



The 10th International Workshop on **PERSONAL COMPUTER AND PARTICLE ACCELERATOR CONTROLS (PCaPAC 2014)** will be hosted by ANKA, the Synchrotron Radiation Source at Karlsruhe Institute of Technology.

14th - 17th OCTOBER 2014

CONTROL SYSTEMS

Status reports
Control system frameworks (EPICS, Tango, Tine)
Network design
Reliability and longevity of control systems
And other...

DATA ACQUISITION

Commercial off-the-shelf systems
(Real-time) operating systems
Embedded systems, PLC's, IOC's
And other...

MANAGEMENT SOFTWARE PROJECTS

Development processes
Quality assurance
In time development
And other...

DATABASES

Control DBs
Extremely large databases (XLDB)
noSQL DBs
And other...

USER INTERFACES AND DATA DISPLAYS

Mobile applications
Web services
High level applications
And other...

CYBER SECURITY

Gateways, internet connectivity
And other...

HARDWARE TECHNOLOGIES

DSP, FPGA and devices
Accelerator specific development
Timing systems
PC busses
And other...

**INFORMATION &
REGISTRATION:**
WWW.PCAPAC2014.KIT.EDU



Synchrotron Radiation Source ANKA

Karlsruhe Institute of Technology (KIT)
Campus North
Hermann-von-Helmholtz-Platz 1
76344 Eggenstein-Leopoldshafen

www.anka.kit.edu



Dear PCaPAC Participant,



Welcome to Karlsruhe!

The 10th International Workshop on Personal Computers and Particle Accelerator Control (PCaPAC 2014) is taking place in Karlsruhe, Germany, from October 14th to 17th 2014. The conference is organized and hosted by the Synchrotron Radiation Facility ANKA, which is part of the Karlsruhe Institute of Technology (KIT). KIT was founded in 2009 with the merger of Forschungszentrum Karlsruhe and University Karlsruhe and with over 9000 employees it is one of the largest teaching and research organizations in the world.

PCaPAC traditionally focuses on the development and management of control and data acquisition systems using PC technology.

This year the focus is on emerging technologies like modern – mobile – user interfaces, DSPs, FPGAs, modern data acquisition systems, cloud computing, software management and databases. An interesting topic will be XLDBs and noSQL DBs.

As is traditional, PCaPAC will start on the first day with a series of tutorials. You will have the chance to join these tutorials and learn ‘hands on’ directly from the experts:

- Tango Control System for newbies, organized by ESRF and SOLEIL, France
- Introduction to noSQL DBs, organized by INFN, Italy
- Introduction into the Web2cToolkit, organized by DESY, Germany
- Set up for success - Creating a development environment from scratch, organized by Softwareschneiderei GmbH, Germany
- Introductory workshop on programmable hardware, organized by KIT, Karlsruhe

The scientific program will include invited talks, contributed talks and a poster session. For the invited talks I am happy to welcome: Pieter Hintjens, subject ZeroMQ; Fabrizio Gagliardi, subject Grids and Clouds; Carola Lilienthal, subject Current Trends in Software Development Management; Heiko Ehrlichmann, subject Machine availability monitoring; Dirk Düllmann, subject overview of current database systems.

On behalf of the Local Organizing Committee, it gives us great pleasure to welcome the more than 100 contributors to PCaPAC 2014, to Karlsruhe, to Baden-Württemberg and to Germany.

The local organizing committee suggests that you see some of the countryside around the city of Karlsruhe. A wine tasting excursion is planned at a vineyard in the palatinate region. A location for technology lovers has been selected for the conference dinner, the Technik Museum Speyer, with a Buran space shuttle on display among other extraordinary exhibits.

Wolfgang Mexner
Chair PCaPAC 2014

Conference Organization

Conference Secretariat

Margit Costarelli
Tel.: +49 (0)721 608 26288
Fax: +49 (0)721 608 26789
E-Mail: pcapac2014@lists.kit.edu

Registration Office

Congress & more,
Klaus Link GmbH
Festplatz 3
76137 Karlsruhe Sebastian Keppler
Fon: 0721 626 939 12
Fax: 0721 626 939 28
E-Mail: kepler@congressandmore.de

Local Organizing Committee

Wolfgang Mexner
Michael Hagelstein
Doris Ressmann
Karlheinz Cerff
Thomas Spangenberg
Margit Costarelli
Leyla Jochim
Robert Sabo
Guido Becker
Michael Drees
Carina Braun

International Program Committee

Richard Farnsworth	Argonne National Laboratory, USA
Pavel Chevtsov	PSI, Switzerland
Matthias Clausen	DESY, Germany
Mark Plesko	COSYLAB, Slovenia
Alain Buteau	Synchrotron SOLEIL, France
P.D. Gupta	Raja Ramanna Centre for Advanced Technology, India
Mark Heron	Diamond, England
Elder Matias	Canadian Lightsource, Canada
Luciano Catani	INFN, Italy
Mark Plesko	CosYLAB, Slovenia
Ralph Lange	HZB (Bessy), Germany
Shen Guoboa	BNL, USA
Igor Verstovsek	CosyLab, Slovenia
Norihiko Kamikubota	KEK (Japan)
Akihiro Yamashita	Spring 8 (Japan)
Takashi Kosuge	KEK (Japan)
Philip Duval	DESY (Germany)
Ralph Baer	GSI (Germany)
Reinhard Bacher	Desy (Germany)

Contents

Preface

Foreword	i
Committees	iii
Contents	iv
	v

Papers

WCO101 – Drivers and Software for MicroTCA.4	1
WCO102 – Controls Middleware for FAIR	4
WCO103 – Integration of New Power Supply Controllers in the Existing Elettra Control System	7
WCO201 – Computing Infrastructure for Online Monitoring and Control of High-throughput DAQ Electronics	10
WCO202 – Data Management at the Synchrotron Radiation Facility ANKA	13
WCO203 – Profibus in Process Controls	16
WCO204 – A Prototype Data Acquisition System of Abnormal RF Waveform at SACLA	19
WCO205 – Upgrade of SACLA DAQ System Adapts to Multi-Beamline Operation	22
WCO206 – Sardana – A Python Based Software Package for Building Scientific Scada Applications	25
WCO207 – A New Data Acquisition Software and Analysis for Accurate Magnetic Field Integral Measurement at BNL Insertion Devices Laboratory	28
WPO001 – Integrating Siemens PLCs and EPICS over Ethernet at the Canadian Light Source	31
WPO003 – Setup of a History Storage Engine Based on a Non-Relational Database at ELSA	34
WPO004 – News from the FAIR Control System under Development	37
WPO005 – Progress and Challenges during the Development of the Settings Management System for FAIR	40
WPO006 – FESA3 Integration in GSI for FAIR	43
WPO007 – The FAIR R ³ B Prototype Cryogenics Control System	46
WPO008 – An Extensible Equipment Control Library for Hardware Interfacing in the FAIR Control System	49
WPO009 – An Optics-Suite and -Server for the European XFEL	52
WPO010 – A Unified Matlab API for TINE and DOOCS Control Systems at DESY	55
WPO011 – Vacuum Interlock Control System for EMBL Beamlines at PETRA III	57
WPO012 – The EMBL Beamline Control Framework BICFROCK	60
WPO013 – Status of the FLUTE Control System	63
WPO016 – Magnet Power Supply Control Mockup for the SPES Project	66
WPO017 – IFMIF EVEDA RFQ Local Control System to Power Tests	69
WPO018 – Upgrade of Beam Diagnostics System of ALPI-PIAVE Accelerator's Complex at LNL	72
WPO019 – STARS: Current Development Status	75
WPO020 – Development and Application of the STARS-based Beamline Control System and Softwares at the KEK Photon Factory	78
WPO021 – Renovation of PC-based Console System for J-PARC Main Ring	81
WPO022 – Control System of Two Superconducting Wigglers and Compensation Magnets in The SAGA Light Source	84
WPO023 – Personnel Safety System in SESAME	87
WPO024 – Clients Development of SESAME's Control System based on CSS	90
WPO026 – The Applications of OPC UA Technology in Motion Control System	93
WPO027 – The Measurement and Monitoring of Spectrum and Wavelength of Coherent Radiation at Novosibirsk Free Electron Laser	96
WPO028 – EPICS BEAST Alarm System Happily Purrs at ANKA Synchrotron Light Source	99
WPO029 – Implementation of the Distributed Alarm System for the Particle Accelerator FAIR Using an Actor Concurrent Programming Model and the Concept of an Agent	102
WPO030 – Vacuum Pumping Group Controls Based on PLC	105
WPO031 – Diagnostics Test Stand Setup at PSI and its Controls in Light of the Future SwissFEL	108
WPO032 – Magnet Measurement System Upgrade at PSI	111
WPO033 – Status of Control System for the TPS Commissioning	114
WPO034 – Network Architecture at Taiwan Photon Source of NSRRC	117
WPO035 – BPM Control, Monitor, and Configuration Environments for TPS Booster	120
WPO038 – A Modular Personnel Safety System for VELA based on Commercial Safety Network Controllers	123
TCO101 – Benefits, Drawbacks and Challenges During a Collaborative Development of a Settings Management System for CERN and GSI	126

TCO102 – Eplanner Software for Machine Activities Management	129
TCO103 – Recent Highlights from Cosylab	132
TCO201 – Managing the FAIR Control System Development	135
TCO202 – Status of Indus-2 Control System	138
TCO204 – First Operational Experience of the ICHAOS Framework	141
TCO205 – Conceptual Design of the Control System for SPring-8-II	144
TCO207 – Common Device Interface 2.0	147
TCO301 – Inexpensive Scheduling in FPGAs	150
TCO303 – TestBed - Automated Hardware-in-the-Loop Test Framework	153
TCO304 – Launching the FAIR Timing System with CRYRING	155
TCO305 – TCP/IP Control System Interface Development Using Microchip* Brand Microcontrollers	158
FCO106 – The Role of the CEBAF Element Database in Commissioning the 12 GeV Accelerator Upgrade	161
FPO001 – InfiniBand interconnects for high-throughput data acquisition in a TANGO environment	164
FPO002 – Picosecond Sampling Electronics for Terahertz Synchrotron Radiation	167
FPO006 – Integration of Independent Radiation Monitoring System with Main Accelerator Control	170
FPO008 – LabVIEW PCAS Interface for NI CompactRIO	173
FPO009 – HLS Power Supply Control System Based on Virtual Machine	176
FPO010 – The Software Tools and Capabilities of Diagnostic System for Stability of Magnet Power Supplies at Novosibirsk Free Electron Laser	179
FPO011 – PyPLC, a Versatile PLC-to-PC Python Interface	182
FPO012 – A Real-Time Data Logger for the MICE Superconducting Magnets	185
FPO013 – Beam Data Logging System Base on NoSQL Database at SSRF	188
FPO014 – New Data Archive System for SPES Project Based on EPICS RDB Archiver with PostgreSQL Backend	191
FPO015 – Device Control Database Tool (DCDB)	194
FPO016 – Status of Operation Data Archiving System Using Hadoop/HBase for J-PARC	196
FPO017 – Managing Multiple Function Generators for FAIR	199
FPO018 – Setup and Diagnostics of Motion Control at ANKA Beamlines	201
FPO019 – FPGA Utilization in the Accelerator Interlock System (About the MPS Development in the LIPAc)	204
FPO022 – New developments on the FAIR Data Master	207
FPO024 – First Idea on Bunch to Bucket Transfer for FAIR	210
FPO026 – ADEI and Tango Archiving System – A Convenient Way to Archive and Represent Data	213
FPO028 – Web Based Machine Status Display for the Siam Photon Source	216
FPO029 – Redesign of Alarm Monitoring System Application "BeamlineAlarmInfoClient" at DESY	219
FPO030 – Control System Software Environment and Integration for the TPS	222
FPO031 – Power Supplies Transient Recorders for Post-Mortem Analysis of BPM Orbit Dumps at Petra-III	225
FPO032 – TPS Screen Monitor User Control Interface	228
FPO034 – Beamline Data Management at the Synchrotron ANKA	231
FCO201 – Renovating and Upgrading the Web2cToolkit Suite: A Status Report	234
FCO202 – OpenGL-Based Data Analysis in Virtualized Self-Service Environments	237
FCO203 – Making it all Work for Operators	240
FCO204 – How the COMETE Framework Enables the Development of GUI Applications Connected to Multiple Data Sources	243
FCO206 – PANIC, a Suite for Visualization, Logging and Notification of Incidents	246
Appendices	249
List of Authors	249
Institutes List	253

DRIVERS AND SOFTWARE FOR MicroTCA.4*

M. Killenberg[†], L. Petrosyan, C. Schmidt,

Deutsches Elektronen-Synchrotron DESY, 22607 Hamburg, Germany

S. Marsching, aquenos GmbH, 76532 Baden-Baden, Germany

M. Mehle, T. Sušnik, K. Žagar, Cosylab d.d., SI-1000 Ljubljana, Slovenia

A. Piotrowski, FastLogic Sp. z o.o., 90-441 Łódź, Poland

T. Kozak, P. Prędkie, J. Wychowaniak, Łódź University of Technology, 90-924 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 PCIexpress driver and a register mapping library allow out of the box operation of all MicroTCA.4 devices which carry firmware developed with the DESY FPGA board support package. The control system adapter provides callback functions to decouple the application code from the middleware layer. Like this the same business logic can be used at different facilities without further modification.

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, a common high-speed data bus (PCIexpress in this case) 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.

THE LINUX KERNEL MODULE

The Linux kernel module (driver) provides access to the MicroTCA.4 devices via the PCIexpress bus. As the basic access to the PCIexpress 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 generic part provides all common structures and implements access to the PCIexpress 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 generic 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 all boards. For devices from other vendors the generic part enables out-of-the-box access to the basic features, which can be complemented by writing a driver module based on the generic driver part. Like this the interface in MTCA4U does not change and the new device is easy to integrate into existing software.

Improved DMA Performance

Latest developments have focused on the improvement of the DMA performance. For a universal driver it is important to keep the interfaces to the user space and the firmware simple. In addition, the implementation in firmware should not be too complicated. The available firmware uses a simple DMA core which allows to transfer one contiguous block of memory from the MicroTCA.4 board into one contiguous block of the memory of the CPU. After the transfer is finished, the CPU is notified via a PCIexpress interrupt. Memory allocated in user space is not contiguous in Linux, but memory allocated in the Kernel address space can be contiguous.

Our original DMA implementation was to allocate one big kernel buffer, perform the DMA transfer into this buffer and afterwards perform a copy to the user space (see Fig. 2 A). Performance measurements showed that the copy to user space took about 50 % of the time of the DMA transfer (PCIexpress Gen. 1 on an Intel Core i7 CPU), which was a significant performance penalty.

The improved version tries to minimise the time until the copy to the user space can start (copy latency) and already executes part of the copying while the rest of the data is still being transferred via DMA. For this it performs multiple DMA transfers of a smaller block size instead of transferring everything at once (B and C in Fig. 2). As every DMA transfer ends with an interrupt, this causes additional load on the system and an additional latency. For this reasons the number of DMA transfers should be limited. Version B in Fig. 2 for instance finishes later than version C, although the

* This work is supported by the Helmholtz Validation Fund HVF-0016 “MTCA.4 for Industry”.

[†] martin.killenberg@desy.de

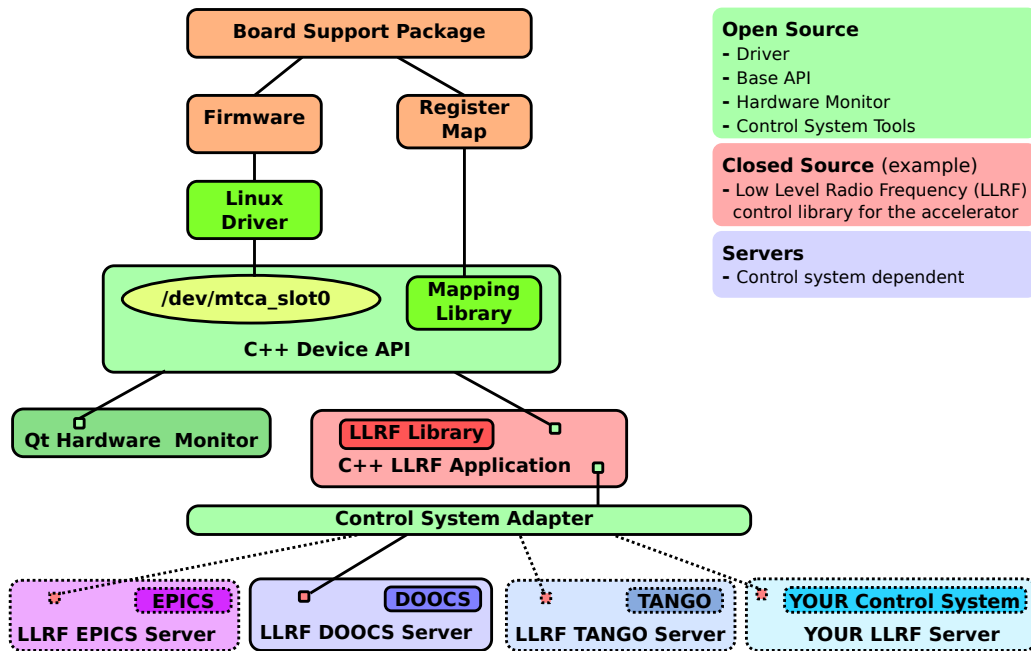


Figure 1: The design concept of the MicroTCA.4 User Tool Kit MTCA4U.

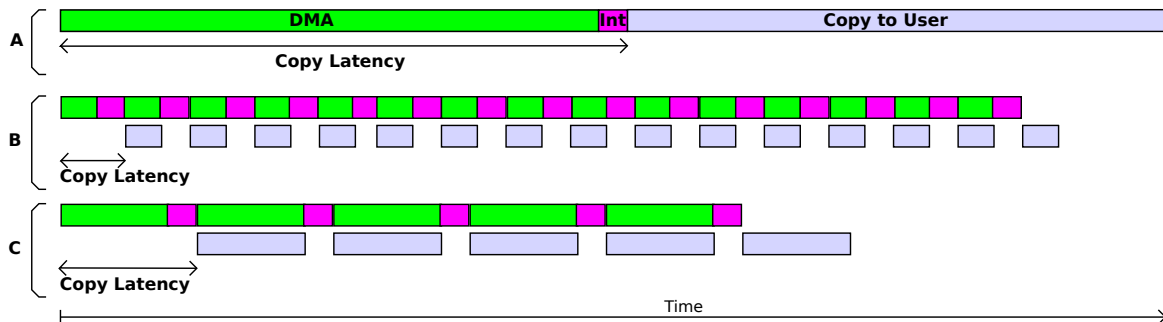


Figure 2: Visualisation of the time required to perform a DMA transfer for different methods.

latency until the first copy to user starts is much shorter. For our tests with a single PCIexpress card a number of about 10 buffers showed the best performance. The last copy to user, which is started after the last DMA transfer, takes about 5 % of the total time required for the DMA transfer, and the total interrupt latency is still almost negligible. This is a significant improvement compared to 50 % performance penalty of the original implementation.

THE C++ DEVICE API

The basic high level API provides C++ classes which allow access to all the functionality provided by the kernel module without requiring the user to have knowledge about implementation details like IOCTL sequences. The interface is that of a generic, address based device.

A main component of the C++ library is the register name mapping. With evolving firmware the address of a register can change in the PCIexpress 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. In addition

the mapping file can contain information for automatic data type conversion, for instance fixed point to floating point or signed 24 bit integer to system types. The required mapping file is automatically generated by the Board Support Package together with the firmware. Performance overhead due to repetitive table look-up is avoided by the use of register accessor objects, which cache the address and provide fast access to the hardware. Currently the mapping file implementation is being changed from plain text to XML. This allows for more flexibility and enables new features like composing the firmware out of precompiled blocks.

GRAPHICAL USER INTERFACE

The mapping file, containing information of all the PCIexpress registers implemented in the firmware, allows to display 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. 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.

THE CONTROL SYSTEM ADAPTER

For larger control applications the different components use a control system or a middleware layer to communicate with each other. For this purpose the control system provides data structures which are used inside the user code. This causes a strong coupling of the code to the control system.

As it is expensive and time consuming to develop control algorithms, it would be desirable to have the core application portable between different control system. This brings contradicting requirements: On one hand the application has to communicate via the control system protocol and provide functionality like logging and data history, on the other hand it should be independent from the specific control system implementation without reimplementing the functionality.

The approach in MTCA4U is the introduction of a control system adapter layer, which should be as thin as possible. The business logic, which is independent from the control system, is only talking to the adapter and does not use the control system directly. The adapter comprises two components: A process variable adapter and a callback mechanism to react on control system events.

The Process Variable Adapter

The process variable adapter is a wrapper around the actual control system specific instance of a variable. From the point of view of the business logic the process variables look like normal simple data types, or arrays of simple data types. Each variable has accessor functions (getters and setters). In addition to these methods, arrays will have iterators and random access operators similar to the C++ `std::array`. This allows to directly operate on the control system's data container without the need of an additional copy, which improves the performance for large data structures.

In addition to the basic accessors, callback functions can be registered which are executed when the variable content changes. Like this the process can react on changes coming from the control system. Apart from these functions, which have a control system independent interface, no other control system specific functionality is reflected by the interface of the process variable adapter. This approach also has an impact on the design of the application code: In principle it has to be able to run without a control system attached. The advantage here is that one can run it for testing with a small, local script or GUI without the need to set up a control system environment. In addition it simplifies unit testing.

Control System Callback Interface

There are three ways how actions in the business logic are called by or synchronised with the control system:

1. Synchronous actions with each read and write operation of a process variable. This is implemented by the “on set” and “on get” callback functions of the process variable adapter.
2. Update functions which are triggered by the control system. This can either be a periodic function or a function triggered by an outside event, like a trigger signal which is sent by the control system.
3. Synchronisation functions which are called within the user code. This might be needed if the user code is running its own thread which for instance might be triggered by the hardware and the business logic has to determine when the synchronisation is executed.

In the first two cases the timing is fully controlled by the control system. In the third option the execution of the synchronisation is triggered by the business logic, but it can be blocked by the control system adapter if there is ongoing communication.

For cases 2 and 3 all process variables are synchronised within one function. The control system adapter provides means to register these functions as callbacks and ensures thread safety for the process variables which are used in the function callbacks. Some control systems require a global lock to be set before accessing a process variable, but these implementation details are handled by the control system adapter, resulting in control system independent business logic.

CONCLUSIONS

The DESY MicroTCA.4 User Tool Kit MTCA4U is a C++ library which allows convenient access to MicroTCA.4 boards via PCIexpress. It comprises a modular, expandable Linux driver, an API with register name mapping and automatic type conversion, and a graphical user interface for fast prototyping. A control system adapter is currently being developed, which 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://svnsvn.desy.de/public/mtca4u>
- [4] The Qt Project, <http://qt-project.org/>

CONTROLS MIDDLEWARE FOR FAIR

V. Rapp, GSI, Darmstadt, Germany
W. Sliwinski, CERN, Geneva, Switzerland

Abstract

With the FAIR complex, the control systems at GSI will face new scalability challenges due to significant amount of new hardware coming with the new facility. Although, the old systems have proven themselves as sustainable and reliable, they are based on technologies, which have become obsolete years ago. During the FAIR construction time and the associated shutdown GSI will replace multiple components of the control system. The success in the integration of CERNs FESA and LSA frameworks had moved GSI to extend the cooperation with the controls middleware and especially Remote Device Access (RDA) and Java API for Parameter Control (JAPC) frameworks. However, the current version of RDA is based on CORBA technology, which itself, can be considered obsolete. Consequently, it will be replaced by a newer version (RDA3), which will be based on ZeroMQ, and will offer a new improved API based on the experience from previous usage. The collaboration between GSI and CERN shows that new RDA is capable to comply with requirements of both environments. In this paper we present general architecture of the new RDA and depict its integration in the GSI control system.

INTRODUCTION

Officially started in 2010 the FAIR (Facility for Antiproton and Ion research) [1] project will extend the existing GSI accelerator by many new installations. Those will introduce a plenty of new hardware, which need to be managed and monitored by the control system.

To benefit from the existing experience GSI has decided to start collaborating with CERN on the device software framework called FESA. This collaboration was successful, so the future devices, which will be introduced within the scope of FAIR will be developed with FESA framework. The FESA framework itself is designed to operate within the CERNs system context and depends on the underlying CERN middleware and the messaging layer called Remote Device Access (RDA). The adaptation of the FESA framework to the current GSI middleware is difficult. Moreover the current version of the GSI control system depends on CORBA [2] which itself is a rather an old messaging library with lack of support and shrinking community. Additionally this year CERN released a new RDA version based on the ZeroMQ [3] messaging framework, which is light, modern and well suited to operate in accelerator environment [4]. Therefore the further devices developed for the FAIR project will utilize the new middleware solution.

On the other hand GSI itself is running a considerable amount of old hardware. This is done by using of the own

developed middleware messaging layer called Device Access. Upgrading of all old hardware will be costly and difficult to accomplish. Hence the middleware architecture of FAIR will consist of two different parts: the CORBA based Device Access and the new ZeroMQ based RDA.

DEVICE ACCESS

Today's control system at GSI can be roughly divided into 3 parts. First the real time processes run on so called Front End Computers (FEC), which control the equipment and collect the information from it. On the other point of the architecture the mostly Java based Graphical User Interface (GUI) applications, which present the gathered information and allow operators in the control center to set hardware parameters of the beam equipment. Additionally the information from FECs is also monitored and collected by different automated services like logging or monitoring. Those two parts are connected by a middleware called Device Access.

Each process running on FEC is modelling a particular device of the accelerator equipment which is physically connected to FEC and managed by it. A typical device model is an object oriented device presentation, which consist of multiple readable and writable properties (e.g. device status, voltage, measured pressure etc.). Each model can also offer particular methods to control the connected device such as "reset" or "initialize", which can be called from other tiers using the middleware. To access the properties from another process like GUI, Device Access provides basic get, set and method call mechanisms, but also allows subscription for particular properties. Device Access also supports a given context for property values, which is used in multiplexed beam operation. To distinguish beams a simple numeric scheme is used. A basic number is associated with particular beam. Those numbers are referred as virtual accelerator.

Device Access allows addressing by device names. Those are rather convenient to users then the network addresses or CORBA IORs. It also provides flexibility to change the network address of particular FEC, without the need of changing the user software. The name resolution is done by the naming server, which stores the CORBA IORs of particular FECs assigned to their device names. The name resolution itself is transparent for the user and is performed by the middleware.

In addition to functions described above Device Access also provides a basis access right control mechanism [5]. This function is provided by the same server as the name resolution.

THE CERN CONTROL SYSTEM

The current control system at CERN is implemented in a similar way as the one at GSI. It is originally based on CORBA and operates with a similar device-property model. However with over 4000 servers [6] it handles a much larger amount of devices then its counterpart at GSI. Additionally the CERNs heterogeneous infrastructure with a lot of legacy equipment and operation systems is facing a bigger compatibility and management issues. Which means however, that it is proven to be flexible and performant enough to operate in environments like FAIR. Due of the scale and restrictive requirements considering logging and fault tolerance CERN is also running additional messaging systems on top control middleware [6].

CERN middleware is using its own developed role based access system called RBAC [7], which bring its own infrastructure and hence the only focus of the naming server (called Directory Server is CERNs context) is device name resolution.

RDA

One of the core parts of the CERNs control system is its messaging layer called Remote Device Access (RDA). Originally RDA was based on a CORBA technology, which can be considered obsolete. As many other CERN libraries RDA is provided natively for both C++ and Java platforms. C++ part of RDA depends on the OmniORB [8] library, while the Java part uses the JacORB [9]. In 2011 CERN have started a project to replace the underlying CORBA layer by modern software. After the evaluation of possible candidates to replace CORBA, ZeroMQ was chosen as the most suitable messaging framework for the Control System tasks [4]. In July 2014 a new version of the device access middleware was released - RDA3. Although, sharing the same concept it is a completely new development based on ZeroMQ. It introduced a new API, which is not compatible to the old one. This was done mainly to simplify the new interface and remove old ballast. The compatibility on the API level however can be achieved by using of additional proxy translation services, which are also a part of the middleware.

Similar to previous version RDA3 is operating on the same device-property model and supports get, set and subscribe operations on defined properties. Same as RDA2 both C++ and Java versions are provided. They share a similar API with same class, method and property names. Both versions are independent from each other and can be used natively in according environment.

RDA uses the same device property abstraction level allowing the users an easy and transparent access to the properties. The naming resolution is achieved by the same service called Directory Server as in previous version. To manage this, the Directory server is keeping track of all deployed devices. Therefore each running RDA server is registering himself on Directory Server and providing him with the necessary connection information (see Fig. 1).

The server is compatible with both RDA2 and RDA3 and uses the same data structure to keep the information. Beside of the servers name the connection information contain a CORBA IOR. In case of the RDA2 or a specific connection identifier string in case of RDA3. Using this data client part of the middleware is able to communicate with a desired server. The information about the particular devices deployed on the server is not transmitted in this step, since the implementation of the device model itself is done in the upper layer. To provide the connection information for a particular device the server is accessing a database, which contains the information about all installed device software and in particular on which RDA3 server those devices are deployed. This database is filled during the device deployment processes.

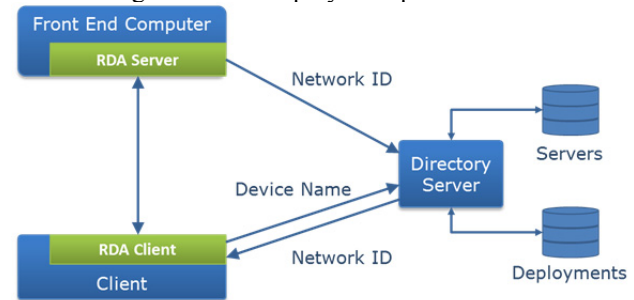


Figure 1: Communication with Directory Server.

Obviously the full and immediate replacement of old middleware by the new version cannot be done on the running system. Hence CERN will replace the old parts step by step. Some parts are already running with RDA3 and show good performance and user satisfaction.

FAIR MIDDLEWARE

As mentioned before the further device software developed for the FAIR project will be using the RDA framework in its latest version. In 2013 GSI has extended the collaboration with CERN by the controls middleware and especially RDA framework and is successfully contributing to the software. Beside of the knowledge exchange, this collaboration also allows GSI to implement own particular requirements.

The big amount of existing devices currently operated with the old Device Access software makes the migration to a new framework a difficult and costly task. Hence at the beginning of the FAIR operation both messaging layers will be present in the control system software landscape. Facing same issues on its side, CERN it has developed a transparent Java API called Java API for Parameter Control (JAPC). This API acts as an additional layer and hides the underlying middleware implementation from the user. It makes a middleware-unaware access possible. Most of the top level application is written in Java, so this API is developed only for this platform.

To access a particular device parameter the user needs only the name of device and property. Using of them he can create a Parameter object, which serves as an access point to particular property. Afterward the get, set or

subscribe is called on the created object. The interface remains the same for all message layers.

JAPC can be easily extended with plugins to work with different middleware, which made it an easy choice for the FAIR control environment. Since the Device Access layer is also a Java based client library, GSI has developed a JAPC plug-in to add the desired functionality (see Fig. 2).

The JAPC was adopted by GSI earlier, forwarded mostly by collaboration in other areas. Hence it was tested in production and has proven to be reliable and easy to use API.

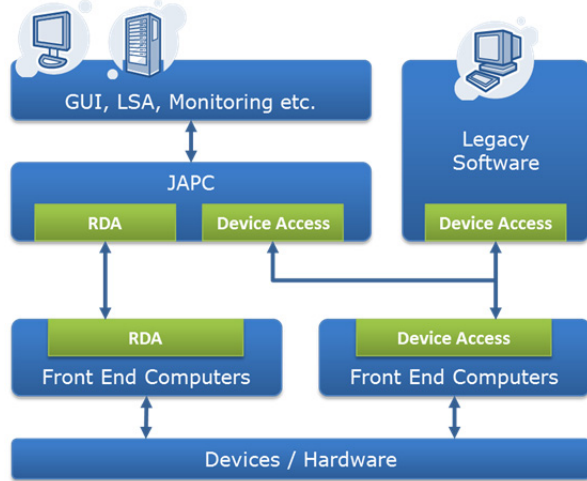


Figure 2: FAIR middleware overview.

STATUS AND FUTURE DEVELOPMENT

At the moment there are only a few devices in the running system based on the new middleware layer used mostly for the beam diagnostics. However multiple user interface applications use the JAPC API already to access particular properties. Those applications will be able to operate easily even if the accessing devices will migrate to the new middleware.

In the near future GSI will start the operation of the CRYRING [10]. Its control system will be based on the new framework. The pre-operational tests showed that the new middleware fulfills all requirements to run this system.

The start of the FAIR operation is scheduled for 2017, hence some of the details of the future middleware system are still a topic of ongoing discussions and some components of the future system are not fully specified.

One of the future developments shall consider the access control mechanisms. Currently the Device Access relies on its own authorization and authentication mechanism, while the RDA3 uses the RBAC, which is also a part of the Control Middleware at CERN. There is no intent to run both systems simultaneously.

As for any long running project there is probably no way to forecast all the requirements, which will be discovered during the development time. However, we are sure that the RDA framework provides the necessary flexibility to fulfil those.

SUMMARY

Control system middleware is integral part of the future FAIR software landscape. With the new facility a large amount of new devices will be put in operation. The extension and further usage of the old communication layer based on CORBA technology will be a difficult task. In addition the CORBA itself can be considered as obsolete. Hence the usage of the old communication framework for the coming system would be bound with many difficulties and costs.

On the other hand the existence of a bigger amount of devices, based on older hardware, make it a costly and difficult task to migrate everything to a new base in one step.

This dilemma can be solved by using generic and middleware unaware API – JAPC, which allows the usage of multiple messaging frameworks simultaneously. Moreover it opens the possibility to change the communication layer of particular devices, without the need of adjusting of the accessing software.

The successful collaboration of GSI on FESA framework and the choice of it as the main system for development of the device software determines the usage of the RDA as the messaging layer between the control applications and devices. Its new version is based on the ZeroMQ framework and fits the requirement of FAIR middleware.

The usage of CERN components does have a great impact on the architecture of the system. So in many parts it will be derived from the existing architecture of CERN.

As it already showed by CERN, the RDA based middleware is providing the required flexibility and scalability to operate accelerator systems of a bigger scale. With it the FAIR control systems will be prepared for the coming challenges.

REFERENCES

- [1] FAIR Project, website: <http://www.fair-center.de>
- [2] CORBA, Common Object Request Broker Architecture, OMG, <http://www.omg.org>
- [3] iMatix ZeroMQ, website: <http://www.zeromq.org>
- [4] A. Dworak et al, "Middleware Trends and Market Leaders 2011", ICALEPS2011, Grenoble, France.
- [5] S. Matthies et al, "Access Control in the Renovated GSI Control System: A Combined Name- and Rights-Server", PCaPAC08, Ljubljana, Slovenia.
- [6] F.Ehm, "Running a Reliable Messaging Infrastructure for CERN's Control System", ICALEPS2011, Grenoble, France
- [7] I.Yastrebov, "Role-Based Access Control for the Large Hadron Collider at CERN", International Journal of Computers Communications & Control, ISSN 1841-9836, 5(3):398-409, 2010
- [8] OmniORB, website: <http://omniorb.sourceforge.net>
- [9] JacORB, website: <http://www.jacorb.org>
- [10] CRYRING@ESR, website: <https://www.gsi.de/index.php?id=4570&L=1>

INTEGRATION OF NEW POWER SUPPLY CONTROLLERS IN THE EXISTING ELETTRA CONTROL SYSTEM

C. Scafuri, S. Cleva, Elettra - Sincrotrone Trieste S.C.p.A., Trieste, Italy

Abstract

The Elettra control system has been running since 1993. The controllers of the storage ring power supplies, still the original ones, have become obsolete and are no more under service. A renewal to overcome these limitations is foreseen. A prototype of the new controllers based on the BeagleBone embedded board and an in-house designed ADC/DAC carrier board, has been installed and tested in Elettra. A TANGO device server running in the BeagleBone is in charge of controlling the power supply. In order to transparently integrate the new TANGO controlled power supplies with the existing Remote Procedure Call (RPC) based control system, a number of software tools have been developed, mostly in the form of TANGO devices and protocol bridges. This approach allows us to keep using legacy machine physics programs when integrating the new TANGO based controllers and to carry out the upgrade gradually with less impact on the machine operation schedule.

RENEWAL OF POWER SUPPLY CONTROLLERS

The Elettra synchrotron light source has been running since October 1993. Many of its subsystems have been replaced or refurbished since then, but some of the core systems, such as the magnet power supplies controllers [1], are still operating with the original components. Unfortunately their VME boards are no longer serviced by the manufacturer and their electronic components are no more available on the market. Although we have acquired the capability to repair some of the most common failures, we cannot guarantee the full working of this vital part of the control system in future, considering that the failure rate of the components may rise as they grow older. In order to prevent this potential crisis, we have started a project to renew this part of the control system, starting with the big magnets: bending, quadrupoles, sextupoles.

Requirements and Constraints

The new power supply controller (NewPSC) that replaces the old VME-based system must satisfy several requirements:

- extend the power supply (PS) control system lifetime for at least 10 years;
- use the existing PS electrical interface to the controller with no change on the power electronics side;
- use open design components;
- simplify the present two level architecture using a single controller connected to the PS hardware and to the Ethernet control network;
- extend diagnostics capabilities and functionalities;

The new PS controller must also respect some constraints:

- have equal or better performance and stability compared to the existing PS controller;
- easy installation and replacement;
- the integration of the new PS controller with the existing software of the control system must be transparent;
- run the machine with a mix of new and old PS controllers for testing or upgrading in batches;

The “NewPSC” BeagleBone Based Controller

The NewPSC is based on a BeagleBone board [2, 3], powered by an AM3358 or AM3359 system-on-chip running at 720 MHz; it has a 100 Mb/s Ethernet interface, expansion connectors for I/O pins, open source approach for both hardware and software and runs several flavors of Linux. This powerful and inexpensive board is used as “smart node” in several applications at Elettra [4–6].

The BeagleBone provides the computing power, Ethernet connectivity for the NewPSC to the rest of the control systems and interfaces to the power supply local electronics via a dedicated carrier board developed in house [7]. The carrier board provides analog to digital conversion (ADC), digital to analog conversion (DAC) and several digital input and output channels, replacing the functions formerly carried out by several VME boards (see Fig. 1). The electrical performance has been optimized and characterized [8] in order to guarantee the performance of the NewPSC is equal or better than that of the old system.



Figure 1: The NewPSC prototype.

INTEGRATION WITH THE EXISTING CONTROL SYSTEM

The control of the NewPSC is implemented as an embedded TANGO device server running on the BeagleBone. TANGO has been adopted at Elettra in 2004 and all new developments since then have been based on TANGO. Many parts of the control system and high level programs are however still running with the old CERN nc/rpc [9] based services, including the controllers of the magnet power supplies. We had to provide some means to integrate the NewPSC TANGO devices with the legacy nc/rpc based programs in

ISBN 978-3-95450-146-5

order to guarantee uninterrupted and smooth operations of the accelerator; moreover we wanted to test the eventual impact of the NewPSC on the performance of the accelerator without introducing unneeded changes to the operating environment. This was accomplished by developing a protocol bridge, known as *lpc2tango*.

Lpc2tango Protocol Bridge

The old Elettra control system is based on 3 levels of computers. The middle level computers, named Local Process Computers (LPC) handle the data exchange from top level computers and applications to the rest of the system; applications interact with LPCs via the so called *lpc* service. This service, based on CERN *nc/rpc* protocol, allows to read and write (get / set) control system variables identified by names (in the form of C strings) and to issue simple commands (no data associated) also identified by names. The control system variables can be of different types: double, integer, string, boolean and their arrays. The *lpc* service implements the minimum set of features needed by a control system. The *lpc* service is supported by an auxiliary library implementing a static database used for associating a named variable to an host name and port number for address resolution.

Named control system variables are easily mapped to TANGO *device/attributes*, while simple commands are mapped to TANGO *command_inout* primitives in the special case without input and output parameters. In other words, the functionalities of the *lpc* service are a proper subset of those of TANGO. We have exploited this fact for developing a dedicated *rpc* server which does the actual mapping from *lpc* protocol (read or write of named variable) to the appropriate TANGO primitive (read or write of device/attribute, *command_inout*). The server is configured by means of a configuration file. TANGO exceptions are transformed into the appropriate error codes and messages too. The behavior of the *lpc2tango* server, from the client point of view, is identical to that of the native *lpc* server.

We took great care to design the software interface and behavior of the NewPSC TANGO device to be compatible with that of the old power supplies. This was a fundamental constrain to be satisfied, otherwise it would have been impossible to smoothly integrate programs written more than twenty years ago with servers written and deployed at the beginning of 2014. Accurate analysis and modeling are two keys of the success of the NewPSC integration.

TANGO Proxies for the Old Control System

The old Elettra control system uses a specialized *nc/rpc* based service for getting and setting the current of all the machine power supplies as a single operation. This service, named SARE, is used for performing the so called SAve/REstore operations. All the magnet power supplies settings are read from the equipment and saved as a single entity in a database table; the saved data is identified by a unique key. This key is used later to recover the settings and restore all the magnet power supplies settings. Save/restore is one

of the most important control room tools and is essential for accelerator operations.

We estimated that writing a protocol bridge for the SARE service would have been technically difficult and not worth the effort for two main reasons:

- need to modify some old code running on 68030 CPUs, with the need to recommission a system that has been running since 1993;
- save/restore is accessed by a single dedicated operator graphic panel;

The TANGO based snapshot database [10] and its companion tools provide the same functions of the old SARE system and its graphic panel. They are already extensively used at Elettra for the booster and FERMI operations, so we decided to use them also for the storage ring power supplies. In order to do this we wrote a set of TANGO devices which act as proxies to the old control system, calling the appropriate *rpc* functions of the *lpc* service for getting and setting all the power supply variables from TANGO. Also in this case we took great care to perform the correct analysis and implementation of the interface and behavior of the TANGO device so that it is compatible with the NewPSC. The programming effort was relatively light since we had to develop only two device servers for handling the differences between the power supplies of the big magnets and those of the steerer magnets. We also exploited inheritance features of TANGO in C++ for sharing about 50% of the code of the two devices.

A desired effect of the implementation and deployment of the TANGO proxies for the power supplies is the possibility to use the old control system with many different tools developed with TANGO and not only with the snapshot database. For example we have recently deployed the Matlab Middle Layer [11] using the Soleil developed TANGO interface.

In the end, legacy and new applications work in a hybrid environment (see Fig. 2).

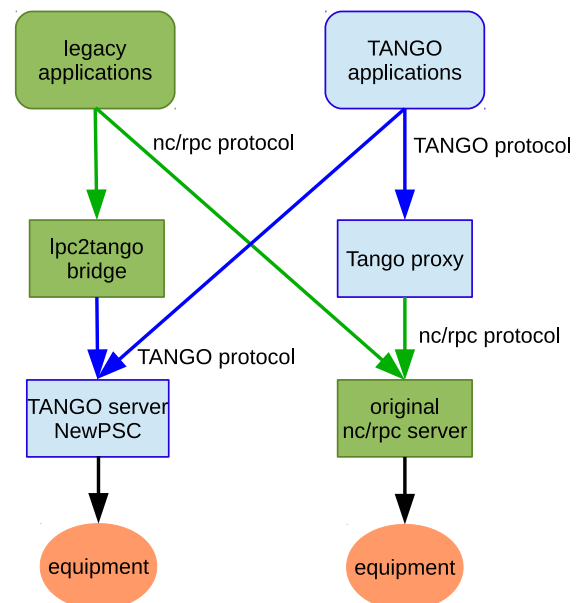


Figure 2: Bridges and proxies in hybrid environment.

STATUS AND FUTURE DEVELOPMENTS

Status

Three power supplies of the Elettra storage ring have been under full control of the NewPSC since march 2014 (see Fig. 3). The performance of the storage ring and that of the power supplies [12] have been completely satisfactory and no problems have been detected. The integration of the NewPSC has been almost transparent for operators and physicists and no additional downtime or impairment of operations have been experienced.



Figure 3: NewPSC installed in Elettra.

Following the positive results of the tests, the renewal of the controllers of the power supplies of the main Elettra magnets has been approved. We began the procurement procedure for building the required number of NewPSC controllers (42 plus spare parts) and planned their installation.

The NewPSC assemblies will be equipped with an LCD touch screen providing local control and diagnostics during maintenance activities.

Future Developments

We are designing the prototype of a new BeagleBone based controller for the Elettra steerer power supplies. The new design may also be used to upgrade the power electronics of the power supplies, moving from the present analog linear design to a digitally controlled PWM power module.

CONCLUSION

The integration of the NewPSC in the Elettra control system has been successful. During the more than six months long test of the first three pre-production controllers no disruptions or delay to machine operations due to the new sys-

tem occurred. We are able to run both legacy and new high level applications transparently using a mix of new and old power supplies controllers. We will thus be able to manage the transition to the new controllers with ease and flexibility: we can decide to upgrade all the power supplies in one go or in different batches.

A careful analysis and design of the NewPSC interface and behavior, which focused also on the backward compatibility with the old system, has been one of the keys of the project success.

REFERENCES

- [1] D. Bulfone, P. Michelini, "The ELETTRA Power Supply Control System", proc. EPAC 92, Berlin, 1992.
- [2] <http://www.beaglebone.org>
- [3] S. Cleva, A. I. Bogani, L. Pivetta, "A low-cost high-performance embedded platform for accelerator controls", proc. PCaPAC2012, Kolkata, India, 2012.
- [4] S. Cleva, L. Pivetta, P. Sigalotti, "Beaglebone for embedded control system applications", proc. ICALEPCS2013, San Francisco, CA, USA, 2013.
- [5] P. Cinquegrana et al., "Optical beam transport to a remote location for low jitter pump-probe experiments with a free electron laser", Physical Review Special Topics - Accelerators and Beams 17, 040702, 2014.
- [6] S. Cleva et al., "How Open Design Solutions are going to affect Particle Accelerators controls and diagnostics: the Elettra Case", AEIT Annual Conference 2014, From Research to Industry: The Need of a More Effective Technology Transfer, Trieste, Sept. 18-19, 2014.
- [7] R. Visintini, M. Cautero, S. Cleva, "New remote control strategies for the magnet power supplies of the Elettra Storage Ring", IECON 2013, Wien, Austria, 2013.
- [8] S. Cleva et al., "Upgrade of the Elettra magnet power supplies controllers", proc. IPAC2014, Dresden, Germany, 2014.
- [9] K. Kostro, "Remote procedure calls for accelerator control at CERN current status and future trends", Nuclear Instruments and Methods in Physics Research A, 1994, vol. 352, pages 262-264.
- [10] S. Pierre-Joesph et al., "Status of the Tango Archiving System", proc. ICALEPCS2007, Knoxville, TN, USA, 2007.
- [11] G. Portman et al., "An Accelerator Control Middle Layer using MATLAB", proc. PAC05, Knoxville, TN, USA, 2005.
- [12] S. Cleva et al., "Performance assessment of the new remote power supply controller for the Elettra Storage Ring magnets", IECON 2014, Dallas, USA, 2014.

COMPUTING INFRASTRUCTURE FOR ONLINE MONITORING AND CONTROL OF HIGH-THROUGHPUT DAQ ELECTRONICS

S. Chilingaryan, M. Caselle, T. Dritschler, T. Farago, A. Kopmann, U. Stevanovic, M. Vogelgesang, Karlsruhe Institute of Technology, P.O. Box 3640, 76021 Karlsruhe, Germany

Abstract

New imaging stations with high-resolution pixel detectors and other synchrotron instrumentation have ever increasing sampling rates and put strong demands on the complete signal processing chain. Key to successful systems is high-throughput computing platform consisting of DAQ electronics, PC hardware components, communication layer and system and data processing software components. Based on our experience building a high-throughput platform for real-time control of X-ray imaging experiments, we have designed a generalized architecture enabling rapid deployment of data acquisition system. We have evaluated various technologies and come up with solution which can be easily scaled up to several gigabytes-per-second of aggregated bandwidth while utilizing reasonably priced mass-market products. The core components of our system are an FPGA platform for ultra-fast data acquisition, Infiniband interconnects and GPU computing units. The presentation will give an overview on the hardware, interconnects, and the system level software serving as foundation for this high-throughput DAQ platform. This infrastructure is already successfully used at KIT's synchrotron ANKA.

INTRODUCTION

There are several challenging tasks to be solved while building the computing infrastructure for high-throughput DAQ electronics. The Linux Kernel Driver Interface is volatile and the kernel drivers are hard to develop and maintain. Additional complexity is added by the necessity to synchronize the development of detector hardware and the required readout software. Due to limited bandwidth of system memory, effective streaming of data requires a

very efficient realization of the DMA protocol. To reduce the impact of memory subsystem, the vendors of HPC hardware have developed special techniques, like *GPUDirect* [1]. If feedback loops are desired, large computing resources must be available to process dense streams of information. This is often solved using various accelerator cards with heavily parallel architectures. To reduce the latencies of feedback loops also techniques like *GPUDirect* may be applied. Another important aspect is the storage and preservation of the produced data. The storage subsystem should be able to handle several gigabytes of data per second for a long time. Finally, the question of scalability often arises. Systematically solving these questions is a difficult challenge and requires expertise in different areas of hardware, system, and software engineering.

Based on our experience building a high-throughput platform for real-time control of X-ray imaging experiments [2-5], we started to develop a hardware infrastructure and a software middleware with the goal to rapidly deploy data acquisition systems for different types of high-throughput detectors with the data rates up to 8 GB/s. We have evaluated various technologies and designed a scalable architecture based on Infiniband interconnects and GPU computing units. The core components of our system are an FPGA platform for ultra-fast data acquisition, the GPU-based Image Processing Framework “*UFO*”, and the fast control system “*Concert*” [6-7]. A lot of work was put in the system services to simplify interaction of the detector, hardware, and our software components. In this work we will focus on the hardware, interconnects, and the system level software serving as foundation for our platform.

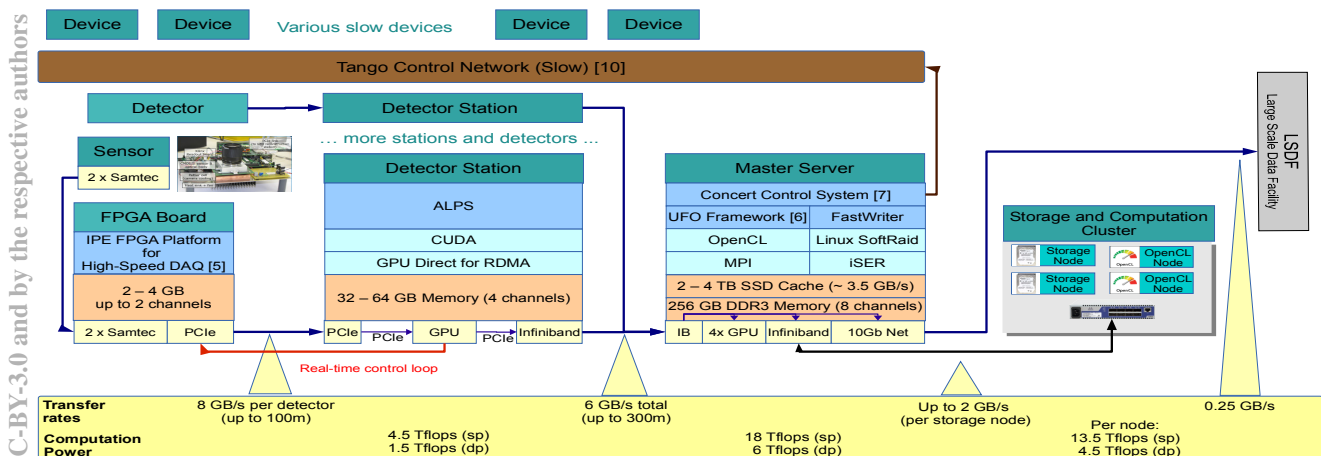


Figure 1: Architecture of control network.

ARCHITECTURE

The components of the platform are presented on the Figure 1. The detector electronics is built based on high-speed FPGA platform developed in-house. However, we can integrate 3rd party hardware as well. It is connected to detector station using external PCIe interface with electrical or optical cables. The optical connection induces a slightly higher latency but provides the option to place the computing infrastructure up to several hundred meters away from the detector (300 hundred with bandwidth limited to PCIe gen2). The detector station is selected mainly based on the memory throughput and availability of IPMI-type remote control. Currently we are using Asus Z9PA-U8 motherboard with Xeon E5-1620 v.2 processor. The station is equipped with 1-2 GPU cards for data preprocessing and to provide occasional feedback to the detector. From the detector station, the data is streamed to the central server equipped with large amount of memory and high-speed SSD Raid for caching overflow data. To ensure the high speed communication, a Infiniband link is used. The distributed setups are served with the optical cable allowing distances of up to 300 meters. The server is equipped with highly parallel computation units to allow on-line processing of complete data stream. Current configurations are based on Supermicro 7047GR-TPRF platform and equipped with 4x NVIDIA GTX Titan cards which provide reasonable compromise between good performance in both single- and double-precision computations and an affordable price. The system can be easily scaled up with PCIe expansion boxes. Such boxes usually contain 4 or more GPU cards and connected to the host system using a single x16 external PCIe interface. All GPUs in the box are commuted using PLX PCIe switch and transparently accessible to the host system as local devices. Further scalability can be achieved with commodity GPU-nodes integrated using the Infiniband fabric. The computational tasks are efficiently managed by *UFO Framework* and scheduled on cluster nodes with *MPI* and GPUs with *OpenCL* [6,8-9]. The *Concert* controls all experiment devices with Tango control system, but manages memory buffers for direct streaming from high-bandwidth detectors using Infiniband RDMA protocol [7,10]. This is released using KIRO (KIT Infiniband Remote cOmmunication) which extends TANGO protocol with secondary high-speed communication channel while still retaining standard control schemes over TANGO with high-throughput and very low latencies [11]. The received data is then passed to *UFO framework* for online processing. Finally, after analysis the selected data is moved for long-term preservation to large-scale storage facilities at computing center.

The high-speed storage is provided using a storage cluster. After evaluation of various cluster file systems like *GlusterFS*, *FhGFS* we found that a relatively high number of nodes are required to deterministically provide the desired streaming performance. To keep costs low we designed a storage work-flow consisting of 2 stages: real-time streaming and long-term storage. It is presented on the Figure 2. During the experiment, the data is streamed

to the smaller but faster real-time storage (RTS). In the pauses between data acquisition sessions, the data is moved from the RTS to the long-term storage. Each storage server in the cluster splits its disk space into the 2 parts. A small and fast space located in the outer edge of the magnetic disks is allocated for RTS. This storage is mapped directly to the Master server using the *XFS* file system over *iSER* (iSCSI over Infiniband) protocol [12-13]. With this solution we avoid performance drops due to network interlocking and may use the remote storage as a local block device. Multiple storage nodes are combined using *Linux Software Raid*. The bigger part of the storage is exposed to the clients and cluster computers using *GlusterFS*. It can't sustain high streaming rates, but provides a good overall performance when multiple clients are analysing the stored data. To avoid performance penalties by standard *POSIX* stack, we have developed an own data streaming library "*FastWriter*" based on *Linux Kernel AIO* [14]. Using just 2 storage nodes with 16 disks each we were able to achieve a stable streaming read-and-write performance of up to 2.5 GB/s on the RTS partition.

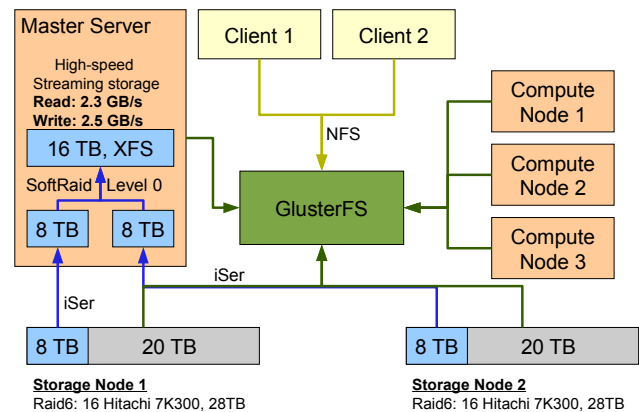


Figure 2: Architecture of the storage subsystem.

ALPS

The central software component is Advanced Linux PCI Services (*ALPS*) allowing us to rapidly implement software support for various PCI-based DAQ electronics. *ALPS* has a modular design and can easily be adopted to new hardware revisions, new DMA protocols, and even new detectors. It provides a flexible scripting interface that enable our hardware engineers to debug the developed electronics. The integration with commercial control software is planned in the next version through Web-Service interface.

The architecture of *ALPS* is presented on the Figure 3. To simplify maintenance we try to keep the Linux driver part as small as possible and move most of the functionality including the implementation of DMA protocol to the user-space library. Basically, the driver is only responsible for device configuration, interrupt handling, and management of DMA buffers. The library provides several API layers to work with the electronics. The PCI Memory layer provides unrestricted access to the device memory, mainly for debugging purposes. The register model is defined in XML and provides an easy

way to configure the device operation. To support more sophisticated hardware, additional registers can be defined in run-time. The high-speed data communication is carried out by DMA engine. The specific DMA protocols are implemented using plugins. The event engine defines an event-based model to integrate device-specific functionality. Each device can define multiple events and for each event several data types. The events will be triggered in hardware or requested by software. The client application may subscribe to get event notifications. Upon event notification, the application can request the desired type of data. The functionality of the library is fully exposed using command line interface.

Despite of user-space implementation, we are able to sustain the data rates produced by currently operating electronics (up to 2 GB/s) and there is high potential to scale for higher loads.

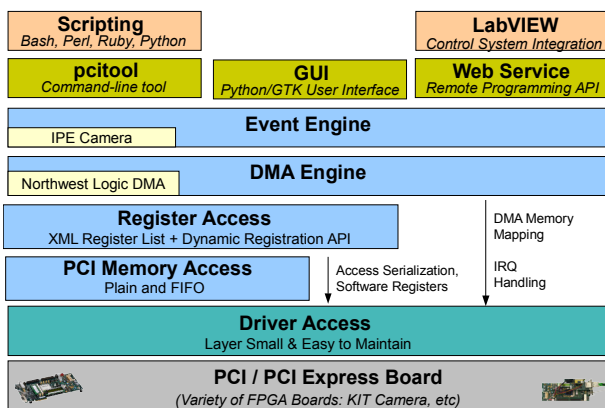


Figure 3: Advanced Linux PCI Services.

OUTLOOK

ALPS was developed having direct PCIe transfers in mind. Currently, we are working to implement the GPUDirect for RDMA and allow direct transfers from the detectors to GPU [1]. This will allow us to process a high data throughput in real-time and create feedback control loops with low latency. Considering that direct GPU-to-Infiniband transfers are already possible with Mellanox drivers, it is possible to configure a direct data flow from the camera to the main processing server bypassing the memory of readout PC at all. This makes high-performance detector readout superfluous and allows us to use small and energy efficient hardware instead. Hence, we get the opportunity to ingrate detector station and electronics. The integrated solution will have Infiniband interface instead of external PCI express which will made it plug-n-play and significantly easier to use. Development of this scheme is in progress using ARM-based SECO GPU Development Kit on the NVIDIA Kayla platform. Restriction to the proprietary NVIDIA *CUDA* technology applied by *GPUDirect* is a single drawback of the approach. However, the restriction only applies to detector station. Further processing at the computing cluster is possible with *OpenCL* which is integral part of our image processing framework and allows more flexibility in selection of parallel hardware.

REFERENCES

- [1] G. Shainer, A. Ayoub, T. Liu, M. Kagan, C. Trott, G. Scantlen, P. Crozier, "The development of Mellanox/NVIDIA GPUDirect over InfiniBand—a new model for GPU to GPU communications", Computer Science - Research and Development, vol 26, issue 3-4, pp. 267-273, 2011
- [2] S. Chilingaryan, A. Mirone, A. Hammersley, C. Ferrero, L. Helfen, A. Kopmann, T. dos Santos Rolo, P. Vagovic, "A GPU-Based Architecture for Real-Time Data Assessment at Synchrotron Experiments," IEEE Transactions on Nuclear Science, Volume 58, Issue 4, pp. 1447-1455, 2011
- [3] D. Haas et al. "Status of the Ultra Fast Tomography Experiments Control at ANKA." Proceedings of the PCaPAC. 2012
- [4] M. Caselle, S. Chilingaryan, A. Herth, A. Kopmann, U. Stevanovic, M. Vogelgesang, M. Balzer, M. Weber, "Ultrafast Streaming Camera Platform for Scientific Applications," IEEE Transactions on Nuclear Science, Volume 60, Issue 5, pp. 3669-3677, 2013
- [5] M. Caselle, M. Balzer, S. Chilingaryan, M. Hofherr, V. Judin, A. Kopmann, N. Smale, P. Thoma, S. Wuensch, A. Müller, M. Siegel, M. Weber, "An ultra-fast data acquisition system for coherent synchrotron radiation with terahertz detectors," Journal of Instrumentation, Volume 9, 2014
- [6] M. Vogelgesang, S. Chilingaryan, T. dos Santos Rolo, A. Kopmann, "UFO: A Scalable GPU-based Image Processing Framework for On-line Monitoring," Proceedings of HPCC-ICISS, pp. 824-829, 2012
- [7] M. Vogelgesang, A. Kopmann, T. Farago, T. dos Santos Rolo, T. Baumbach. "When Hardware and Software Work in Concert." Proc. of the 14th Intl. Conf. on Accelerator and Large Experiment Physics Control Systems, 2013
- [8] MPI: A Message-Passing Interface Standard. Available from: <http://www.mpi-forum.org/docs/docs.html>
- [9] OpenCL - The open standard for parallel programming of heterogeneous systems. Available from: <http://www.khronos.org/opencl/>
- [10] A. Gotz, E. Taurel, J. Pons, P. Verdier, J. Chaize, J. Meyer, F. Poncet, G. Heunen, E. Gotz, A. Buteau, et al., "Tango a corba based control system," in ICALPCS, 2003
- [11] T. Dritschler, S. Chilingaryan, T. Farago, A. Kopmann, M. Vogelgesang, "Infiniband Interconnects for High-Throughput Data Acquisition in Tango Environment," In proc. of the PCaPAC. 2014
- [12] XFS User Guide. Available from: <http://xfs.org>
- [13] M. Ko, J. Hufferd, M. Chadalapaka, U. Elzur, H. Shah, P. Thaler, "iSCSI Extensions for RDMA Specification," 2013. Available from: <http://www.rdmaconsortium.org>
- [14] M. Jones, "Boost application performance using asynchronous I/O," 2006. Available from: <http://www.ibm.com/developerworks/library/l-async/>

DATA MANAGEMENT AT THE SYNCHROTRON RADIATION FACILITY ANKA*

D. Ressmann, W. Mexner, A. Vondrous, A. Kopmann, V. Mauch, KIT, Karlsruhe, Germany

Abstract

The complete chain from submitting a proposal, collecting meta data, performing an experiment, towards analysis of these data and finally long term archive will be described. During this process a few obstacles have to be tackled. The workflow should be transparent to the user as well as to the beamline scientists. The final data will be stored in NeXus [1] compatible HDF5 [2] container format. Because the transfer of one large file is more efficient than transferring many small files, container formats enable a faster transfer of experiment data. At the same time HDF5 supports to store meta data together with the experiment data. For large data sets another implication is the performance to download the files. Furthermore the analysis software might not be available at each home institution; as a result it should be an option to access the experiment data on site. The meta data allows to find, analyse, preserve and curate the data in a long term archive, which will become a requirement fairly soon.

INTRODUCTION

The synchrotron radiation facility ANKA [3] is located at the Karlsruhe Institute of Technology, providing light from hard X-rays to the far-infrared for research and technology. It serves as a user facility for the national and international scientific community. The traditional data management for users of ANKA was based on transportable media. A user took its data home once the beam time was over. This approach is not feasible any more, for several reasons:

- Modern detectors produce more and more data
- No analysis resources at the home location
- Only manual or no bit preservation
- Poor search ability
- Difficult to share results with the global research community

In addition new regularities of the European Union research grants, request long term archiving for research data. Under discussion are periods of ten years. Therefore to preconfigure the analysis tools in a site local cloud environment would ease the usability. As a result we create a new workflow within the ASTOR project. This project is based on an Analysis-as-a-Service (AaaS) approach, based on the on-demand allocation of VMs with dedicated GPU cores and corresponding analysis environment to provide a cloud-like analysis service for scientific users [4]. In order to allow remote access to analyse and download the data, we had to implement an Authentication and Authorization Infrastructure (AAI). In this paper we discuss the workflow from a submission of

a proposal to authentication and authorization towards the beamline data management, the analysis and archival of the large raw datasets.

Submission of a Proposal

For all beam time proposals a peer review process is carried out. A user has to submit an experiment proposal in the ANKA proposal submission system ANNA. This is a web interface where details about the experiment are submitted and evaluated. For accepted proposals the users have the option to supply further meta data about each sample. The beam time will be scheduled and the user can prepare the travel to ANKA.

AUTHENTICATION AND AUTHORIZATION INFRASTRUCTURE

Usually a group of people is participating in one proposal and not all of these co-proposers are traveling to ANKA, however usually all members need access to the created data and the meta data. For the workflow at ANKA several web portals and services are required. To ease the usability a Single Sign On (SSO) approach will be provided. As a result each portal becomes a Service Provider (SP) and we introduce our own Intity Provider (IDP) for ANKA. This way we will also be able to include other IDPs, like Umbrella [5] or the DFN [6] (German National Research and Education Network), to authorize people. This has the big advantage, that user do not have to remember several accounts including their passwords.

During the experiment time at ANKA, the user can log on to the beam line data management (BLDM) frontend with the same credentials already used for the submission system.

THE BEAMLINE DATA MANAGEMENT

The beam line data management (see Figure 1) system collects all relevant data from the proposal system to make them available for the experiment. As such no duplicated data is inserted into the system, and proposal meta data can directly be connected to the measurements. This includes all logging information available via Tango [7], e.g. experimental data or vacuum level, sample conditions from the SCADA system WinCC Open Architecture [8]. Each sample gets an ANKA wide distinct identifier. Depending on the sample it can be identified via QR Code or Barcode. Once the user decides that this experiment is finished, one can close this newly created dataset, which triggers the beamline data management workflow.

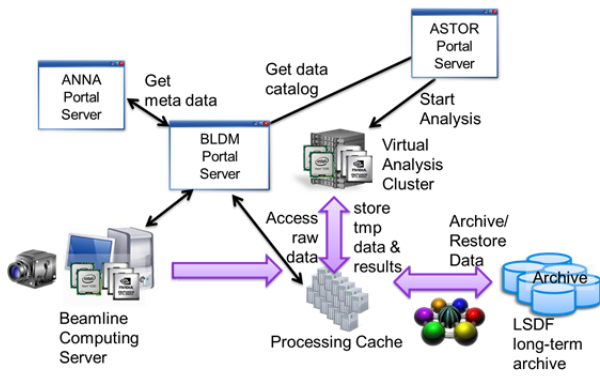


Figure 1: The Beamline Data Management.

Beside meta data acquisition and ingest into the data repository, the BLDM is responsible to hide distributed storage instances. An on site analysis is only feasible on the processing cache because the data archives are usually not designed for high-speed data access.

The BLDM is integrated into the TANGO control system and connects the storage resources to meet the requirements for data analysis and data ingest. The workflows and the current state of execution are persisted in a relational database. Monitoring and control of the data sets can be provided by the database. All TANGO devices of the BLDM are actively polling the database for work instructions like closing a dataset, computing the checksum, checking the checksum, migrating datasets or ingesting data. A message driven approach is planned to enable event driven task execution.

Storage

The idea is that each detector creates its own HDF5 file schema. When a dataset will be archived at the end of a scan, all HDF5 files produced with this scan are combined to one or more bigger file containing all data and meta data. An archived dataset can be moved around to different storage location, the analysis storage as well as the archival storage. To guarantee that each of these copies stays the same, the data cannot be changed anymore and only read access to this data is permitted. Optionally during this process a persistent identifier (PID) [9] can be created. When data will be archived for up to ten years, one has to make sure that the data can be recognised again. As time goes by, it will be difficult to remember what was measured. Keywords are needed to find this data again. As such all meta data created during a scan are stored inside of the dataset as well as inside of a database. The database can then be searched for all possible use cases.

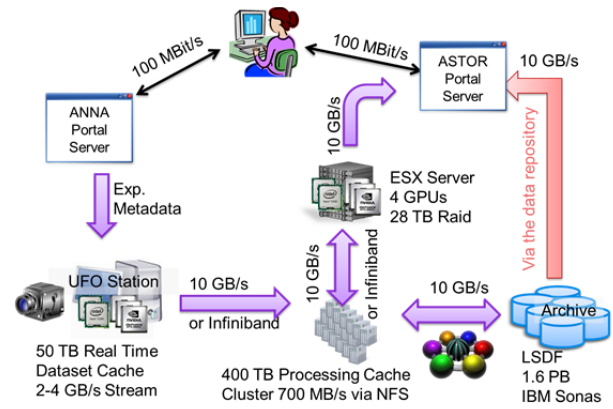


Figure 2: File based Storage.

Figure 2 shows the file based data storage approach. Another option would be to store the data not file based, within a no SQL database system (NoSQL DB) [10] as shown in Figure 3. This approach is currently being investigated to have faster parallel access of these data, if a cluster of NoSQL Nodes is used. This approach uses a key/value store, has a flexible scheme and is massive scalable. Just add more nodes to add further throughput. A NoSQL approach also supplies automatic redundancy, which results in higher performance and availability.

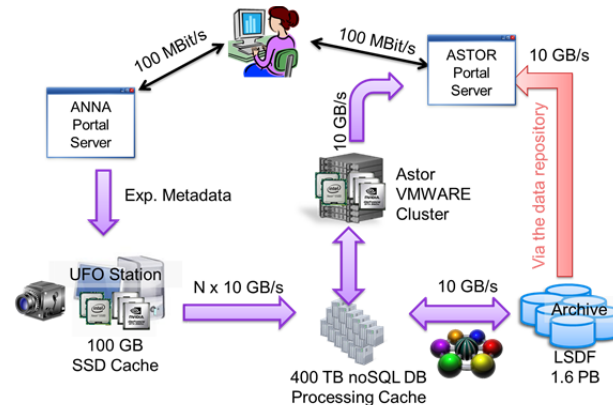


Figure 3: Storage in a NoSQLDB.

DATA ANALYSIS

Within the ASTOR project we develop an environment to analyse the data generated at ANKA with the help of virtual machines providing 3D data analysis tools like AMIRA [11] and volume graphics studio max [12] (see Figure 4). Each virtual desktop has access to its corresponding datasets through a dedicated mount point. This causes one complication. We need to map the authentication down to the file system layer in order to guarantee the access for authorized users and deny the access for others. We will have a file system group for each proposal. The members of this group will be all users participating in this proposal.

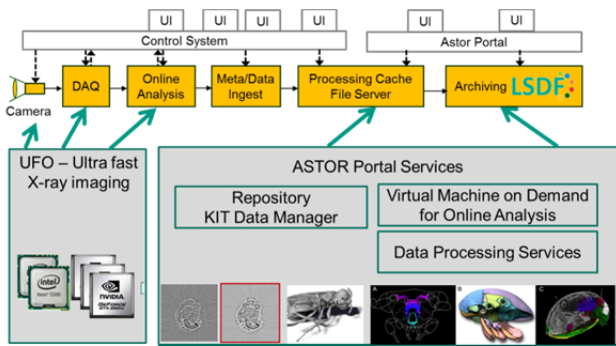


Figure 4: Astor Portal.

However users will always be able to take their data home, or download them using a web portal, to do offline analysis of their data. Data analysis results in new data. This newly created data will not be stored in the original dataset, but new datasets will be created and related to the original dataset. These new datasets will be connected inside of the database, as such duplicated storage will be avoided and the work can be easily reproduced.

DATA ARCHIVE

When a dataset will be ingested into the data repository, it can be marked as a long term dataset, even though the analysis has not been completed. This is feasible since the data will not change any more. However the user responsible for this data has to decide, for how long it will be archived if at all. Especially users coming from an industrial background might not want their data stored at a location outside of their own control. People with an academic background might want to make their data available for the public after a certain amount of time, when the initial publications are completed. At the end of the agreed archive time the data will be automatically deleted, if no new contract is negotiated. We currently create an online portal, where the user can log on with their credential and then have access to their data, either for download or direct online analysis. Inside of this portal, it will also be possible to open the access for everybody after a specified time.

CONCLUSION

Parts of the presented approach are still in the evaluation phase. We still need to discuss appropriate business models. Furthermore some political questions have to be evaluated and resolved. User tend to keep all kind of data, however storage comes not for free. We would need a way to reduce the amount of archive storage needed. Maybe there will be some costs implied for certain storage and analysis hours. Some questions need to be answered, for example, what happened to the data if a proposer of the data changes the affiliation? Will this user's access be revoked? Or if new users enter the collaboration what are the legal steps to add new users into an existing group.

ACKNOWLEDGMENT

We want to thank the Federal Ministry of Education and Research [13] in Germany for the funding of our research within the project ASTOR.

REFERENCES

- [1] "NeXus A common data format for neutron, x-ray and muonscience", <http://www.nexusformat.org/>.
- [2] "HDF5 is a data model, library, and file format for storing and managing data", <http://www.hdfgroup.org/HDF5/>.
- [3] "ANKA the Synchrotron Radiation Facility at Karlsruhe Institute of Technology", <http://www.anka.kit.edu/>.
- [4] V. Mauch et al., "OpenGL-Based Data Analysis in Virtualized Self-Service Environments", Proc. PCaPAC2014, <http://jacow.org/>.
- [5] "Umbrella", <https://www.umbrellaid.org/euu/>.
- [6] "German National Research and Education Network, DFN", <https://www.dfn.de/en/>.
- [7] J. Chaize, A. Goetz, W.-D. Klotz and J. Meyer, "TANGO – an object oriented control system based on corba", in International Conference on Accelerator and Large Experimental Physics Control Systems, 1999.
- [8] "Siemens SCADA System Simatic WinCC Open Architecture", <http://w3.siemens.com/mcms/human-machine-interface/en/visualization-software/scada/simatic-wincc/Pages/default.aspx>.
- [9] "Persistent Identifier", <http://www.persistent-identifier.de/?lang=en>
- [10] J. Pokorny, "NoSQL databases: a step to database scalability in web environment", in Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services, New York, 2011.
- [11] "Amira 3D Analysis Software for Life Sciences", <http://www.vsg3d.com/amira/overview>
- [12] "VG Studio MAX", <http://www.volumegraphics.com/products/vgstudio-max/>.
- [13] "BMBF", <http://www.bmbf.de/en/index.php/>.

PROFIBUS IN PROCESS CONTROLS

M. Clausen, T. Boeckmann, J. Hatje, O. Korth, J. Penning,
H. Rickens, B. Schoeneburg, DESY, Hamburg, Germany

Abstract

The cryogenic installations on the DESY campus are widely distributed. The liquid Helium (LHE) is produced in a central building. Three cryogenic plants are installed. One is in operation for FLASH the other two are currently in the commissioning phase and will be used for the European XFEL. Thousands of I/O channels are spread over the campus this way. The majority of the I/O devices are standard devices used in process control. The de facto standard for distributed I/O in process controls in Germany is Profibus. So it is obvious to use this bus also for cryogenic controls. Subsequently we developed also special electronics to attach temperature and level readouts to this field bus. Special diagnostic tools are available and permanently attached to the bus. Condition monitoring tools provide diagnostics which enable preventative maintenance planning. Specific tools were developed in Control System Studio (CSS) which is -the-standard tool for configuration, diagnostic and controls for all cryogenic plants. We will describe our experience over the last years with this infrastructure.

THE USAGE OF FIELD BUSES

Cryogenic installations for liquid Helium are typically widely distributed. Starting from one or more central helium liquefiers the helium is typically distributed on side by long transfer lines. All of the process I/O is installed along these lines.

It is obvious that the signals may not be connected directly to the process computer. The cable length would exceed technical limits for noise reduction or may cause grounding problems.

Starting with the HERA helium plant we introduced field buses for distributed I/O. Two decades ago we used SEDAC. SEDAC is an in-house standard at DESY. It may be used for distances as long as one kilometre. The disadvantage of an in-house solution is that there's no commercial support and any kind of signal conditioning must be developed and built in house.

In the 1990th it was decided that DESY cryogenic control should go more commercial. As a basis the CAN field bus was chosen. This provided access to various I/O modules with several different signal conditioning types including all of the basic types for process controls like: 4-20mA(in/out) and relays(in/out).

PROFIBUS IN PROCESS CONTROLS

The next level of signal conditioning came into play for the European XFEL. One basic decision was to make use of the state of the art intelligent I/O controllers like pressure transducers and valve positioners. New

controllers provide intelligent local diagnostics which is useful e.g. for valve controllers. In addition they can be configured 'on the fly' through the field-bus which is useful e.g. for pressure transducers – to change the operation range.

Two families of devices came into play: Devices for ProfiNet and those for Profibus. At that time it was not clear whether ProfiNet components would actually make their way into the I/O business or whether ProfiNet would more likely form the connection between PLCs and supervisory controls.

The decision in favour of Profibus proved to be the right one. Up to now no significant number of intelligent I/O devices found their way into the ProfiNet market.

CUSTOM PROFIBUS DEVICES

In cryogenic controls it is not sufficient to rely on the basic I/O signal conditioning. An additional family of cryogenic specific signals must be supported:

Low Temperature Readout

Temperatures below 10K are typically read out with special sensors. Small measurement currents help to avoid heating of the sensor itself. This introduces a high noise to signal ratio. Special custom readout electronics are necessary to achieve high accuracy and a good reproducibility.

Helium Level and Heater Control

Reading helium levels with a thin resistive wire is not an easy task. The measurement current must be high enough to run the wire into normal conductivity. On the other hand the current must be supervised to avoid damage when the vessel is empty or even under vacuum.

Fig. 1 shows the I/O modules which are connected by an internal CAN bus to the Profibus coupler.



Figure 1: Custom I/O: Low Temperature and Level- / Heater Control.

CONFIGURATION

All Profibus I/O has a specific way how the data is represented in the Profibus address space. This kind of information is stored in so called GSD (Geräte Stamm Datei) files. Commercial configuration tools read these GSD files in and provide the possible options to the user. The final configuration is written to system specific binary files. These files are used in PLCs or process controllers to map all of the Profibus nodes and their local I/O to the local address space.

In the DESY/ XFEL case we are using EPICS as our process control system and the operating system is VxWorks. It is obvious that the binary files will not work in this configuration. So we decided to create our own configuration tool which generates OS independent XML files. All of our configuration tools are developed as Eclipse plugins written in Java. They form the CSS (Control System Studio) toolkit together with all the applications in operation in the control room.

Hardware Layout

The CSS plugin is called I/O configurator and it acts in a similar way like the commercial configuration tools. It reads the GSD files and offers all of the possible Profibus devices to the user. By picking and combining the individual devices the user can configure a complete Profibus segment. Each segment is connected to an individual Profibus card in the EPICS IOC (Input Output Controller).

Software: The Epics Database

Similar to the Profibus configuration we developed a configuration tool for the so called EPICS 'databases'. These databases are ASCII files which specify the instantiation of records which in turn define the processing and dataflow. The DCT (Database Creation Tool) supports the creation of hierarchical record structures. This helps defining record structures of the same kind.

SIMULATION

Wouldn't it be nice to test the control software before the equipment is installed in the field? We found this to be true and valuable. Being able to test the control software with simulated I/O is extremely useful in two cases:

Changing Existing Configurations

Following the requirement that the existing control software must be significantly modified, but the time for testing and commissioning is very limited, we decided to introduce a simulation system. The system simulates all Profibus devices on the Profibus segment. All signals provide the correct signal type and reasonable signal levels. The system is called WinMod and is running on PC hardware. It requires specific Profibus controller boards. The system is configured by an XML file. The file is also generated by the I/O configurator plugin. No specific configuration is necessary.

Following this approach it was possible to reduce the overall commissioning time significantly.

Testing New Configurations

The first steps for new systems are to check whether all of the signals are correctly configured. Second the control loops must be checked and third the state notation or sequence programs must be controlled.

All of these steps can be run through by means of a system which simulates the connected I/O. In this case the 'original' software on the IOC may be used without changes specific to the simulation mode. Depending on the steps which shall be run it will be necessary to add more functionality to the simulation system. This may also cover control loops or logic states of equipment.

OPERATION

Once a Profibus segment is in operation – it is running very stable. But it may take some time until this point is reached. Typical problems are: Cable end resistors which are not connected or at the wrong location; configuration of fibre optical link modules; configuration of nodes where the actual memory layout does not match the one configured in the Profibus master (here: the IOC).

The advantage of operating Profibus I/O is the predictable update rate which is constant and can be calculated based on the bus parameters. This can be taken into account for control loops. The process engineer must be aware of the fact that equipment behind a DP/ PA gateway is updated at a lower (fixed) rate.

Existing XFEL Profibus Networks

The Profibus networks are separated into segments. This is designed based on logical units (signals grouped for one subsystem) and/ or technical specs (the number of nodes and bytes supported in a Profibus segment). See Table 1. The number of devices is typically a factor of 10 higher than the number of nodes. The number of individual signals is another factor of ten higher.

Table 1: XFEL Profibus Segments and Nodes

System	Segments	Nodes
AMTF	7	310
XFEL Linac	13	~500

REDUNDANCY

Cryogenic systems are typically operated in 24/7 mode. This requires special precaution for the layout of the whole system [1] including:

- Cryogenic Hardware
- Process Controller
- Computer Networks
- Field Buses
- Electric Power

Redundancy should be introduced wherever reasonable and adequate. E.g. redundant temperature sensors are adequate redundant flow meters aren't.

Redundant Profibus Links

The backbone Profibus link within the cryogenic building consists of multimode fibre optic cables and the associated OLMs. The topology forms a ring. This setup guarantees continuous operation even with one damaged connection in the backbone. To span the distances in the XFEL tunnel it is required to use single mode fibre. Again we will establish a ring topology to avoid single points of failure in the backbone link.

Redundancy in DP/PA Gateways

Profibus equipment connected to the PA gateways is powered by the gateway through the Profibus cable. This adds a critical functionality to the gateway power supplies. Therefore these power supplies a typically redundant throughout the system.

Redundant Process Controller

Besides redundant power supplies attached to redundant UPS systems we also introduce redundant process controllers for all critical cryogenic components.

This adds a certain level of complexity to the system because Profibus does not support such an approach 'out of the box'. Two aspects are obvious: Profibus does not support multiple master systems. In addition it is not supported to switch mastership between nodes with different Profibus IDs. As a result this would cause the whole bus to run a reset/restart sequence.

To overcome these problems a new firmware for the Profibus cards in the process controllers was necessary. In this case the firmware is provided by the company Softing which also produces the cPCI (Compact PCI) Profibus cards.

DIAGNOSTICS

Several levels of diagnostic tools are available for Profibus from the supervisory channels in the control system down to the hardware level.

Diagnostic Channel in EPICS

Each Profibus node and each OLM is identified by a logical channel in the process control system (EPICS). Each channel defines the health state of one of these nodes. The channels are displayed on synoptic displays used by operators and the on call shift.

Diagnostic Equipment on the Profibus

Special diagnostic equipment on the Profibus can be directly attached to Ethernet. The online status information of the Profibus segment is displayed on individual web pages (see Fig. 2). The Ethernet connection is typically routed to the diagnostic network to stay off the production network with diagnostic traffic.

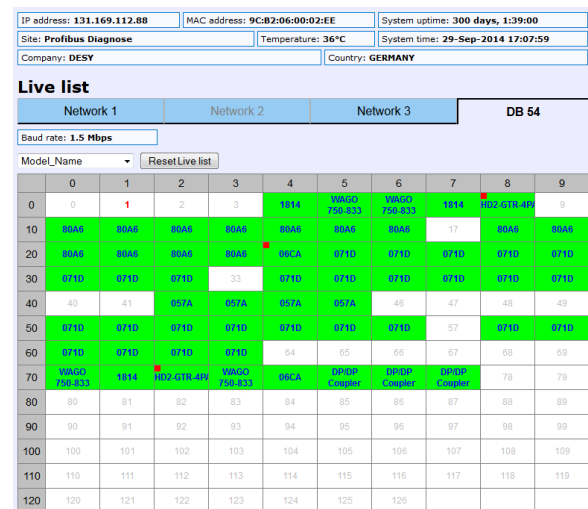


Figure 2: Online Profibus diagnostic web pages

Hardware Diagnostics

In addition to the online information on web pages it is possible to connect diagnostic equipment directly onto the Profibus cable. This allows dedicated diagnostics on the signal level on the protocol level. The Profibus segments at DESY are prepared to provide Profibus connectors for this purpose.

CONCLUSION

Profibus has become a stable and reliable 'partner' at DESY for the cryogenic control system. It is used for all current cryogenic projects for about a decade. Special firmware in the Profibus cards is mandatory for redundant process controllers. Permanent online diagnostic tools help to identify problems on the fly.

REFERENCES

- [1] T.Böckmann, "Profibus bei 2 Kelvin – Supraleitung für die Forschung mit Photonen", PI Conference, March 2013

A PROTOTYPE DATA ACQUISITION SYSTEM OF ABNORMAL RF WAVEFORM AT SACLA

M. Ishii[#], M. Kago, JASRI/SPring-8, Hyogo 679-5198, Japan
 T. Fukui, T. Maruyama, T. Ohshima, RIKEN/SPring-8, Hyogo 679-5148, Japan
 M. Yoshioka, SPring-8 Service Co., Ltd., Hyogo 679-5165, Japan

Abstract

We developed a data acquisition (DAQ) system for abnormal RF waveforms at the X-ray Free Electron Laser facility, the SPring-8 Angstrom Compact Free Electron Laser (SACLA). When a problem occurs, we must diagnose the source quickly. For this purpose, we developed a system that captures an abnormal RF waveform when a problem occurs in an RF system, and stores the waveform data in a database. The system consists of the VME systems, a waveform server, and a NoSQL database system, Apache Cassandra. When the VME system detects an abnormal RF waveform, it collects all related waveforms of the same shot. The waveforms are stored in Cassandra through the waveform server with shared memory as cache to complement Cassandra's eventual consistency model. We constructed a prototype DAQ system with a minimum configuration and checked its performance. In this paper, we report the scheme of the waveform DAQ system and the test results.

INTRODUCTION

SACLA has been operating for user experiments since March 2012. To maintain the scheduled user time as much as possible, we must reduce down time due to failure. To diagnose a failure source, it is quite helpful to collect the data of many accelerator components.

We have been installing two types of database systems since the beginning of commissioning. One is a data logging system with a cycle of several seconds, which provides diagnosis for slow fluctuation, such as environmental temperatures, the flow of cooling water, and the receiving voltage from the electric cubicles. The data are stored in a Sybase relational database management system [1]. Another database system is an event-synchronized data acquisition (sync-DAQ) system that collects beam currents, beam positions, and the phase and amplitude of the RF cavity pickup signals in synchronization with the beam operation cycle at the current maximum of 60 Hz. Shot-by-shot data are tagged with a master trigger number to identify the beam shot to which the data belong. The master trigger number is generated by counting a master trigger signal distributed from the master oscillator system. Shot-by-shot data with trigger numbers are stored in the MySQL relational database management system [2]. In addition, the DAQ system collects RF waveform data every 10 minutes. At the collection, however, it is

difficult to catch an abnormal RF waveform from a rare failure event that may occur only a few times a day. If we can catch the abnormal RF waveform, it is very helpful to analyze the phenomenon, because it has more information than point data, which is sampled from a waveform. Therefore, we developed a DAQ system to capture abnormal RF waveforms, an abnormal WFM-DAQ system. As a first step, we constructed a prototype system at a test stand.

LOW-LEVEL RF CONTROL SYSTEM

The linear accelerator of the SACLA comprises an electric gun, a 238-, 476-, 1428-MHz multi-sub harmonic bunching and accelerating system as an injector, eight S-band accelerating structures, and 128 C-band accelerating structures and in-vacuum undulators beamlines [3]. The low-level RF (LLRF) system controls the 74 RF units in the SACLA accelerator. A VME system, used at each RF unit, consists of a CPU, a trigger delay unit (TDU), a DAC and three ADC boards. The TDU board delivers the delayed trigger signal to the ADC and DAC boards, and generates the master trigger number by counting the master triggers. The ADC and DAC boards running 238-MHz clock detect the phase and amplitude signals generated by klystron, and generate signals of the IQ modulator for klystron input.

In the VME system, many processes are in operation including an equipment management process (EM), a data logging process, a Sync-DAQ process (SYNQDAQ-EMA) in synchronization with the beam operation cycle, and a feedback process (PID-EMA) for the stabilization of the phase and amplitude of the RF cavity with 100-ms sampling intervals [4].

SCHEME OF ABNORMAL WFM-DAQ SYSTEM

The abnormal WFM-DAQ system consists of a VME system, a waveform server, and Apache Cassandra, which is a key-value database system. Figure 1 shows a diagram of the abnormal WFM-DAQ system.

Detection of an Abnormal Waveform and Transfer of Data

The ADC board has four channels and four-memory banks that can store 512 waveform data in synchronization with a trigger signal. In addition, the ADC board generates an interrupt signal when it detects an abnormal waveform by comparing it with a reference waveform. When the sampled waveform exceeds a defined tolerance

[#] ishii@spring8.or.jp

of the reference waveform, the board sends an interrupt signal to the CPU board [5, 6].

We developed an event-driven process, ALM-EMA that detects an abnormal waveform, and collects all related waveforms at the same shot [7, 8]. ALM-EMA treats 12 ADC channels. Each channel is assigned a human-readable signal name, such as “xfel_llrf_cb01_1_iq_acc_1_dload_ta_q/waveform_err”. To prevent unnecessary abnormal detection at a start-up, an operator issues a start command of abnormal detection from the operation GUI with a stable RF condition. When ALM-EMA receives a start command, the process sets a normal waveform as a reference waveform and transfers the reference waveform to Cassandra through a waveform server. When an ADC board detects an abnormal waveform, the board issues an interrupt signal to a CPU. ALM-EMA performs the following actions.

- By receiving the interrupt signal, ALM-EMA disables abnormal detection and takes meta-information, such as a timestamp, master trigger number, bank number to which the waveform data are written, and the address point of the memory area to which the abnormal waveform data are written in the bank.
- 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 previous waveforms and following waveforms of the abnormal one on memory.
- ALM-EMA sends each waveform with meta-information to a waveform server and receives a reply message from the server.
- ALM-EMA enables the abnormal detection and waits until receiving a start command from the operation GUI.

We defined three types of messages for transfer to the waveform server. One is a signal registration message including a signal name, sampling number, and one point data size of waveform. Another is a waveform data transfer message, including a signal name, a timestamp, master trigger number, error flag that indicates whether the waveform is normal or abnormal, and waveform data. The third message type is a replay message from the

waveform server.

A Database System

Cassandra has features such as a column-oriented data structure, high write performance, fault tolerance, and no single point of failure (SPOF). It is especially easy to increase total throughput by adding more nodes to the system [9]. We designed the data structure to efficiently handle time series, trigger numbers and error flags. Figure 2 shows the data structure. 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 contains collections of columns. One column consists of a name and a value. A name is formed from the timestamp in addition to a key word such as “trig”, “err”, or “wfm”.

Cassandra is possible to elicit high performance by parallel programming. However, the parallel programming requires many steps in some programming languages. We developed a Web service framework, implementing a simple MapReduce model for reading waveform data by the parallel processing from Cassandra [9].

Cassandra can construct multi node clustering to achieve a redundant and load-balancing system. From our study of the Cassandra cluster, when the data are taken from a cluster with six nodes and a replication factor of three, we found that the time required for guaranteeing consistency is as much as 1-s [10]. To prevent this inconsistency, we developed a cache server mechanism.

DATA	
row-key: "xfel_llrf_sb_1_iq_acc_2_dload_ta_i/waveform_err:20140918"	
Column name	Column value
1411012800200100:trig	12345678
1411012800200100:err	1
1411012800200100:wfm	Waveform data (binary)
....

row-key: "xfel_llrf_cb01_1_iq_acc_1_dload_ta_q/waveform_err:20140918"	
Column name	Column value
1411012800401760:trig	12345678
1411012800401760:err	0
1411012800401760:wfm	Waveform data (binary)
....

Figure 2: Cassandra's data structure.

Waveform Server

A waveform server consists of a waveform handling process (wfm-handler) and a waveform readout process (wfm-deliver) with two kinds of shared memory as a cache. A wfm-handler writes waveform data received from ALM-EMAs to shared memory and Cassandra in parallel. We can flexibly increase the number of the wfm-handler to distribute accesses from 74 LLRF VME systems. When a wfm-handler receives a signal registration message, the process writes signal information on shared memory (comshm). All signal information is consolidated on comshm. When a wfm-handler receives waveform

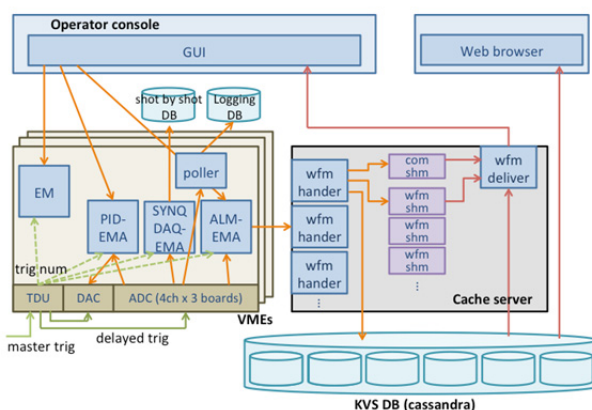


Figure 1: A diagram of abnormal waveform DAQ system.

data, the process writes the data on shared memory (wfmshm) for waveform created each signal. wfmshm keeps data for 2-s from the current time to complement Cassandra's eventual consistency.

A wfm-deliver takes waveform data from wfmshm or Cassandra in accordance with a timestamp that the operation GUI requires, and transfers the data to the operation GUI.

TEST

We constructed a prototype system with a minimum configuration to ensure the performance of the LLRF VME system, a waveform server, and Cassandra at an RF test stand. The VME system was set up with the same configuration of LLRF control of SACLA. The Cassandra cluster had three nodes and a replication factor of two.

It took 1.1-ms to take 16KB of waveform data from an ADC channel at our measurement. The round trip time to transfer waveform data from the VME to Cassandra through a waveform server and to the receive reply message was 10-ms. To prevent blocking VMEbus access from PID-EMA, ALM-EMA took each waveform data at a 500-ms interval. The CPU load of ALM-EMA was less than 1%. Therefore ALM-EMA does not prevent other processes.

Figure 3 shows an example of an abnormal waveform detection of a C-band cavity with a 30-Hz operation cycle at the RF test stand. The WFM-DAQ system detected an abnormal waveform shown by the red line at event number 714639. The previous waveform (event number = 714638) and the following waveform (event number = 714640) were almost consistent with the reference waveform. These shot-by-shot data were stored in Cassandra. At that time, worsening of the vacuum caused an abnormal waveform of the klystron forward RF signal. The prototype WFM-DAQ system was able to capture the sudden abnormality.

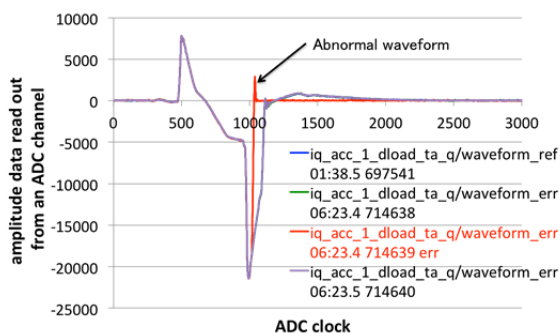


Figure 3: An example of an abnormal waveform detection at the test stand. The red line is an abnormal waveform, and the blue line shows a reference waveform. The green and purple lines are the previous and following waveforms of the abnormal one, respectively. The vertical axis is amplitude data read out from an ADC channel. The horizontal axis is a clock number.

INSTALLATION

In September 2014, we installed the WFM-DAQ system into the injection part of the SACLA configured by five LLRF VME systems. The construction of the Cassandra cluster is six nodes and a replication factor of three. We are currently developing the operation GUI. We will start waveform DAQ in November 2014 and finally will take waveform data from 74 LLRF VME systems in 2015.

SUMMARY

We developed a DAQ system for abnormal RF waveforms. The system captures an abnormal RF waveform that suddenly occurs, and stores the waveform data in Cassandra. A prototype system was checked at the test stand. The prototype system worked successfully. We started to install the system into the injection part of LLRF control system in September 2014. The collected data will help to improve the stability of the accelerators.

REFERENCES

- [1] R. Tanaka et al., "Inauguration of the XFEL Facility, SACLA, in Spring-8", Proc. of ICALEPCS2011, p.585-588
- [2] M. Yamaga et al., "Event-Synchronized Data-Acquisition System for SPring-8 XFEL", Proc. of ICALEPCS2009, p.69-71
- [3] Y. Otake et al., "Overview of SACLA Machine Status", Proc. of LINAC2012, p.427-431
- [4] H. Maesaka et al., "Recent Progress of the RF and Timing system of XFE/SPring-8", Proc. of IRCAL-EPCS2009, p.85-89
- [5] 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, p.02.050-4
- [6] Y. Otake et al., "SCSS RF Control toward 5712 MHz Phase Accuracy of One Degree", Proc. of APAC2007, p.634-636
- [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. Kago et al., "Development of a Scalable and Flexible Data Logging System Using NoSQL Databases", ICALEPCS2013, p.533-535
- [10] 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

UPGRADE OF SACLA DAQ SYSTEM ADAPTS TO MULTI-BEAMLINE OPERATION*

Toshinori Abe, Yukito Furukawa, Takaki Hatsui, Yasumasa Joti, Takashi Kameshima, Takahiro Matsumoto, Kensuke Okada*, Takashi Sugimoto, Ryotaro Tanaka, Mitsuhiro Yamaga, JASRI/SPring-8, Hyogo, Japan
Makina Yabashi, RIKEN Spring-8 Center, Hyogo, Japan

Abstract

This paper presents details of a major overhaul of the DAQ (data acquisition) system for user experiments at SACLA (SPring-8 Angstrom Compact Free Electron Laser [1]). The DAQ system has been providing a common experimental framework to various SACLA users since March 2012.

In 2014, SACLA introduced a third beamline to increase the capacity of experiments that can be performed. With respect to the DAQ system, it is a challenge to operate multiple experiments simultaneously. To ensure that the increased capacity of experiments can be handled, the network architecture was redesigned so that controls and data streams are made independent. A new tag supply system, which guarantees reliable synchronization, was implemented. In addition, a 90 TFLOPS supercomputer was installed to meet the growing demand for offline analyzing power.

INTRODUCTION

At SACLA, the DAQ system is required to store shot-by-shot information synchronized with an X-FEL (X-ray Free Electron Laser) beam of 60 Hz at maximum repetition rate. The data throughput goes up to 6 Gbps with images (e.g., X-ray diffraction images) obtained from twelve sensors of an MPCCD (multiport charge-coupled device [2]). The data are stored in a hierarchical storage system capable of more than 7 PB at the last stage. The DAQ system incorporates prompt data processing performed by a 14 TFLOPS PC cluster.

For multi-beamline operation, the control and data streams are duplicated and separated for the beamlines, i.e., they are made independent. In addition, the architecture of the control line is reformed to reduce risk of mishandling. The new tag supply system implemented for synchronization has been operating stably. In the offline part, a 90 TFLOPS supercomputer was installed to boost data analysis capacity.

In the following section, first, we provide an overview of the system; then, we describe recent major upgrades. Further, we briefly discuss features to be implemented in the near future, and finally provide a summary.

DAQ OVERVIEW

The SACLA DAQ system has been described in detail elsewhere [3]. The DAQ system consists of online and offline parts, which have been described in the following sub-sections.

Online Part

Figure 1 shows a schematic view of the online part of the SACLA DAQ system. Image data are transferred over the 10-gigabit Ethernet to a cache storage system, which has multiple writing servers to handle the high throughput. In the data stream are the data-handling servers, which can analyze image data on time. One of important functions on the data handling server is sending live view images to user terminals for the monitoring purpose. Many other small-sized data such as photo diode amplitudes of beam monitors and pulse motors' position of various instruments are stored in the database. The database is also used to store experimental conditions. Because at SACLA, the X-FEL target specimen is likely destroyed by a single shot and certain experiments are sensitive to a slight variation of the beam characteristics, the DAQ should be able to recall and provide the information related to each shot. To this end, all components of the DAQ are synchronized with X-FEL beam shot and data are stored with a unified tag number assigned to each shot.

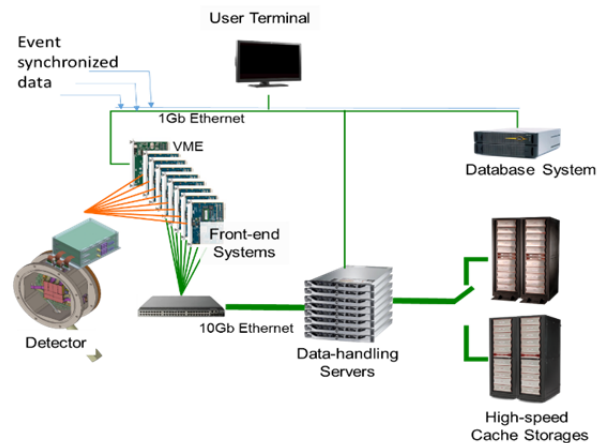


Figure 1: Online part of the SACLA DAQ.

From September 2013 to July 2014, 300 TB of image data were stored through the cache storage, and two cache storages were switched just once.

Offline Part

Figure 2 shows a schematic view of the offline part of the SACLA DAQ system. Through the offline part, SACLA provides a platform for analysis. Users can access the data stored in the cache or in the archive system along with various parameters stored in the

database. The HDF5 format is selected as the common data format for users, and a translator that translates the experimental format into the HDF5 format is provided. A computing farm with many nodes is set up for data analyses. Users use a job scheduler to submit their tasks. A part of the system newly installed in 2014 is explained in the next section.

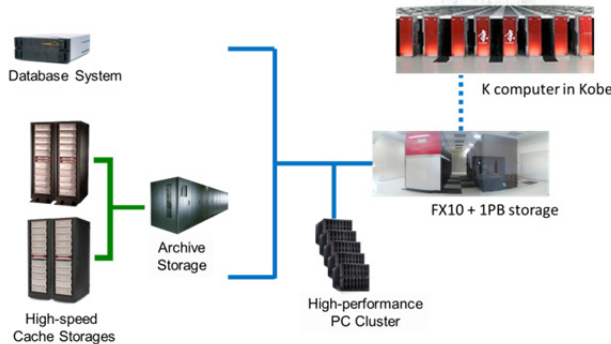


Figure 2: Offline part of the SACLA DAQ.

Monitoring

Online processes are monitored by a custom-made monitoring system. For example, in case the shot-tag counter on one subsystem malfunctions, it sets off an alarm.

UPGRADES

In this section, recent major upgrades made to the SACLA DAQ system are presented. The new tag supply system increases the reliability of event synchronization. The network architecture is redesigned to maintain the data throughput during multi-beamline operation and for the new centralized control system. A supercomputer system is installed to meet the increasing demand of offline analyzing power.

Tag Supply System

Each subsystem uses the common tag number associated with a single X-FEL shot. Before the introduction of the tag supply system, a local count system was used, which occasionally lost the correct count. The new system is based on a tree structure, and a tag-data-master delivers the tag number associated with an X-FEL shot to all subsystems. The tag supply system, which has been described in detail in [4], has been working stably since April.

Network Architecture

To handle multi-beamline experiments, the network segments were reformed. Figure 3 shows the new design of network segmentation. The large-volume (image) data streams to the cache storages are physically separated between beamlines to achieve the peak throughput independently. However, they are merged at the stage of archiving so that the offline analysis has one single gateway.

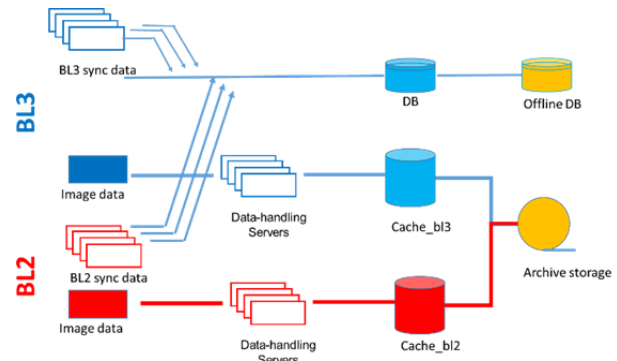


Figure 3: New network segmentation design based on the concept of a multi-beamline DAQ system. The large-volume (image) data stream is assigned a separate network segment for each beamline.

The other goal of this overhaul is related to the control line to build an architecture that prevents accidental modification of critical DAQ settings by end-users, such as timing offsets or interference with other beamline users. In the original system, accidental modification of critical DAQ settings was possible because anyone could freely access operator consoles in the DAQ segment. In the new architecture shown in

Figure 4, the control scheme is secured by the VLAN settings for beamlines and access groups. The BL-master server is the hub of all commands to each component, where the access control is applied in a centralized way. For example, it is possible to create a list of motor axes limited only for facility use. The messaging scheme is based on MADOCA II [5] [6] framework. To hop more than one message server, MADOCA II has implemented its extension.

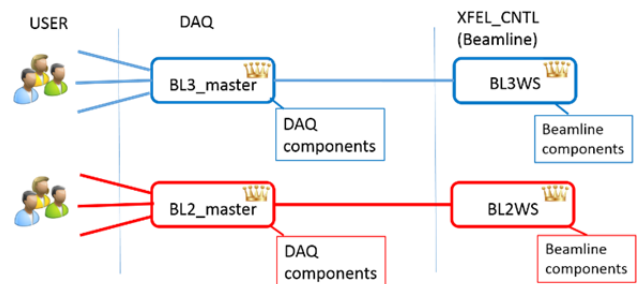


Figure 4: The separation of beamlines and access groups is secured by the VLAN settings. The access control is centralized at BL-masters and BLWSs.

FX10 System

As experimental technique get sophisticated and the user time is increased with the new beamline, more computing power is required for analysis. In March 2014, a 90 TFLOPS supercomputer was installed in the campus. The system consists of 384 computing nodes of the "FUJITSU PRIMEHPC FX10" system, and storage

systems of 500 TB, 100 TB, and 1 PB for global, local, and external use, respectively. Its CPU architecture was chosen to match with the K computer [7], which is a 10 PFLOPS machine at RIKEN AICS, located 100 km away from the SACLA site. Therefore this system at the SACLA site can be used to develop analysis routines, before executing more computationally heavy analysis on the K computer.

In this regard, data transfer speed is a matter of concern. Using a simulated data set, we tested the speed of data transfer to RIKEN AICS. Figure 5 shows the results of our test. In addition, we emulated a whole analysis chain including a job submission to the K computer's batch system using 800 GB of simulated data. Though the performance of the whole chain depends on the number of available nodes and the type of analysis, a single model case [8] related to one experiment (say, with 20 TB of experimental data) can be analyzed in a reasonable time (a day or so).

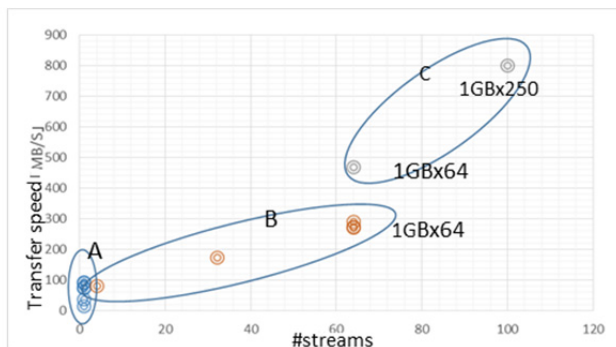


Figure 5: Speed of data transfer from SACLA to HPCI storage [8][9] at RIKEN AICS as a function of the number of parallel streams. A: gsiscp protocol (#stream=1), several encryption types; B: gfpcopy protocol with default setting; C: gfpcopy protocol after TCP window size tuning. The highest performance was achieved when 250 1 GB-sized files were transferred using 100 parallel gftp streams.

FUTURE CHALLENGES

In this section, we briefly discuss topics to be considered for future operation.

Currently, the volume of the cache storage is sufficient to store data for a certain amount of time such that the data can be migrated to the archive system manually. However, we expect that the data accumulation rate will increase as experiments become more sophisticated. During the implementation of the next set of replacements, we are considering the introduction of a staging mechanism wherein the migration is built-in, such as a file system.

In 2015, SACLA plans to start a operation mode with rapid beamline switching in any intended pattern. In addition to the ability of multi-beamline operation discussed earlier, at least the path of the beam needs to be recorded shot by shot. A better option for the distribution

may be in a manner similar to the tag number so that detectors can be prepared for every shot.

At the operation side, we expect to schedule more beam time to industrial users, where demands for proprietorship exist. We have started discussing plans to implement proprietorship without loss of convenience.

Finally, a next-generation image sensor is in the development stage [10]. It requires data transfer speeds to be increased by an order or more; thus, this will be the next major upgrade of the DAQ system.

SUMMARY

SACLA provides a common DAQ platform including online and offline parts. In 2014, several upgrades have been applied in order to make the DAQ system adapt to multi-beamline operation. The new tag system is implemented for reliable synchronization. The network architecture is redesigned—the data stream is duplicated and provisions are made to allow the addition of another stream in the future. The control flow became more secure. The offline analysis part is improved by the addition of a new 90 TFLOPS supercomputer.

REFERENCES

- [1] T. Ishikawa et al., “A compact X-ray free-electron laser emitting in the sub angstrom region”, *Nature Photonics*, vol. 6, pp. 540-544 (2012).
- [2] T. Kameshima et al., “Development of an X-ray pixel detector with multiport charge-coupled device for X-ray free-electron laser experiments”, *Review of Scientific Instruments*, vol. 85, (2014) Art. Num. 033110.
- [3] 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).
- [4] T. Abe et al., “Development of New Tag Supply System for DAQ for SACLA”, *5th International Particle Accelerator Conference IPAC'14*, Dresden, Germany (2014).
- [5] R. Tanaka et al., “The First Operation of Control System at SPring-8 Storage Ring”, *Proceedings of ICALEPCS 97*, Beijing, China, 1(1997).
- [6] T. Matsumoto et al., “Next-Generation MADOCA for The SPring-8 Control Framework”, *Proceedings of ICALEPCS2013*, p. 944, San Francisco, CA, USA
- [7] K-computer website: <http://www.aics.riken.jp/en/>
- [8] A. Tokuhisa et al., “High-speed classification of coherent X-ray diffraction patterns on the K computer for high-resolution single biomolecule imaging”, *Journal of Synchrotron Radiation*, vol. 20, pp. 899-904 (2013).
- [9] HPCI website: <https://www.hpci-office.jp>
- [10] SOIPIX collaboration, Developments of SOI monolithic pixel detectors, *Nuclear Instruments and Methods in Physics Research Section A*, vol. 623(1) pp. 186 (2010)

SARDANA – A PYTHON BASED SOFTWARE PACKAGE FOR BUILDING SCIENTIFIC SCADA APPLICATIONS

Zbigniew Reszela, Guifre Cuni, David Fernandez-Carreiras, Jorg Klorá [on leave], Carlos Pascual-Izarra, CELLS-ALBA Synchrotron, Cerdanyola del Vallès, Spain
Tiago Coutinho, ESRF, Grenoble, France

Abstract

Sardana 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[1]. Sardana, thanks to the Taurus library[2], allows the user to build modern and generic interfaces to the laboratory instruments. It also delivers a flexible python based macro environment, via its *MacroServer*, which allows custom procedures to be plug in and provides a turnkey set of standard macros e.g. generic scans. Thanks to the *Device Pool* the heterogeneous hardware could be easily plug in based on common and dynamic interfaces. The Sardana development started at Alba, where it is extensively used to operate all beamlines, the accelerators and auxiliary laboratories. In the meantime, Sardana attracted interest of other laboratories where it is used with success in various configurations. An international community of users and developers[3] was formed and it now maintains the package. Modern data acquisition approaches guides and stimulates current developments in Sardana. This article describes how the Sardana community approaches some of its challenging projects.

SARDANA AND ITS COMPONENTS

Sardana is a distributed control system based on the client-server model. The communication protocol is Tango[4]. Different Sardana configurations are possible depending on the scale of the installation. As an example, a small laboratory could have a single Sardana server exporting one Device Pool and one MacroServer. If needed, multiple Device Pool servers could be distributed over different hosts. Configurations with many MacroServer servers are also possible. Multiple Graphical User Interface (GUI) and Command Line Interface (CLI) clients can communicate with the Sardana system, at the same time.

Device Pool

Scientific installations are characterized by multiple and heterogeneous hardware. The particle accelerators comprises power supplies, vacuum equipment, radio frequency stations, insertion devices and many diagnostics and actuators among others. The laboratories, e.g. a synchrotron beamline, usually consists of even more diverse instruments like, diffractometers, monochromators and sophisticated detectors full of moveable axes and experimental channels. These laboratories frequently require to modify their configuration depending on the experiment being performed. Sardana interfaces all the equipments via the

Device Pool and its plugin *controller* classes. Sardana elements could be classified by their types in three main groups:

- moveables: physical motors e.g. stepper or servo motors, piezo actuators and logical pseudo motors e.g. the energy, composed translations
- experimental channels: counters and pseudo counters; 0D – scalar values based on mathematical operations e.g. averaging or integration of samples; 1D – one dimensional detectors e.g. Multi Channel Analyzers, Position Sensitive Detectors; 2D – two dimensional detectors e.g. CCD cameras
- other elements: communication channels, enumerated scalars called I/O registers

The controllers group the Sardana elements, which are organized and identified by the axis number. All these elements are represented by TANGO devices, and could be spread in many Device Pool servers or grouped in a single one. The Device Pool optimizes common control processes. The grouped acquisitions are handled by the *Measurement Group*, which configures and synchronizes the experimental channels. The grouped motions are implemented inside the Pool and if the hardware allows that, motion of all the axes can be started simultaneously by one single command. The API of the controllers and the programmed algorithms take into account all these particularities so access to the hardware is optimized.

MacroServer

A basic requirement for the scientific SCADA system is to provide a sequencer capable to plan and execute experiment procedures. MacroServer, together with its *Doors*, provide these and other functionalities (via Door different client applications access the MacroServer engine). The MacroServer allows the execution of multiple procedures (called *macros*) sequentially or even simultaneously if different Doors are used. A macro can accept input parameters and return a result or produce data, which might be interchanged between chained or nested macro sequences. It is possible to interrupt the sequence execution at any moment as well as temporarily pause and resume it. Features for adding, editing and browsing the available macros are accessible via the MacroServer at runtime. Sardana provides a miscellaneous set of standard macros. Their naming and parameters often follow conventions adopted by SPEC[5] what optimizes the users learning curve. Probably the most useful and sophisticated macros are the generic, n-dimensional scans. The scan process consist of data acquisition of the involved experimental channels while varying the scanning variable (typically a moveable

object). Scans in Sardana are available in various modes: step, hybrid and continuous, where distinct actions like motion, data acquisition or 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 despatch them to the destination in the correct format e.g. a file, console output, a data post-processing program.

Taurus Based Human Machine Interfaces

Sardana client applications (CLI and GUIs) are developed with the Taurus library. Taurus was originally conceived for connecting client side applications to Tango device servers using the PyTango library. For the GUI part, Taurus is built on top of PyQt[6]. Taurus is designed with a Model-View-Controller (MVC) architecture and has currently transcended its Tango-centric origins by accepting *scheme* plugins. A scheme plugin provides the model objects for a certain source of data and/or of controllable variables. Schemes exist for several hardware access layers (Tango, Epics, SPEC) and even for interacting with a Python interpreter. Each model object has a unique URI based name and it can be transparently used by the higher levels of the library. This makes it possible to mix in one application variables from different control systems, values retrieved from the archiving systems or any other source of data.

Taurus comprises a complete set of widgets (forms, plots, macro execution and experiment configuration panels, etc.) which provide the View and Controller components of the MVC and which assure a standardized look-and-feel of the user interfaces. They are easy to program thanks to the user-friendly API and are even integrated in the standard graphical Qt designer. The TaurusGUI and its configuration wizard allows the creation of complete GUI applications with just few clicks. The GUI is typically configured with a synoptic view representing a number of instruments, and can be managed with different perspectives. All these features considerably speed up the application development process (to the point that users often create ad-hoc temporary GUIs to solve particular tasks).

Spock is a Command Line Interface built on IPython and makes extensive use of Taurus as well. Its main functionalities are macro execution and control of any Sardana element. Spock provides context tab completion and an easy access to the history of executed commands which makes it user-friendly and efficient.

CURRENT & FUTURE DEVELOPMENTS

Testing Framework

Many enhancements are currently being evaluated and developed. Some of them require refactoring of the Sardana and Taurus core concepts, which increases the risk of regression. The Sardana Testing Framework was developed in first order to mitigate this risk. The

framework provides guidelines regarding test organization, naming conventions and the required test case documentation. It comprises a base test case and test suite classes as well as tools to ease tests development. From now on, many new developments follow the test driven development process. As the next step, it is planned to set up the continuous integration service, which will enable the real benefits of the Sardana Testing Framework.

Continuous Scans

The Generic Scan Framework was recently extended by the continuous mode, however present solutions still lack of generalization on various levels. The future scans must provide transparency between the different synchronization modes of the experimental channels and external attributes involved in the acquisition process. The precise timestamp distributed over the involved hardware and hosts could be used for software triggering and data timestamping. A new element type – trigger/gate and a corresponding controller will be added to the Device Pool. Their interfaces must be determined to allow configurations of equidistant or arbitrary sequences of events in time or distance domains. Modern devices could deliver multiple functionalities e.g. one device could control a moveable axis, generate trigger signal and provide an experimental channel of the position measurement. The current design of the controller concept needs to be enhanced to allow handling elements of different types by one controller class. Usually the continuous scan requires the acquisition process to be performed while the scanning variable changes with a constant rate. In the case of the pseudomotors with non-linear formulas this require control of the motion trajectories of the involved physical motors. Current implementation approximates this cases and maintains the constant velocity of the physical motors.

Other Enhancements

A diffractometer is a common instrument in the synchrotron beamlines and is available in various geometries e.g. Eulerian 4-circle or 6-circle, kappa 4-circle etc. Experiments involving diffractometers are based on scans in reciprocal space, which require complex diffraction calculations. A base diffractometer controller class (psudomotor) uses the HKL library[7] and encapsulates all the complex calculations. A set of derived classes just defines the geometric specific physical and pseudo motor roles. In addition, any diffractometer instrument, could be controlled with the specific Taurus based widgets e.g. diffractometer alignment or hkl scan.

High speed and high resolution detectors and cameras require advanced control and optimized image post processing. The Lima library[8] fulfils these requirements. It was used to develop an early version of a 2D experimental channel controller. Its functionalities allow scans involving 2D experimental channels. Further

improvements are already planned e.g. to access the Lima objects directly instead of accessing them via Tango or to allow passing references to image data instead of the full raw images.

While some laboratories (such as Alba) use Sardana and Taurus as a complete scientific SCADA suite, other users are interested in using some components integrated in their running control system. The installation can be tailored and/or extended to fit different needs. Taurus and Sardana currently provides several extension points e.g. schemes, domain specific widgets, core extensions, macros, controllers, recorders, but these are implemented in different ways. A generic plugin system is one of the future goals which would not only make currently mandatory dependencies optional, but would also simplify and unify the extension of Taurus and Sardana.

SARDANA COMMUNITY

The origins of Sardana reach 2005, when early decisions about the ALBA Control System were taken. First the TANGO was chosen as the control system framework for ALBA. Common requirements of the beamlines sketched the first Sardana specification. The development process started immediately, and the core of the system was mostly maintained by one developer. Sardana was successfully commissioned during the installation and commissioning of the accelerators and beamlines which was completed in 2012. It grew to a mature product and other institutes, mainly synchrotrons, selected it as their control system. Sardana is free and open source (LGPL) which attracts many new users. The big interest of current and potential users together with many ongoing enhancement projects lead Alba to open also the project management by pushing for the formation of the Sardana Community in 2013. Sardana is currently used in DESY – Germany, MaxIV – Sweden and Solaris – Poland. These three institutes, together with Alba, form the actual collaboration. To a lesser extend, Taurus is also used at the ESRF and within Tango collaboration (by most official and unofficial participants). All these institutes actively participate in the community activities. Commercial support is given by the Nexeya and the Cosylab.

The first decision of the community was to formalize discussion process about the proposals of improvements and modifications. The Sardana Enhancement Proposal (SEP) process was introduced, and up to now 12 SEPs are defined (some of them already accepted). All of the Sardana repositories, the core and the third-party macros and controllers were migrated from SVN to GIT, which fits better to Sardana Community profile. The code contribution workflow, the branching model and naming conventions were defined. In order to provide the highest quality of the Sardana core code, all its contributions must pass through an open and transparent public code review

and integration process. Another organizational change, which is currently in progress, is the Taurus library separation. Sardana and Taurus codes were isolated and soon the Taurus code will be migrated to its own GIT repository and project. Sardana is hosted on the Sourceforge platform and uses a number of offered tools e.g. an issue tracker, mailing lists, wikis etc. They are widely used by the Sardana Community (both users and developers) in their daily life. The Sardana Workshop meetings, often organized as satellites to the Tango Meetings, take place once a year and the virtual conferences are organized according to the needs.

The Sardana and Taurus release cycle includes two official releases per year: in January and in July. The latest installers, for Linux and Windows platforms, are available to download from the PyPI repository or could be installed directly with pip or easy_install. Sardana and Taurus could also be installed directly from sources, which could be obtained from their GIT repositories, with the setup tools installation. Debian users have also access to the official debian packages.

ACKNOWLEDGEMENT

Many people have worked in this project. This is a work achieved with the effort of the whole ALBA controls group. The ALBA scientists took part in the specification refinement process, what has to be recognized, together with the beam time offered for the acceptance tests. Important contributions were received from T. Nuñez and J. Kotanski from DESY, F. Picca from Soleil and V. Valls, E. Taurel (who as the Tango expert guided the initial steps of the Sardana developments), A. Homs and V. Rey from the ESRF. The authors would also like to thank the experts from other institutes, for their valuable feedback and ideas, in particular to T. Kracht from DESY, N. Leclercq from Soleil, D. Spruce and A. Milan from MaxIV and P. Goryl and L. Żytniak from Solaris.

REFERENCES

- [1] T. Countinho et al. "Sardana, The Software for Building SCADAs in Scientific Environments", ICALEPCS2011, Grenoble, WEAAUST01
- [2] The Taurus library web page: <http://www.taurus-scada.org>
- [3] The Sardana project web page: <http://www.sf.net/projects/sardana>
- [4] TANGO web page: <http://www.tango-controls.org>
- [5] SPEC web page: <http://www.certif.com/spec.html>
- [6] The PyQt library web page: <http://www.riverbankcomputing.com/software/pyqt>
- [7] The HKL library web page: <http://www.synchrotron-soleil.fr/portal/page/portal/Instrumentation/EnvironnementInstrumental/hkl>
- [8] The Lima library web page: <http://lima.blissgarden.org>

A NEW DATA ACQUISITION SOFTWARE AND ANALYSIS FOR ACCURATE MAGNETIC FIELD INTEGRAL MEASUREMENT AT BNL INSERTION DEVICES LABORATORY*

M. Musardo[#], D. Harder, P. He, C. A. Kitegi and T. Tanabe
National Synchrotron Light Source II, Brookhaven National Laboratory
Upton, New York 11973-5000, USA

Abstract

A new data acquisition software has been developed in LabVIEW to measure the first and second magnetic field integral distributions of Insertion Devices (IDs).

The main characteristics of the control system and the control interface program are presented. The new system has the advantage to make automatic and synchronized measurements as a function of gap and/or phase of an ID. The automatic gap and phase control is a real-time communication based on EPICS system and the eight servomotors of the measurement system are controlled using a Delta Tau GeoBrick PMAC-2.

The methods and the measurement techniques are described and the performance of the system together with the recent results will be discussed.

INTRODUCTION

The National Synchrotron Light Source II (NSLS-II) at Brookhaven National Laboratory is a new 3 GeV electron storage ring of third generation designed to provide synchrotron radiation with a very broad energy range and an ultra high brightness and intense flux using advanced insertion devices [1]. In order to validate that IDs delivered to NSLS-II meet the tight specification, they are accurately surveyed before installation into the storage ring. Two benches, a Hall probe bench for local field measurement and the Integrated Field Measurement System (IFMS), have been purchased to magnetically survey the IDs [2]. The IFMS supplied by ADC USA Inc, is a set of three field integral measurement systems, a stretch wire bench, a moving wire and a flip coils. An upgrade program has been started; this article reports the first stage of the IFMS upgrade, focused on the flip coil system.

FLIP COIL MEASUREMENT SYSTEM

The measuring bench consists of a long coil of 10 turns of 38 AWG beryllium copper wire stretched between two granite blocks located on both sides of an ID. Each support includes three linear motorized stages to position and move the coil in the horizontal and vertical axes and a rotary stage employed to rotate the coil of 360° with an angular accuracy of 40 arcsec and encoder resolution of

about 0.005 deg. It is equipped with rotary encoders mounted to the vertical axis linear stage in horizontal orientation. The flip coil mounts to a special spool on the rotary axis located on each pedestal, it is looped between these spools to form one continuous loop. The width of the coil is 4 mm and the length is about 5 m. In order to reduce the coil sagging the longitudinal axis stage can be tensioned and adjusted manually by varying the zero position. A tensiometer placed in the bobbin give an indirect value of the gravitational sagging of the coil. The tension sensor is made by Omega Engineering and has a range of 100 pounds and a resolution of $\pm 0.2\%$ of full scale. All stages are assembled on the pedestals, which have an extremely flat tolerance on the top surface of about $\pm 10 \mu\text{m}$. Each pedestal has three leveling feet that also provide 10 mm adjustment in the horizontal X and longitudinal Z directions and 50 mm in the vertical Y direction. All linear axes have a Renishaw RELM linear encoder feedback with $0.1 \mu\text{m}$ resolution and $\pm 1 \mu\text{m}$ accuracy. The Y and Z axes have 150 mm travel and the X axis have 300 mm travel, which is sufficient to ensure a measuring range appropriate for each magnetic device. Each axis has a brushless DC motor operated in closed loop mode with limit switches and a home limit switch that is used to tell the motion controller that it is near the home pulse located on the encoder. The linear encoders on the motors provide velocity, acceleration and position feedback. The motion control system is based on a Delta Tau GeoBrick PMAC-2 controller, which is responsible for the coordinated motion of four pairs of motors in a master-slave configuration. A serial communication is used to control and synchronize the various hardware components of Delta Tau. Geo-Brick generates the spatial and temporal triggers and convert the analog tension signal to digital. It provides eight axis of servo control and has a position synchronized triggering on four axes, X, Y, Z and rotary. Delta Tau GeoBrick PMAC-2 is a fully programmable motion controller and several software parameters must be configured to properly control each motor.

A digital multimeter Keithley DMM 2701 is used to record sensitive induced voltage signals from the coil. Model 2701 is a $6^{1/2}$ -digit high-performance multimeter data acquisition system. It is equipped with a custom switching card to select the measurement source. Results of the DMM are transmitted to the host computer using a standard communication serial RS232 port. Keithley

*Work supported by DOE contract DE-AC02-98CH10886

[#]musardo@bnl.gov

provides a high rate of sampling with a good stability. The integration time used is 16.67 msec (1 PLCs), which is a good compromise between higher sensitivity and better spatial resolution. To help maintain a good stability and accuracy during the measurements a digital filter has been selected. It takes a selected number of reading conversions, averages them, and yields a reading. The filter is then cleared and starts collecting conversions all over again. An external trigger is provided by Delta Tau, the encoder signal is fed to the multimeter while the motor itself is controlled independently. There are two types of triggers planned, spatial and temporal, both are generated by the GeoBrick.

DATA ACQUISITION SOFTWARE

In order to perform accurate and precise synchronized measurements as a function of gap and/or phase adaptable for each device and to ensure that the integral multipole components are maintained within the specification requirements a new data acquisition software for the IFMS has been developed. This software has been written in LabView, which is a graphical programming platform from National Instruments.

The complete control of the measuring bench, such as homing, multi-axis positioning and movement have been implemented. The main features of the new software include the data acquisition and magnetic analysis, setting and reading of the gap and phase for any device and an user-friendly control panel to perform field integral measurements on the transverse plane in automatic gap and phase control.

Additional expert control panels are under development. They will be available for testing and configuration of the field integral measurement system, in particular some diagnostic tools will be necessary for the monitoring and tuning of the Delta Tau amplifiers. Further specific functionality will allow to view and manipulate basic system settings and controls with an extensive support for accessing to the instrumentation hardware.

The LabVIEW application software was developed on a PC which is the central control unit and operator interface of the measurement system. PC was configured to utilize a channel access for Experimental Physics and Industrial Control System (EPICS). It is used extensively in particle accelerators throughout the world and is the main control system used at NSLS-II; because it is run from UNIX based machines a Windows EPICS Client has been set up on the PC in order to have a completely control of a ID during the field integral measurements. The EPICS PC configuration allows the computer to issue directives to EPICS over the command line through Process Variables (PV). The channel access communication between LabVIEW and EPICS is created using the CaLab interface [3].

Flip coil system are commonly used to perform field integral measurements of insertion devices used in synchrotron light sources, where very strict tolerances on the magnetic field quality are required.

The first integrals of the transverse field components Bx and By along the longitudinal axis Z, defined by:

$$I_x = \int B_x dz ; I_y = \int B_y dz \quad (1)$$

are obtained from the induced voltage ε in the coil in according to the Faraday's law: $\varepsilon = -N \frac{d\Phi}{dt}$. Where Φ is the variation of magnetic flux linkage through the surface defined by the coil and N is the number of turns.

In addition to the usual point by point measurements, an important feature in this field integral bench is the capacity to perform "on the fly" measurements. This measurement technique consists to measure I_y and I_x by a continuously rotating coil from 0° to 360° around the z-axis at the starting and ending point followed by a translation movement at constant speed along the x-axis over the whole scanned range, with horizontal orientation to measure the variation of I_y and with vertical orientation to measure the variation of I_x . The field integral variations at the end of the translation movements are given by:

$$\Delta I_y = -\frac{\int V_y dt}{ND} ; \Delta I_x = -\frac{\int V_x dt}{ND} \quad (2)$$

where the voltage readings V_x and V_y are recorded by digital multimeter at constant steps and subsequently transferred to PC for processing. ΔI_x equation is valid only if the variation of I_x over a vertical distance equal to the coil's width D is small enough. During the measurement the coil is rotated forward about the longitudinal axis through the zero reference. At this point the measurement is triggered initially. The rotation continues and the measurement proceeds through 360° where the acquisition ends. At regular angular intervals the DMM is triggered electronically to collect the induced voltages and angular intervals. Once the forward measurement is completed, the coil is then rotated backwards so that data can be collected from the two different rotational directions and averaged. At the completion of the measurement cycle, the multimeter outputs the voltage at each encoder location. Assuming that variation of the magnetic field within the width of the coil is small enough, the voltage integrated over the measurement time $t_2 - t_1$ is given by:

$$\int_{t_1}^{t_2} \varepsilon dt \approx ND \left(I_y \int_{\theta_1}^{\theta_2} \sin \theta d\theta + I_x \int_{\theta_1}^{\theta_2} \cos \theta d\theta \right) \quad (3)$$

Therefore I_y is obtained by a rotation from 0° to 180° and I_x from -90° to 90° .

The full control of the measurement system is carried out by the main GUI illustrated in Fig. 1. This panel allows the user to initialize Delta Tau, enable and disable the server motors and to set up various system parameters such as the width of the coil, number of turns, linear and

angular speed. It also performs homing, positioning and movement in master-slave mode for each axis.

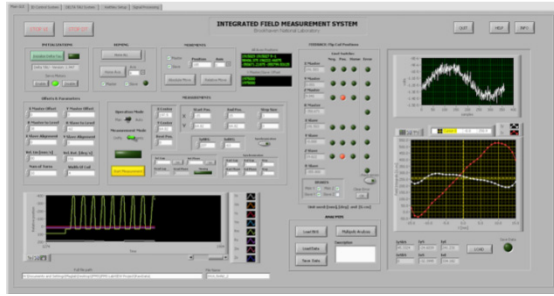


Figure 1: Main GUI interface.

Multiple scans can be taken at different locations in X or Y and the method of measurement (point by point or on the fly) can be selected. The GUI also allows access to offsets to align the slave to the master and the master to the beam line. The user can enter measurement information for data acquisitions such as measurement range and step, mechanical centers of the coil, background field integrals and the filename destination. The operation mode button allows the user to select measurements synchronized with gap and/or phase movements of an insertion device. After a measurement is completed, the data are analysed and graphed on the screen. The new software has been developed using a modular programming approach. This consists to separate the functionality of the software into independent and interchangeable modules, known as subVIs, each of which accomplishes everything necessary to execute only one specific aspect of the desired process. Figure 2 shows three subVIs, the "Integral RotCoil" (Fig. 2A) and "Integral MovCoil" (Fig. 2A) to calculate respectively the field integrals by rotation and translation movements of the coil and "OnFly Measurements" (Fig. 2C) to perform automatic measurements and data acquisition. It embeds a set of other SubVIs which are employed to control and synchronize the various hardware components in the measurement system.

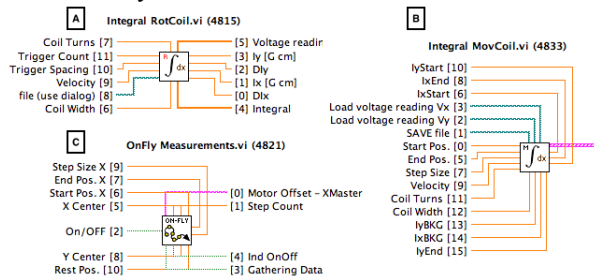


Figure 2: Main SubVIs.

The SubVIs in Fig. 3 provides a real-time monitoring of the global status of all motors on the system. Conditions on each axis such as position, home and limit switch, following error and amplifier fault error are checked and made visible on the main panel during the measurement process.

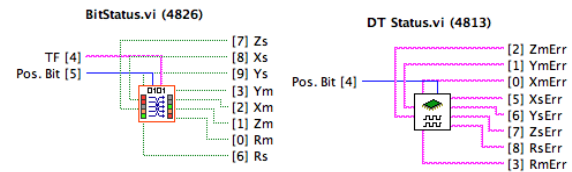


Figure 3: SubVIs for Delta Tau status.

Further control panels are available on other secondary tabs for testing, configuration and debugging of the motion controller, data acquisition system and IDs control system.

Figure 4 (left) shows the horizontal (red line) and the vertical (blue line) point by point field integral measurements versus the transverse position of a 1.5m IVU at 8 mm gap.

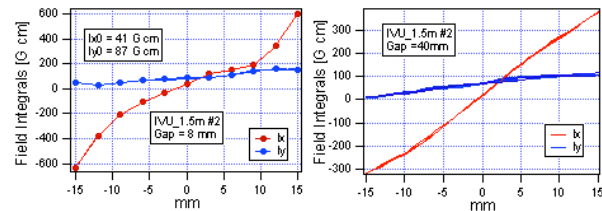


Figure 4: Field integrals of a 1.5m IVU.

In order to verify the accuracy and repeatability of the on-the-fly measurements 5 scans have been taken at 40 mm gap (Fig. 4 right). The standard deviation is 5 G cm for the on-the-fly measurements and 3 G cm for the point by point.

CONCLUSION

Although the insertion devices are very different, we strived to find a common design to make them look similar to the operator and to hide the different levels of complexity behind a common user-friendly interface.

ACKNOWLEDGMENT

The authors are sincerely grateful to Alexandra Valiton, Field Sales Engineer from National Instruments for support, assistance and cooperation and pleased to thank Carsten Winkler from Helmholtz-Zentrum Berlin for providing the LabVIEW-EPICS interface CALab 1.5.0.5.

REFERENCES

- [1] T. Tanabe et al., "The Latest Status of NSLS-II Insertion Devices", Journal of Physics, Conference Series 493 (2014) 012031
- [2] M. Musardo et al., "NSLS-II Magnetic measurement system facility" PAC'13, THPAC15, Pasadena, CA USA (2013).
- [3] Helmholtz-Zentrum Berlin für Materialien und Energie GmbH, Berlin, Germany (HZB). http://www-csr.bessy.de/control/SoftDist/CA_Lab

INTEGRATING SIEMENS PLCs AND EPICS OVER ETHERNET AT THE CANADIAN LIGHT SOURCE

R. Tanner, S. Hu, G. Wright, CLSI, Saskatoon, Canada
E. Matias, Mighty Oaks, Victoria, Canada

Abstract

The Canadian Light Source (CLS) is a 3rd-generation synchrotron light source on the University of Saskatchewan Campus in Saskatoon, SK, Canada. The control system is based on the Experimental Physics and Industrial Controls System (EPICS) toolkit [1]. A number of systems delivered to the CLS arrived with Siemens, PLC-based automation. EPICS integration was initially accomplished circa 2003 using application-specific hardware; communicating over Profibus. The EPICS driver and IOC application software were developed at the CLS. The hardware has since been discontinued. To minimize reliance on specialized components, the CLS moved to a more generic solution, using readily-available Siemens Ethernet modules, CLS-generated PLC code, and an IOC using the Swiss Light Source (SLS) Siemens/EPICS driver [2]. This paper will provide details on the implementation of that interface. It will cover detailed functionality of the PLC programming, custom tools used to streamline configuration, deployment and maintenance of the interface. It will also describe handshaking between the devices and lessons learned. It will conclude by identifying where further development and improvement may be realized.

SYSTEM OVERVIEW

The general system overview is shown in Fig. 1. Siemens PLCs from the S7-400 and S7-300 families have been fitted with Industrial Ethernet modules.

These PLCs have been delivered by vendors. There are two S7-300s for the storage ring SRF (Superconducting Radio Frequency) cavity, one in the BR (Booster Ring) RF controls, and S7-400s for the Linde Kryotechnik-supplied cryogenics plant and SR (Storage Ring) Thales RF amplifier.

At the time of this writing, EPICS integration of the TUV-certified, failsafe PLCs used in the CLS Access Control and Interlock System (ACIS) and O2 monitoring is still under review.

Because the PLCs perform many critical functions, a separate Virtual Local Area Network (VLAN) had been dedicated for them (referred to as a “plantbus” by the manufacturer), the EPICS IOC (Input-Output Controller) and Engineering Stations (“ES”). Access to the PLCs is meant to be accomplished only via the development stations or the IOC. The ESs have an industrial Ethernet adapter for connecting to the plantbus and commonly – available adapter for connecting to the “office” network. The IOC connects using the default VMWare Ethernet adapter and is configured to be on the VLAN, as well.

Any other services are configured to be available to the VLAN rather than accessible via a gateway or router (the IOC also provides NTP, for example). Traffic is restricted on the VLAN for performance and security considerations.

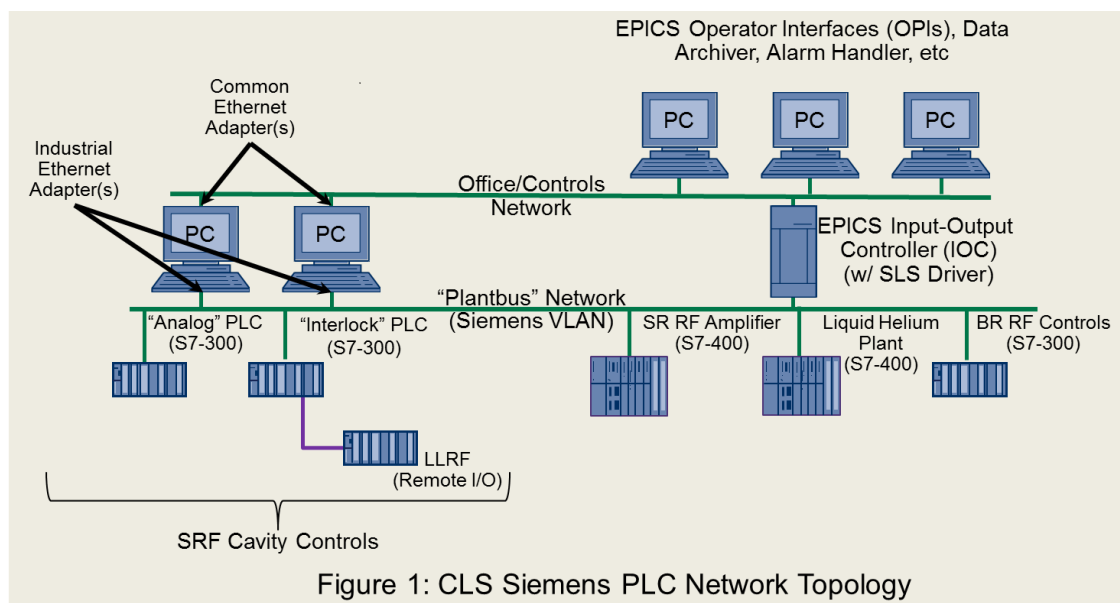


Figure 1: CLS Siemens PLC Network Topology

IOC

The IOC is running as an Intel-based, virtual machine (VM) on an ESXI VMWare. The O/S is Red Hat Linux 7.2 (Enigma).

The SLS Ethernet driver and IOC application were compiled against EPICS 3.14.6. All PLCs can be interfaced with a single IOC application, or a specific app written for each PLC or any combination thereof. The CLS will likely move to one IOApp/PLC; which is more modular and fault-tolerant.

The IOC application was modified slightly to include a synchronization step with the PLC before it is allowed to set values in the PLC.

PLC CODE

The PLC code is written using Siemens PCS 7; which provides a number of development languages ranging from an Assembly-like option called Statement List (STL), a Pascal-type language called Structured Control Language (SCL) and graphical languages including a Ladder Logic offering (LAD), Function Block Diagram (FBD) as well as higher-level languages such as Sequence Function Chart (SFC) for state logic programming and Continuous Function Chart (CFC), the language used for safety system programming.

The PLC side of the EPICS interface was written using SCL.

Functions

All the code and variable definitions are contained in a single SCL file. Comments indicate where automatically-generated code is to be inserted.

The CLS PLC code consists of four functions; IE_COMMS, CHECKSYNC, PACK_TX_DB and UNPACK_RX_DB. The core functionality is handled by the IE_COMMS function.

IE_COMMS calls PACK_TX_DB which copies data from various internal memory areas to a buffer (IE_TX). The data consists of all the values meant to be sent to the IOC as well as the current values the IOC is interested in setting (readback values). IE_COMMS then calls the SEND function and waits for it to return. Upon return, it logs any errors and starts the process again. A watchdog timer will also restart the procedure if SEND times out.

IE_COMMS also monitors the RECEIVE function watching for the arrival of data. Before the IOC can manipulate data in the PLC it must synchronize with it by demonstrating it has the current values of the variables in question. When data first arrives, if the SYNC flag is not set, IE_COMMS calls CHECKSYNC. If CHECKSYNC succeeds, the SYNC flag is set and UNPACK_RX_DB is called which moves data from IE_RX to the internal PLC locations using a combination of pointer-based and explicit transfers as outlined below.

As with the SEND function, errors are logged before preparing for the next packet of data.

Data Transfers

Transfers are accomplished either by explicit assignments or pointer-based operations. Explicit assignments are simple Var1 = Var2 statements (i.e. “IE_TX.DBD8 := DB4.DBD12;” copies the double word at offset 12 of datablock 4 to IE_TX at offset 8).

In cases where a number of variables are adjacent in a contiguous section of memory, it is convenient and simpler to call the BLKMOV function (analogous to memcpy in C) which takes “ANY pointers” as source and destination parameters. ANY pointers are 10 bytes long and contain fields for the length of the data to copy, the memory area (i.e. input, output, global memory, datablock, etc), the datablock number (if applicable) and the *bit offset* in that memory area to start the copy. For example, if 16 boolean values, 4 integers and 6 floating point values were located in a single datablock starting at byte 4, they could be copied in a single statement using the pointer, “P##DB2.DBX4.0 BYTE 34”.

In order to access the fields of the ANY pointer, a User Data Type (UDT) is defined (called AnyPoint) that specifies the pointer layout. The UDT is then associated with ANY pointers in a manner similar to a union{} in C as follows:

```
BUFFER_TX: AnyPoint; // UDT
ANY_SRC at BUFFER_TX: Any; // POINTER
```

For both PACK_TX_DB and UNPACK_RX_DB, a list of pointers is maintained in datablocks (DBs) named DB_TX_PTRS and DB_RX_PTRS, respectively. The pack and unpack functions iterate through the pointers performing the pack or unpack operations as required.

The IE_TX and IE_RX buffer are also datablocks. Datablocks (DB) are user-defined records analogous to a C struct{ }.

SYMBOL MAP

The PLCs delivered as part of larger systems do not generally have internal symbols which adhere to the CLS naming convention. Therefore, a symbol map of each PLC is manually generated mapping the internal symbol name to the CLS-defined PV name. It also includes the internal (PLC) address of the variable, its offset in the TX or RX buffer, the description, type of transfer (Explicit or Pointer) as well as a number of EPICS-specific definitions required for the substitution files.

Scripts

Adding or inserting another process variable (PV) to the map requires a lot of manual editing for both the IOC and the PLC, which is painstaking and leads to errors. The symbol map was built in MS Excel which provides an interface for automation. VBA scripts were written to generate EPICS substitution files for the IOC as well as to

generate Siemens SCL code to be copied in to the environment for compilation.

The scripts are in a relatively primitive stage (i.e. not very flexible and with minimal exception-handling) but even so, have proven to save time and significantly reduce clerical errors.

LESSONS LEARNED

IE_TX and IE_RX have four-byte headers followed by the data. One has to be cautious of relative vs absolute offsets. Byte 0 of the PLC IE_TX data is byte 4 of the TX DB (and of the IOC input buffer).

The CLS implementation allows the PLCs to transmit as fast as they can (rather than on a set period). The S7-400s turned out to be too fast; flooding the data archiver and causing jarring updates on the display(s). An input parameter was added to allow for an additional delay interval.

Some systems come complete with an HMI. IE_COMMS has a Boolean input (LOCAL/REMOTE). If the parameter is set to LOCAL, IE_COMMS will not unpack IE_RX. It will however, keep the IOC updated via the readback values in IE_TX.

PLC timers are explicitly defined. One must be careful that timers used for the EPICS functions, are not assigned elsewhere.

The use of pointers is notoriously wrought with pitfalls and ANY pointers are no different. One must be diligent with respect to data types (integer vs word) because the PLC compilers are less flexible, forcing the developer to be precise regarding definitions.

The PLC user code/compilers allow direct access to digital values (i.e. the 3rd channel of a digital input card starting at address 4 would be "I 4.2"). To accommodate such addressing, the ANY pointer offset is a *bit offset*. To refer to byte 4, the ANY offset would be set to 32 (or rather 20, because it is a hex value).

As with data types, the format of some function parameters must be explicitly defined and are dependent on the language in which the function is being used. For example, a 3-second timeout in CFC is represented as "3s" whereas in a LADDER diagram it would be specified as "S5T#3S".

The vendor-supplied PLC RECEIVE function expects periodic refreshes from its connection partner. This acts as a heartbeat to confirm that the other device is still connected and functioning. If it does not get them, it assumes an error. Because the IOC driver only sends data when there are updates, the PLC throws multiple errors/second. As a workaround, logging of that error has been commented out of the code.

FUTURE DEVELOPMENT

Currently, the SYNC flag is only reset when the watchdog timer expires or when the PLC is restarted. A slightly more robust realization could result if the EPICS driver sent periodic updates as expected by the RECEIVE function (rather than only when it has new data for the

PLC). A loss of communication could reset the SYNC flag and logging of these events reintroduced to provide potentially useful insight in to how often that occurs.

Significant convenience would be realized by having the IOC populate the TX and RX pointer tables. This would completely remove all EPICS configuration from the PLC, limiting modification to the addition of the code and DBs required for communicating with the IOC. It would remove a lot of the manual work that is still required on the PLC end of configuration.

Having systems delivered with symbols already named in accordance with a naming convention would permit that table to be exported to the Symbol Map. Building the Symbol Map is the largest effort in the current work flow so improvements here have the biggest impact.

CONCLUSION

The use of SCL greatly simplifies integration with existing projects irrespective of the tools used to develop them. It similarly simplifies uploading an existing PLC binary, compiling the EPICS-specific blocks in to the code and downloading to the PLC, which is most often the case for vendor-supplied automation.

To date, four vendor-supplied PLCs have been set up to use the CLS-modified SLS driver for interfacing with EPICS with a fifth being targeted for our 2014 fall shutdown. A very small amount of data has been written to the PLCs from EPICS, with considerably more planned in the future. A larger amount has been read out from them. The SLS driver, IOC application and CLS-developed PLC code has not required any modification or debugging since installation. The combination has proven stable over more than a year of deployment.

The investment in development time has simplified future installations, eliminated the reliance on exotic hardware, reduced implementation time and clerical errors as well as enhancing performance.

ACKNOWLEDGMENT

The CLS gratefully acknowledges the contributions of its operating funding partners:

- University of Saskatchewan (UofS)
- Canadian Foundation for Innovation (CFI)
- Canadian Institutes of Health Research (CIHR)
- National Research Council (NRC)
- Natural Sciences and Engineering Research Council (NSERC)
- The Government of Saskatchewan
- Western Economic Diversification (WD)

REFERENCES

- [1] <http://www.aps.anl.gov/epics>
- [2] <http://epics.web.psi.ch/style/software/s7plc/s7plc.html>

SETUP OF A HISTORY STORAGE ENGINE BASED ON A NON-RELATIONAL DATABASE AT ELSA

D. Proft*, F. Frommberger, W. Hillert, ELSA, Bonn, Germany

Abstract

The electron stretcher facility ELSA provides a beam of unpolarized and polarized electrons of up to 3.2 GeV energy to external hadron physics experiments. Its in house developed distributed computer control system is able to provide real time beam diagnostics as well as steering tasks in one homogeneous environment. Recently it was ported from HP-UX running on three HP workstations to a single Linux personal computer.

This upgrade to powerful PC hardware opened up the way for the development of a new archive engine with a noSQL database backend based on Hyptertable. The system is capable of recording every parameter change at any given time. Beside the visualization in a newly developed graphical history data browser, the data can be exported to several programs - for example a diff-like tool to compare and recall settings of the accelerator.

This contribution will give details on recent improvements of the control system and the setup of the history storage engine.

INTRODUCTION

The main features of the ELSA accelerator control system [1, 2] include a completely event based data handling model and a separation of the core functionality (database and event handling by the *kernel*) from userspace applications. It combines steering tasks and real time beam diagnostics in one homogeneous environment. A transparent design allows access to the X windows-based graphical user interface from any computer. An overview of the hard- and software layers of the whole system is given in Figure 1, [3].

A key component of the control system is a kernel managing a central shared memory database. The database is separated into several parts, i.e. the *resource base* containing structural information about parameters like limits, max. number of vector elements and the quantity's physical unit. The structural information is complemented by the online database filled with actual parameter values, which are updated continuously at runtime.

There are currently 14 827 parameters defined in the control system. These are grouped into *controlled* (≈ 4000), *measured* (≈ 9000) and other parameters. Each group consists of four different data types: analog values (represented by floating point numbers), digital values (mostly switching values or integers), strings (character sequences) and arbitrary byte sequences.

The update of controlled parameters occurs rather rarely, and is mainly invoked by user interaction or automatic measurement processes. On the other hand most measured

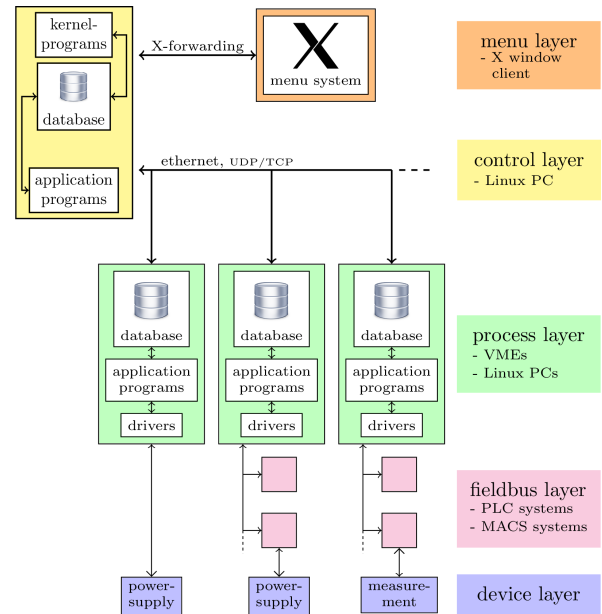


Figure 1: Hard- and software layers of the control system.

parameters are updated on a regular basis, either cycle-synchronous (typically every 5 s) or with arbitrary rates up to 10 Hz. Parameters with higher rates are accumulated in vectors and transferred (also on a regular basis) with a slower rate.

The data rate produced by 675 updates/s (on average) is roughly 50 kB/s to 100 kB/s¹. This results in a total volume of ≈ 6.1 GB per day.

Primary goal of the newly developed archive engine is, of course, to archive all these changes together with a time-stamp, regardless of the type or source of the values. Second goal is to keep the investment cost as low as possible. Therefore the archive database should run on a regular desktop computer with no special hardware needs. Here, a bottleneck could be the access time, in which the data can be returned back from the database. For best user experience access times in the magnitude of few seconds are required.

DATABASE BACKEND

Hypertable is a non relational database with Google's *Bigtable* design which was chosen as the database backend. It runs on top of several file systems, including distributed ones (e.g. *HDFS*) and storage in the local file system. The instances of the main server, called *RangeServer*, can be distributed among different machines with one *Master* process for administration.

¹ 50 kB/s during maintenance, 100 kB/s during usual operation

* proft@physik.uni-bonn.de

Hypertable uses a *key-value* based data storage model. The key itself is made up by a row key string, a high resolution nanosecond timestamp, a column *family:qualifier*-pair and control flags. The timestamp can be understood in two ways: First as a simple timestamp either assigned automatically upon creation or given by the user and second as a revision of the *key-value* row. The column family² represents the *column name* in relational databases. These fields are assigned to the archive engine fields as shown in Table 1.³

Table 1: Hypertable \leftrightarrow Archive Engine Field Assignment

hypertable	archive engine
row key	parameter name
column	fixed column family name "data"
timestamp	recorded parameter change date
value	parameter value

The *key-value* pairs are sorted by their key and stored inside the memory in *CellCaches* or they are written to compressed *CellStores* residing on disk. The data on disk is supplemented by a block index, to increase search performance.

This type of data storage directly implies the optimal way of data readout: Because the data is sorted by the key (i.e. parameter and timestamp) it is most efficient to read out a big time frame for a single parameter. This is exactly what most of the history-tools (and especially the history-browser application) require, so it matches the requirements for the database backend.

Currently the hypertable database (one *Master* and one *RangeServer*) is running on the same machine as the control system. It is equipped with an Intel i7 CPU with six physical cores, 40 GB RAM⁴ and two desktop harddrives with each 3000 GB capacity configured as a raid1 (no distributed file system is used at the moment).

INTEGRATION INTO THE EVENT SYSTEM

The interface to the database backend is set up on the control host. The shared memory database running here has a consistent view of all parameters and their current values. Upon each parameter update, the event system is triggered to inform other applications of the value change. At this point a new hook was installed to communicate with the history database.

For the implementation, emphasis was put on the strict separation of the control system's core and the database communication. Therefore a new shared memory database was

² Because only one column is used for the historic data, this feature is effectively unused.

³ The parameter name used as the row key is suffixed by a date based string for faster indexed searches and due to a maximum revision count in hypertable.

⁴ Before a recent upgrade of the control system to 64 bit the usable RAM of the database was limited to 2 GB. All further performance analysis has been performed with this limitation present.

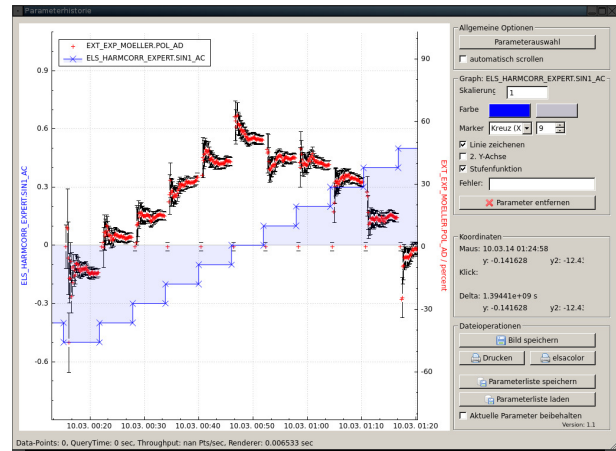


Figure 2: Graphical user interface: parameter history browser showing the correlation between set point value and corresponding measurement value.

introduced to act as an intermediate database. Whenever a parameter gets updated, a nanosecond timestamp with the current time is created and stored in the shared memory database along with the parameter's name and value. Numerical values (integers and floating point values) are stored in their binary representation with 32 bit size⁵ and strings as zero terminated character arrays.

The isolation from the control system core is achieved by using only one application with access to both systems. Its purpose is to flush the contents of the intermediate database every three seconds and insert the appropriate records into the hypertable database. Each new record is filtered by a regular expression during the insert to filter away unneeded parameters by name to save storage size.

TOOLS

For interaction with the history data a couple of tools have been developed. The most important one is a graphical user interface, which can be directly invoked from the accelerators menu system (see Figure 2). Within the GUI, one can ask for values of multiple parameters and have them plotted versus time. The application is based on *QCustomPlot*, a Qt plotting widget with integrated support for easy panning and zooming by mouse.

Beside simple tools for extraction of data to ASCII files and plotting in gnuplot there are two important applications:

cshdiff

This application creates two snapshots of the values of all parameters at two given dates and afterwards reports any differences between them. The number of parameters can be filtered by type (e.g. controlled or measured parameters) and by name (regular expression). That way, a comparison of two different machine states is easily achieved. This information may then be used to restore the accelerator to a previous state.

⁵ Accordingly, vectorial parameters are stored as $n \times 32$ bit values.

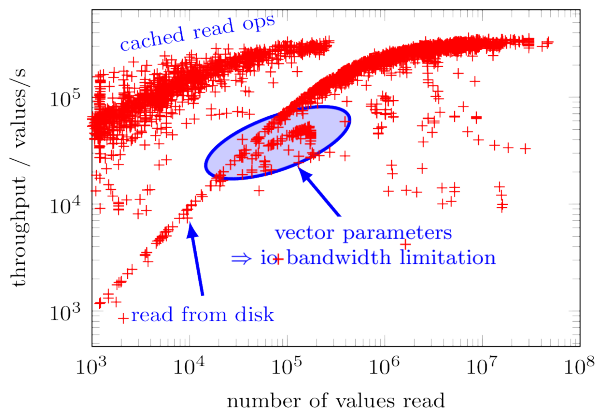


Figure 3: Parameter readout performance vs. number of parameters read.

csplayback

This application is designed to be executed on a development system running the control system. Instead of connecting the control system to process hosts, only a single application (beside the crucial kernel applications) is running. It downloads all parameter values from the database system for a given time range and feeds them back to the development system. This allows a real time playback of all accelerator parameters of any time range in the past⁶. From the control systems point of view there is no difference in data handling in comparison to normal operation with attached process hosts, because the same event- and notification system is used.

The most important scenario where this feature can be used is software development: On the development system new software, for example measurement applications, can be tested offline and no dedicated beam time is required.

PERFORMANCE

The most used feature of the archive engine is the history database browser. For maximum user experience a fast readout and display of the data is required.

Figure 3 shows the basic readout performance of the database system after 8 month of operation. Every data point represents the throughput during readout of all values stored in the database of one accelerator parameter. Dependent on the update rate of each specific parameter, the total number of values per parameter varies among 5 orders of magnitude. The data was collected in a random order during usual system load, especially the collection of new data was not interrupted.

Most of the parameters with only few data points (less than 1×10^5) can be hold in cache, thus being accessible directly from the RAM. These queries can be executed at high throughput and are located in the upper left region of figure 3. If the data is not cached, the readout of small amounts of data takes significantly longer due to an additional overhead by

I/O latency of the hard drives and on-the-fly decompression of the data. The throughput increases with bigger amounts of data being read, because the time needed for preparation of the data is constant. On the other hand, parameter values which are vectors (n-tuples of scalar values instead of single scalar values) can only be read out by a lower rate due to I/O bandwidth limitations.

Figure 4 shows the total time required for export versus the number of values queried from the database. The readout again was performed in a random order and takes less than 1.5 s for the readout of up to 10 000 values. Above that point the throughput is dominated by the delay given by I/O operations for reading the *CellStores* from disk and the corresponding decompression.

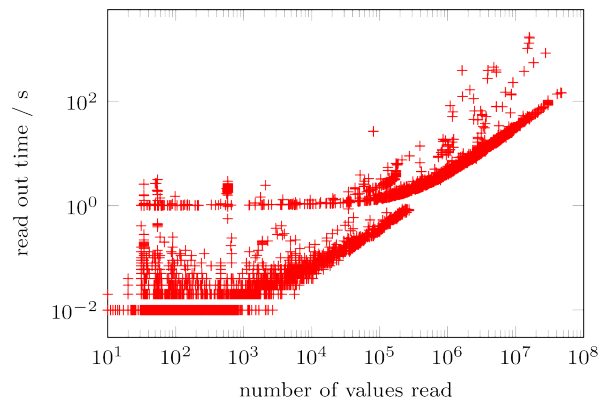


Figure 4: Readout time vs. number of parameters read.

CONCLUSION

The possible uses of the archive engine overshoot the simple recording and display of data: Now post-mortem analysis of component failures are possible. One can find correlations between different parameters - either controlled or measured ones - and watch their evolution over time. For that, the most important improvement introduced is the graphical history browser application. It quickly became an integral and vital part of the control system.

REFERENCES

- [1] T. Götz, "Entwicklung und Inbetriebnahme eines verteilten Rechnerkontrollsystems zur Steuerung der Elektronen-Stretcher-Anlage ELSA, unter besonderer Berücksichtigung der Anforderungen des Nachbeschleunigungsbetriebes bis 3.5 GeV", PhD theses, University of Bonn, 1995
- [2] M. Picard, "Entwurf, Entwicklung und Inbetriebnahme eines verteilten Rechnerkontrollsystems für die Elektronen-Stretcher-Anlage ELSA, unter besonderer Berücksichtigung der Extraktion im Nachbeschleunigungsbetrieb bis 3.5 GeV", PhD theses, University of Bonn, 1995
- [3] D. Proft, "The accelerator control system at ELSA", IPAC2013, Shanghai, May 2013, THPEA002, p. 3149.

⁶ since begin of the recording of course

NEWS FROM THE FAIR CONTROL SYSTEM UNDER DEVELOPMENT

R. Bär, C. Betz, D. Beck, J. Fitzek, U. Krause, S. Jülicher, M. Thieme, R. Vincelli
GSI Helmholtz Center for Heavy Ion Research, Darmstadt, Germany

Abstract

The control system for the FAIR (Facility for Antiproton and Ion Research) accelerator facility is presently under development and implementation. The FAIR accelerators will extend the present GSI accelerator chain, then being used as injector, and provide anti-proton, ion, and rare isotope beams with unprecedented intensity and quality for a variety of research programs.

This paper summarizes the general status of the FAIR project and focusses on the progress of the control system design and its implementation. This paper presents the general system architecture and updates on the status of major building blocks of the control system. We highlight the control system implementation efforts for CRYRING, a new accelerator presently under re-commissioning at GSI, which will serve as a test-ground for the complete control system stack and evaluation of the new controls concepts.

FAIR

FAIR is a unique new international accelerator facility for the research with antiprotons and heavy ions. When finished (planned for 2019), FAIR will be a host laboratory for basic research for about 3000 scientists from approximately 50 countries.

The FAIR accelerators are a major extension of the present GSI accelerators then being used as injectors. Figure 1 gives an overview of the full FAIR accelerator complex. FAIR will be built in a modular approach, starting with the synchrotron SIS100, two consecutive storage rings CR and HESR and the p-linac proton injector.

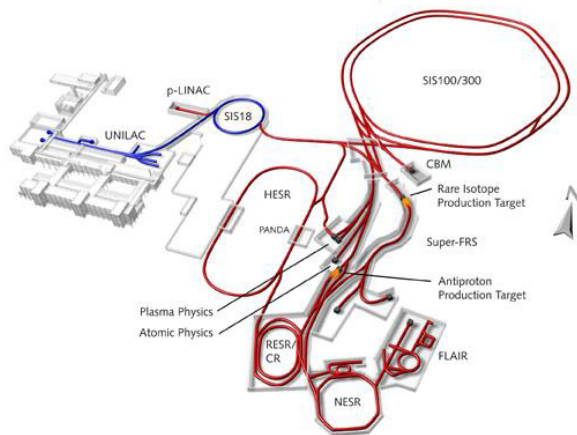


Figure 1: Schematic overview of the GSI (blue, existing) and FAIR (red, to be built) accelerator complex.

Civil Construction

The FAIR complex will cover an area of 20 hectares and require 600,000 cubic metres of concrete as well as 35,000 tons of steel [1]. Construction teams will be building a tunnel to house the heart of the complex, the SIS-100 ring accelerator with a circumference of 1.1 kilometres. The 24 buildings and several tunnels provide sufficient room for a total of 3.5 kilometers of beam-lines as well as huge detectors and a complex technical infrastructure.

Figure 2 gives an impression of the construction site. After clearing and preparing the construction area, approximately 1350 bore piles have already been built to stabilize the subsoil, minimize building settlement and ensure that buildings will settle evenly.

Moreover, several kilometers of access roads have already been built across the site for the construction site traffic.



Figure 2: Aerial photo of the construction site taken on May 25, 2014 (photo: J. Schäfer, FAIR).

CONTROL SYSTEM OVERVIEW

The FAIR accelerator control system comprises the full electronics, hardware, and software to control, commission, and operate the GSI/FAIR accelerator chain with multiplexed parallel beams. The development of the control system takes advantage of several collaborations with CERN by using, adapting and improving framework solutions like the settings management framework LSA, the front-end software framework FESA and the White Rabbit (WR) based timing system as core components.

The general structure of the FAIR accelerator control system is organized in three layers. The equipment layer consists of equipment interfaces, embedded system controllers, and software representations of the equipment. A dedicated real-time network based on White Rabbit is used to synchronize and trigger actions on equipment level. The middle (business) layer provides

service functionality to both the equipment layer and the applications layer. The application layer combines the applications for operators as GUI applications or command line tools written in Java. The next paragraphs especially focus on several systems out of these three layers.

EQUIPMENT CONTROL HARDWARE

The standard equipment controller for most accelerator devices, with the exception of beam instrumentation DAQ systems and industrial type accelerator infrastructure systems (e.g. vacuum components), is the Scalable Control Unit (SCU). The SCU provides a uniform platform connected to both the timing- and Ethernet communication network and is designed to real-time control and synchronize equipment actions of up to 12 purpose-built slave cards, connected in a proprietary 3 HU crate by a parallel backplane bus.

Slave boards provide additional functionality and the necessary hardware interfaces to control the actual accelerator components, e.g. a wide range of magnet power converters, RF components, etc.

Status

After intensive testing of 50 SCU engineering samples in a dedicated test environment last year, a minor redesign was recently performed to address some remaining issues and slightly modify the front plate interface. Presently a first batch of 100 pre-series SCUs is being produced. The production of 600 units for FAIR is presently prepared and shall be completed by the end of 2015.

As part of the SCU solution toolkit several slave boards have been developed. Besides a general purpose AD/DA board (2 channels each) a complex FPGA rear board has been developed that can be equipped flexibly with a set of low-complexity front-side interface boards (see Figure 3). This approach greatly reduces development effort and was inspired by the need to produce several DIO and DA boards to control legacy CRYRING equipment. Furthermore, a board was developed that features the MIL-1553 field-bus that allows to connect to the present GSI timing system and control present (non-FAIR-standard) GSI equipment with SCU controllers as part of our future strategy to control “legacy” equipment.

In order to feature real-time data supply for ramped accelerator equipment, a Function Generator (FG) functionality was developed. Up to 12 function generators can be implemented in slave-board FPGAs and can be controlled from one SCU [2].

FRONT-END SOFTWARE FRAMEWORK

Software for the Front-End equipment control computer will be developed using the FESA (Front-End Software Architecture) framework, which was originally established by CERN. In collaboration between CERN and GSI the framework was redesigned completely. The new version, FESA3, is now ready to be used in a production environment.

Emphasis in the redesign was on customization to specific needs of the contributing institutes. In all layers of the framework, a common core can be extended by site specific packages [3]. This modular allows establishing a GSI-specific set of standard properties, common for all devices, and to integrate the FAIR timing system with its proposed multiplexing concept. Providing the framework as set of RPM packages easily supports installation at any location where front-end software is developed.

First FESA3 front-end software has already been developed for an ion source test-bench. Several FESA classes are under development now and will be used in the CRYRING which will be the first machine to be operated completely with the FAIR control system.

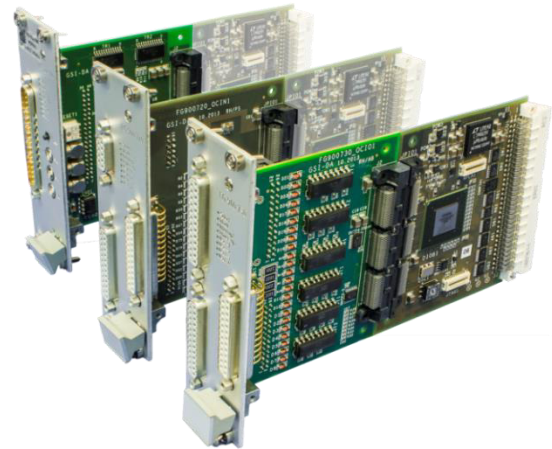


Figure 3: Modular SCU slave board concept: Complex rear-board with several types of simple front-interfaces.

SETTINGS MANAGEMENT

LSA (LHC Software Architecture) is the settings management framework which originates at CERN and since a few years is being enhanced in collaboration with GSI. This framework is used for settings management within the FAIR accelerator control system. The services written in Java are located in the control system’s middle (business) layer. They support offline generation of machine settings, management of these settings and sending them to all involved equipment controllers as well as programming the schedule of the timing system. The settings management is based on a physics model of the accelerators. This includes optics, twiss, machine layout, a parameter hierarchy and overall relations between parameters including calculation rules. A standardized API allows accessing data in a common way as basis for client applications for all accelerators. A set of generic client applications is already being provided that allows accessing all features of the system. They can be used as expert programs and serve as basis for institute specific application development.

LSA is already being used at GSI for certain machine development experiments at the existing synchrotron SIS18. For FAIR, the LSA system is currently being enhanced to support the demanding flexible beam

operations [4]. It is planned to use these new features and verify them even before FAIR will be commissioned.

TIMING SYSTEM

For hard real-time control of all components of the GSI and FAIR accelerator complex, a first version of a General Machine Timing system has been implemented.

Overview

The GMT is based on a common notion of time of all connected nodes, provided by the White Rabbit Precision Time Protocol (WR-PTP) [5]. A dedicated network, based on WR switches distributes commands broadcasted by a Data Master to all Timing Receivers. Relevant commands are decoded and enqueued for timely execution. One time, none, one or more specific actions on the local configuration of the Timing Receiver are being executed. A central Clock Master serves as grand-master clock to which all nodes in the WR network synchronize their clock, phase and time. As the GMT is a time-based system, the precision for synchronising actions only depends on the quality of clock and phase synchronisation via WR-PTP and not the propagation time of commands distributed by the Data Master.

Nodes of the Timing System

The focus of the past two years has been the development of nodes of the timing system in all aspects: Hardware, Gateware (VHDL code), Firmware (embedded CPU code), drivers and software. The hardware for the form factors PCIe, VME and standalone has been implemented. Another form factor, the Scalable Control Unit (SCU) [3] has been integrated into the GMT too. On-time actions such as digital output, complex I/O via the SCU bus or signalling actions to the FESA software is not provided by the GMT. The key feature common to all nodes is the Event-Condition-Action (ECA) unit [6].

Data Master

The Data Master schedules actions by broadcasting commands to Timing Receivers via the WR network. A first Data Master has already been implemented based on the WR node of the form factor PCIe. The key component of the Data Master is a Lattice Micro 32 multicore cluster of soft-CPU's embedded in the FPGA of the PCIe module [7].

Status

A first version of the GMT is "ready for installation" at CRYRING. Besides, a timing network connecting six GSI buildings has been set up with about 15 WR switches. This allowed also for merging the GMT and DAQ systems and provided a decisive test on the potential stability of WR based timing systems.

NEXT STEPS: CRYRING

As Swedish in-kind contribution to FAIR, the heavy-ion storage ring CRYRING has been decommissioned in

Stockholm, transported to GSI, and is presently being installed behind the existing GSI Experimental Storage Ring (ESR) [8]. CRYRING (see Figure 4) can decelerate, cool and store heavy, highly charged ions that can come from ESR down to a few 100 keV/nucleon. It is equipped with its own injector line that will allow CRYRING operation even while the full GSI accelerator chain is shut-down until mid of 2017 for necessary FAIR upgrade and civil construction work.

As CRYRING has been dedicated as a test ground for the FAIR accelerator control system (as well as for a variety of other technical subsystems), the next step is to set up the control system for re-commissioning CRYRING in the next months. The main intention is to test and validate fundamental concepts, technologies and gaining experience under real conditions in order to identify possible design flaws or limitations, and to assure the quality of the control system components involved. While in the beginning the control system and its building blocks will only provide basic features, the intention is to add more and more functionality in the coming control system releases.

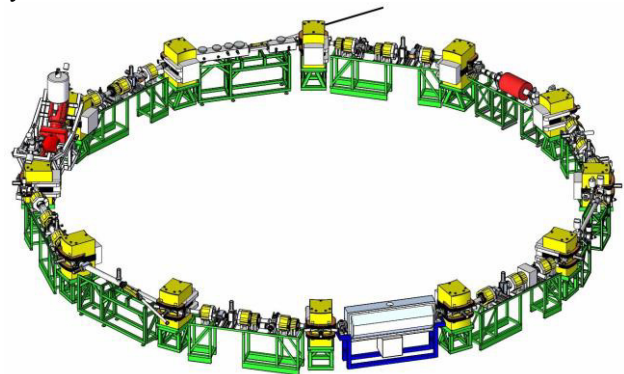


Figure 4: Overview of the storage ring CRYRING under installation at GSI (injection lines not shown).

REFERENCES

- [1] www.fair-center.eu
- [2] S. Rauch, "Managing Multiple Function Generators for FAIR", FPO017, PCaPAC 2014.
- [3] S. Matthies et al., "FESA3 Integration in GSI for FAIR", WPO006, PCaPAC 2014.
- [4] H. Hüther et al., "Progress and Challenges during the Development of the Settings Management System for FAIR", WPO005, PCaPAC 2014.
- [5] J. Serrano et al., "The White Rabbit Project", TUC004, ICALEPCS (2009), Kobe, Japan, 2009.
- [6] W. Terpstra et al., "Inexpensive Scheduling in FPGAs", TCO301, PCaPAC 2014
- [7] M. Kreider et al., "New developments on the FAIR Timing Master", FPO022, PCaPAC 2014
- [8] F. Herfurth et al., "The Low Energy Storage Ring CRYRING@ESR", THPM1HA01, COOL2013, Murren, Switzerland.

PROGRESS AND CHALLENGES DURING THE DEVELOPMENT OF THE SETTINGS MANAGEMENT SYSTEM FOR FAIR

H. Hüther, J. Fitzek, R. Müller, D. Ondreka, GSI, Darmstadt, Germany

Abstract

A few years into development of the new control system for FAIR (Facility for Antiproton and Ion Research), a first version of the new settings management system is available.

As a basis, the CERN LSA framework (LHC Software Architecture) is being used and enhanced in collaboration between GSI and CERN. New aspects, like flexible cycle lengths, have already been introduced while concepts for other requirements, like parallel beam operation at FAIR, are being developed.

At SIS18, LSA settings management is currently being utilized for testing new machine models and operation modes relevant for FAIR. Based upon experience with SIS18, a generic model for ring accelerators has been created that will be used throughout the new facility. It will also be deployed for commissioning and operation of CRYRING by the end of 2014.

During development, new challenges came up. To ease collaboration, the LSA code base has been split into common and institute specific modules. An equivalent solution for the database level is still to be found. Besides technical issues, a data-driven system like LSA requires high-quality data. To ensure this, organizational processes need to be put in place at GSI.

FAIR CONTROL SYSTEM REQUIREMENTS

Construction work for FAIR has been started in 2011 next to the existing GSI complex. In May 2014, the first key construction phase has been completed with the conclusion of pile-drilling work for the foundations of the facility. Once fully operational, FAIR will provide nine new accelerator installations, using the existing linac and synchrotron SIS18 as injectors [1]. See Fig. 1 for an overview of the FAIR accelerator complex.

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.

Tight resource restrictions make meeting these requirements even more challenging. After thoroughly evaluating possible options for most effectively implementing a new settings management component for the FAIR control system, enhancing CERN's existing LSA framework was identified as the most suitable approach.

ISBN 978-3-95450-146-5

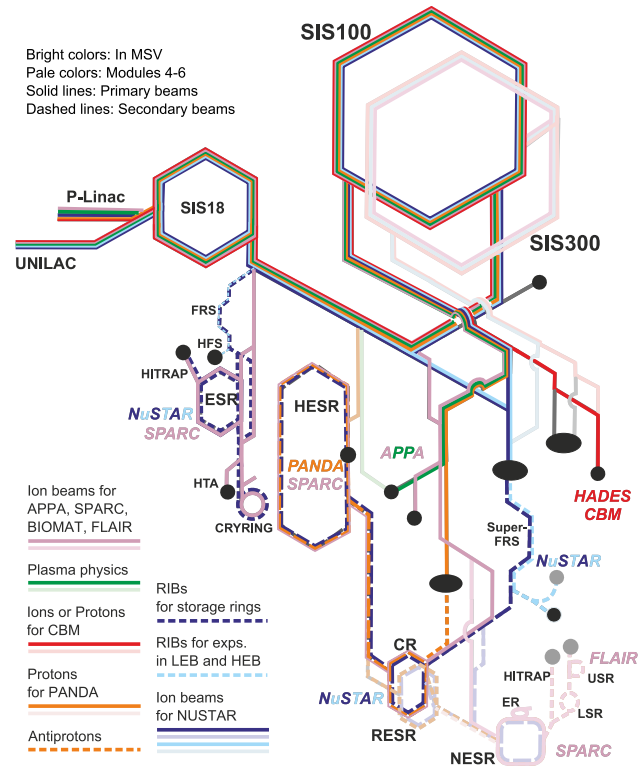


Figure 1: Overview of the FAIR accelerator complex, P. Schütt, GSI.

COLLABORATION WITH CERN

LSA is being developed at CERN since 2001. In the following years, it has constantly evolved and matured [2]. Today, CERN uses LSA to control the majority of its accelerators.

Since 2007, GSI and CERN collaborate to enhance LSA for mutual benefit and to use it as the core component for settings management within the FAIR control system.

LSA has been chosen because of its completeness regarding all important settings management aspects and also certain characteristics that are especially important for implementing enhancements towards FAIR. The framework has been designed with extensibility in mind, reflected by modular structure and separation of concerns in its design. It provides generic means of modelling different accelerators and plug-in mechanisms for adding functionality while a modern architecture ensures scalability.

As powerful as LSA already was when the collaboration started, it still requires major enhancements to support the specific requirements of FAIR. Significant steps towards this goal have been made since progress has last been reported on publicly.

PROGRESS SINCE 2010

The LSA framework is now feature-complete for SIS18. Although the machine model for this accelerator has not been fully completed and not all of its operation modes are covered yet, the system allowed for operating SIS18 to carry out machine development work already targeted at the future FAIR facility. Two examples certainly worth mentioning include resonance compensation experiments using tune variations at injection and extraction plateaus as well as commissioning of the new H=2 cavity.

These advances show that the teams keep their focus on the requirements for the new facility, with the test installation of the CRYRING storage ring to be commissioned in the coming months. Some of the fundamental concepts targeted at FAIR's flexible operation modes have already been implemented and shall be tested with CRYRING. Two of them, being most distinguished from a settings management perspective, shall be described in more detail.

GENERIC MODEL FOR RING ACCELERATORS

The fact that the structure of LSA adheres to the principle of separation of concerns shows its benefits, amongst others, in decoupling work packages between physicists and computer scientists. While the framework itself is responsible for executing algorithms according to the machine model and supplying settings to devices, development of the machine model itself is done using tools that supply the machine model to the framework database with physics algorithms being implemented in a plug-in style.

Modelling accelerators in LSA works by defining a hierarchy that describes parameters for accelerator control from top-level physics parameters (e.g. particle type, energy) via parameters that represent intermediate calculation results (e.g. revolution frequency) down to hardware parameters (e.g. power converter currents) and the relations between them. Associated with each relation is an algorithm component (called "rule" within in LSA) that calculates a dependent parameter from its parents [3].

For FAIR, the fact that several new accelerators are being built concurrently calls for an approach that maximizes reuse of modelling artefacts and optimizes resource usage. Instead of creating different models for each accelerator from scratch, machine physicists at GSI decided to create a generic model for ring accelerators that shall be the basis for all synchrotrons and storage rings within FAIR. This model was being used to operate SIS18 and will subsequently be used for CRYRING.

All rings within FAIR will be equipped with a uniform controls interface, making it possible to take advantage of great similarities on the level of physics modelling including input parameters, relations between them and rules responsible for their calculation. Also on the technical level, advantages of a consistent implementation are being leveraged, including handling of timing and RF.

Using this generic approach, similarities can be expressed using a single rule for calculations in multiple accelerators and a generic hierarchy that is part of each accelerator's model. Distinctive features of certain accelerators are added to the model while keeping the generic structure, similarly to the concept of abstract and concrete classes in software engineering. Figure 2 shows an excerpt of the SIS18 parameter hierarchy as an example.

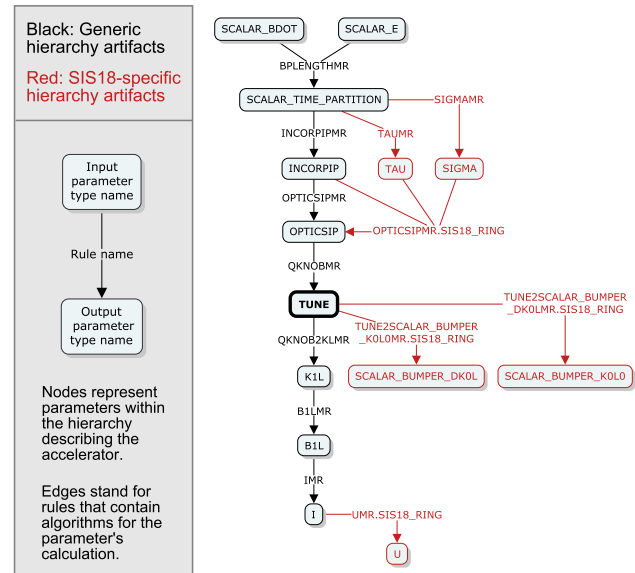


Figure 2: Hierarchy for inputs and outputs of the tune parameter type, showing how distinctive features of SIS18 are added while keeping the generic ring accelerator model structure intact.

There are two main advantages to the described approach. It reduces duplication, which results in less effort for modelling all additional rings and also increases consistency, which makes machine models and operation processes easier to understand. Consequently, the effectiveness of training and maintenance measures is also expected to benefit from this generic modelling approach.

BEAM PRODUCTION CHAINS AND PATTERNS

Beam production chains and patterns are the central technical concept which shall allow for highly flexible operation planning across the whole FAIR facility. They will be tested for the first time during commissioning and test-operation of CRYRING.

Representing a major change in perspective, beam production chains establish a beam-oriented view on the facility compared to the accelerator-oriented view towards settings management dominant in LSA up to this point.

However, beam processes, which are the most central building blocks within LSA for describing settings within a time span, will continue to be used. Beam processes represent a specific procedure on the beam within one accelerator (e.g. injection, ramp, extraction). Rather than being con-

tained in higher-level cycles and super cycles, which are also accelerator-specific, they will now be part of beam production chains, which allow for scheduling a beam across accelerators. To be able to coordinate multiple beams traversing the facility in parallel, beam production chains will be grouped into patterns. An example of typical parallel operation for the modularized start version of FAIR is given in Fig. 3.

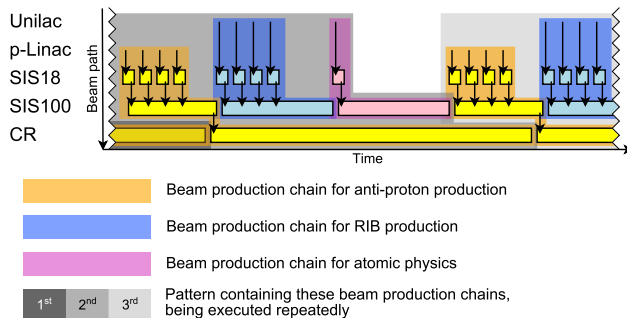


Figure 3: Example for parallel beam operation showing aggregation of beam process chains into patterns. HESR accumulating the anti-protons is omitted.

Relying on the existing beam process concept of LSA, beam production chains will not contradict the framework's principles, but become an additional modelling element, complementary to existing cycles and super cycles, leaving it to the framework's user to decide which one to use, based on the requirements.

Some modifications to LSA were necessary as prerequisites to the implementation of the described concept. All of them were carried out in close cooperation and with most valuable support from CERN, including flexible beam process lengths and optics definitions which are now relative regarding time [4]. Most recently, flexible assignment of timing information to different levels of contexts has been realized.

REMAINING DIFFICULTIES

Within the years past, the collaboration between CERN and GSI continued to thrive and mature, even allowing for a constructive retrospect and good practices to be extracted [5]. Still, on a technical level, some difficulties remain.

The LSA core module is being built separately at both institutes, making completely controlling the process possible for each of them. Other CERN libraries however, which are not included in the collaboration agreement, have to be treated as third party libraries by GSI. They cannot be built at GSI, but in the future, being able to apply bug fixes to them might be necessary to ensure operability of the FAIR facility.

For historical reasons that have proved difficult to overcome, CERN and GSI continue to use different build systems to prepare their LSA releases, repeatedly causing version inconsistencies in deployed packages. A new build system

currently being developed at CERN is expected to resolve this issue once the transition has been completed.

A joint effort to divide the LSA code base into common and institute specific modules has proved most beneficial. On the database level, an equivalent solution is still being investigated. This missing separation on the database level leads to tables for certain features of LSA not used at GSI being empty or reported missing by CERN database queries. Both cases are a potential source of issues. Also, it has proved to be problematic to synchronize the state of code and database changes between institutes. While a manual interim solution is in place, a properly tool-supported, permanent solution is still to be found.

OUTLOOK

The commissioning of CRYRING during the coming months will be the next major milestone for the usage of LSA at GSI and FAIR. Having been contributed by the Manne Siegbahn Laboratory of Sweden, the machine itself and necessary infrastructure is currently being set up. CRYRING consists of a small ion storage ring with electron cooling, an RFQ linear accelerator and two injectors for different types of ions. It will be operated solely through the new control system utilizing LSA for settings management. It will also be the first machine to be operated using the new beam production chain and pattern concept.

Building on this, the next logical step is to make operations applications adhere to the beam-focused perspective represented by beam production chains and patterns as well. As all former applications were accelerator-focused, this will be a major change for users and operators.

Apart from purely technical advances, GSI will have to further refine its business processes. As LSA is a data driven system, high-quality data is crucial to its operation. Defined and well-established processes are needed for introducing, verifying and maintaining data including device representations, calibration curves and accelerator master data. Establishing these processes and thus ensuring data quality represents an important step towards operating the new FAIR facility.

REFERENCES

- [1] R. Bär et al., "Development of a new Control System for the FAIR Accelerator Complex at GSI", ICALEPCS'09, Kobe, Japan, TUP107.
- [2] G. Kruk et al., "LHC Software Architecture (LSA) - Evolution toward LHC Beam Commissioning", ICALEPCS'07, Knoxville, Tennessee, USA, WOPA03.
- [3] D. Ondreka et al., "Setting Generation for FAIR", IPAC'12, New Orleans, Louisiana, USA, THPPR001.
- [4] J. Fitzek et al., "Settings Management within the FAIR Control System based on the CERN LSA Framework", PCaPAC'10, Saskatoon, Saskatchewan, Canada, WEPL008.
- [5] 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.

FESA3 INTEGRATION IN GSI FOR FAIR

S. Matthies, H. Bräuning, A. Schwinn, GSI, Darmstadt, Germany
S. Deghaye, CERN, Geneva, Switzerland

Abstract

GSI decided to use FESA (Front-End Software Architecture) as the front-end software tool kit for the FAIR accelerator complex. FESA was originally developed at CERN. Since 2010 FESA3, a revised version of FESA, is developed in the frame of an international collaboration between CERN and GSI. During development of FESA3 emphasis was placed on the possibility of flexible customization for different environments and to provide site-specific extensions to allow adaptation for the contributors. GSI is the first institute different than CERN to integrate FESA3 into its control system environment. Some of the necessary preparations have already been performed to establish FESA3 at GSI. Examples are RPM packaging for multiple installations, support for site-specific properties and data types, first integration of the White Rabbit based timing system, etc.. Further developments such as e.g. integration of a site-specific database or the full integration of GSI's beam process concept for FAIR will follow.

INTRODUCTION

GSI's FAIR [1] project is a challenge and a chance to establish a revised control system solution. A couple of years ago it was decided to develop the main parts of the future control system for FAIR (such as FESA, LSA [2] and the middleware CMW [3]) in the frame of an international collaboration with CERN.

This paper gives an overview of how the FESA3 framework is extended to suit into the future FAIR control system environment.

MODULARITY OF FESA3

To establish the FESA framework on sites different than CERN the main focus during development of FESA3 was modularity and extensibility. Modularity is achieved by clear separation of its components and involved technologies into core- and site-specific packages. Extensibility of the FESA3 framework is ensured by the possibility to provide site-specific extensions to the core packages. This involves the FESA3 framework packages as well as the components of the FESA3 plug-in for the integrated development environment Eclipse.

In general the core packages contain the common code base that is used by both participating sites. The common part provides the interfaces, (abstract) base classes as well as the functionality that does not have to be extended.

The FESA3 framework combines the usage of different technologies and programming languages such as XML, XSLT, Python, C++ and JAVA.

The site-specific components extend the common part by providing the functionality required only by the implementing site. This is realized by using software design concepts such as inversion of control and inheritance, depending on the technology used. Figure 1 gives an overview of the main FESA3 framework components. The extension packages are marked by “-EXT” which stands for either CERN or GSI. Accordingly a similar structure is realized for the parts that constitute the JAVA based FESA3 Eclipse plug-in.

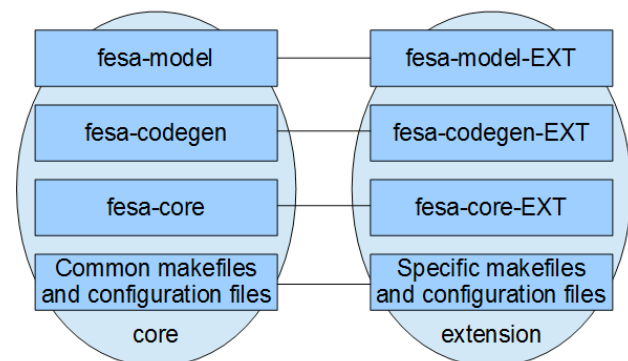


Figure 1: FESA3 Framework Components.

SITE-SPECIFIC FESA FEATURES

The FESA3 installation at GSI has several site-specific features and extensions.

Standard Properties

To provide a common interface of FESA3 based equipment software to the application layer, an elaborated set of standard properties is defined at GSI. Standard properties are common properties that each accelerator device should provