haemmerli, F.
- Higgs, C.E.
- Hovel, T.
- Humar, T.
- Janousch, M.
- Janser, G.
- Johansen, E.
- Jud, G.
- Kalantari, B.
- Kapeller, R.
- Krempaská, R.A.
- Lauk, D.J.
- Laznovsky, M.P.
- Luscher, C.
- Lutz, H.
- Maier-Manojlovic, D.
- Mayor, A.
- Märki, F.
- Pal, T.
- Panepucci, E.H.
- Portmann, W.
- Rawat, A.
- Rees, S.G.
- Regele, H.A.
- Sala, L.
- Wang, Z.
- Wojdyla, J.A.
- Zamofing, T.
- Zellweger, C.
- Zimoch, D.
- Zimoch, E.

RAS/INR
Moscow, Russia
- Kurepin, A.N.

Research School of Astronomy & Astrophysics, Australian National University
Weston Creek, Australia
- Price, I.A.

RIKEN Nishina Center
Wako, Japan
- Fukunishi, N.
- Komiyama, M.
- Uchiyama, A.

RIKEN SPring-8 Center, Innovative Light Sources Division
Hyogo, Japan
- Fukui, T.
- Hosoda, N.

RIKEN SPring-8 Center
Sayo-cho, Sayo-gun, Hyogo, Japan
- Inagaki, T.
- Maesaka, H.
- Ohshima, T.
- Otake, Y.
- Sakurai, T.
- Takebe, H.

RIKEN/SPring-8
Hyogo, Japan
- Maruyama, T.

SAGA
Tosu, Japan
- Iwasaki, Y.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- Kaneyasu, T.
- Koda, S.
- Takabayashi, Y.

Sao Paulo University, São Carlos Campus
São Carlos, Brazil
- Grossi, D.R.

SESAME
Allan, Jordan
- Dabain, Y.S.
- Ismail, A.
- Saleh, I.

SES
Hyogo-pref., Japan
- Hasegawa, T.
- Yoshioka, M.

Shikoku Research Institute Inc.
Kagawa, Japan
- Nakanishi, Y.

Siemens AG, Corporate Technology
München, Germany
- Roshchin, M.

SINAP
Shanghai, People's Republic of China
- Chen, Z.C.
- Dai, X.L.
- Lai, L.W.
- Leng, Y.B.
- Liu, M.
- Yan, Y.B.
- Yin, C.X.
- Zhao, L.Y.

SKA South Africa, National Research Foundation of South Africa
Cape Town, South Africa
- Alberts, M.
- Brederode, L.R.
- de Villiers, C.C.A.
- Gatsi, T.
- Joubert, F.
- le Roux, G.M.
- Marais, N.
- Mokone, O.J.
- Ockards, M.T.
- Slabber, M.J.
- Swart, P.S.
- Van den Heever, L.
- Xaia, B.

SLAC
Menlo Park, California, USA
- Allison, S.
- Alverson, S.C.
- Babel, S.
- Boyes, M.
- Brown, G.W.
- Campell, M.L.
- Carrone, E.
- Chin, E.P.
- Condamoor, S.
- Cyterski, C.
- D'Ewart, J.M.
- Decker, F.-J.
- Ding, Y.
- Gilevich, S.
- Hoobler, S. L.
- Kim, K.H.
- Krejcik, P.
- Lam, B.
- Larsen, R.S.
- Li, L.F.
- Loos, H.
- Luchini, K.
- Maxwell, T.J.
- Murphy, J.M.
- Olsen, J.J.
- Rogind, D.
- Sass, R.C.
- Shankar, M.V.
- Shoaee, H.
- Smith, S.R.
- Straumann, T.
- Tao, F.
- Turner, K.T.
- Vetter, S.
- White, G.R.
- Williams, E.
- Xu, C.
- Yee, C.
- Young, A.

SLSA
Clayton, Australia
- Aishima, J.
- Aragao, D.
- Caradoc-Davies, T.
- Clift, M.
- Corvetti, L.
- Cowieson, N.
- Felzmann, C.U.
- Glover, C.
- Hobbs, N.
- Jong, L.M.
- Kappen, P.
- Martin, P.
- Mudie, N.
- Peele, A.
- Shepherd, E.L.
- Starritt, A. C.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

Solaris National Synchrotron Radiation Centre, Jagiellonian University
Kraków, Poland
- Stankiewicz, M.J.

Solaris
Kraków, Poland
- Bocchetta, C.J.
- Bulira, P.
- Goryl, P.P.
- Wawrzyniak, A.I.
- Wawrzyniak, K.
- Å»ytniak, Ł.

SOLEIL
Gif-sur-Yvette, France
- Abeillé, G.
- Abiven, Y.-M.
- Bisou, J.
- Blache, F.
- Buteau, A.
- Corruble, D.
- Elattaoui, X.
- Gagey, B.
- Kheffafa, C.K.
- Langlois, F.
- Lê, S.
- Leclercq, N.
- Medjoubi, K.
- Mocuta, C.
- Monteiro, P.
- Poirier, S.
- Renaud, G.
- Zhang, S.Z.

STFC/RAL/ISIS
Chilton, Didcot, Oxon, United Kingdom
- Akeroyd, F.A.
- Baker, K.V.L.
- Clarke, M.J.
- Howells, G.D.
- Keymer, D.P.
- Knowles, K.J.
- Moreton-Smith, C.
- Oram, D.E.

STFC/RAL
Chilton, Didcot, Oxon, United Kingdom
- Das, D.
- Guerrini, N.
- Marsh, B.
- Nicholls, T.C.
- Sedgwick, I.
- Turchetta, R.

Stony Brook University
Stony Brook, New York, USA
- Chitnis, P.
- Robertazzi, T.G.

Tata Research Development and Design Centre
Pune, India
- Banerjee, A.S.
- Muralikrishna, G.
- Patwari, P.
- Roy Chaudhuri, S.

TEMF, TU Darmstadt
Darmstadt, Germany
- Ferrand, T.

Tessella
Abingdon, United Kingdom
- Bell, M.
- Bush, I.A.
- Nelson, R.F.
- Ward, K.
- Woods, K.

The HDF Group
Champaign, Illinois, USA
- Koziol, Q.
- Pourmal, E.

The University of Melbourne
Melbourne, Victoria, Australia
- Buyya, R.

TRIUMF, Canada's National Laboratory for Particle and Nuclear Physics
Vancouver, Canada
- Dale, D.
- Ezawa, K.
- Fong, K.
- Hui, H.
- Iranmanesh, R.
- Kavarskas, J.
- Keitel, R.
- Morris, D.B.
- Nussbaumer, R.B.
- Pon, J.J.
- Rapaz, S.
- Richards, J.E.
- Rowe, M.
- Tateyama, T.M.
- Tikhomolov, E.
- Waters, G.
- Yogendran, P.J.

Tsinghua University
Beijing, People's Republic of China
- Gong, G.H.
- Li, H.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

- Li, J.M.
- Xue, T.

TUB
Beijing, People's Republic of China
- Gong, H.

TUL-DMCS
Łódź, Poland
- Cichalewski, W.
- Kozak, T.
- Makowski, F.
- Napieralski, A.
- Prędki, P.
- Wychowaniak, J.

U. Sannio
Benevento, Italy
- Girone, M.

UCLA
Los Angeles, California, USA
- Erhan, S.

UCSD
La Jolla, California, USA
- Branson, J.
- Cittolin, S.
- Holzner, A.
- Pieri, M.

UMD
College Park, Maryland, USA
- Levine, W.S.

UNIPD
Padova (PD), Italy
- Bertocco, M.

Univerity of Medicine and Pharmacy 'Carol Davila'
Bucharest, Romania
- Verga, N.

University of Bristol
Bristol, United Kingdom
- Page, R.F.

University of Coimbra
Coimbra, Portugal
- Zenha-Rela, M.

University of Oviedo
Oviedo, Spain
- Abalo Miron, D.

University of Pavia
Pavia, Italy
- Vaga, F.

University of Valladolid
Valladolid, Spain
- de Frutos, L.
- de Prada, C.
- Mazaeda, R.

University of Western Australia
Crawley, Australia
- Bond, C.S.

Università degli di Perugia
Perugia, Italy
- Crisanti, M.

Université de Genève, Observatoire Astronomique
Versoix, Switzerland
- Mégevand, D.

USTC/NSRL
Hefei, Anhui, People's Republic of China
- Huang, Z.
- Li, C.
- Li, J.Y.
- Li, W.
- Lin, L.
- Liu, G.
- Liu, Y.T.
- Song, Y.
- Wan, K.
- Wang, J.G.
- Wang, L.
- Xuan, K.

USTC
Hefei, Anhui, People's Republic of China
- Wang, W.T.

UW-Madison/PD
Madison, Wisconsin, USA
- Zelepoukine, S.

VINCA
Belgrade, Serbia
- Adzic, P.
- Cirkovic, P.
- Jovanovic, D.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

VSC-ESA
Valencia, Spain
- Ercolani, A.

W.M. Keck Observatory
Kamuela„ Hawaii, USA
- Mader, J.A.
- Tsubota, K.T.

Warsaw University of Technology, Institute of Electronic Systems
Warsaw, Poland
- Grzegrzólka, M.
- Rutkowski, I.
- Rybaniec, R.

XFEL. EU
Hamburg, Germany
- Abeghyan, S.
- Beckmann, A.
- Boukhelef, D.
- Coppola, N.
- Esenov, S.G.
- Fernandes, B.
- Gessler, P.
- Giambartolomei, G.
- Hauf, S.
- Heisen, B.C.
- Januschek, F.
- Karabekyan, S.
- Kumar, M.
- Kuster, M.
- Lang, P.M.
- Maia, L.G.
- Münnich, A.
- Parenti, A.
- Pflüger, J.
- Raab, N.
- Rüter, T.
- Silenzi, A.
- Sotoudi Namin, H.
- Sztuk, J.
- Szuba, J.
- Teichmann, M.
- Tolkiehn, J.
- Turcato, M.
- Weger, K.
- Wiggins, J.
- Wrona, K.
- Yakopov, M.
- Youngman, C.

ZENIKA
Paris, France
- Boissinot, G.

ZHAW
Winterthur, Switzerland
- Hilbes, C.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors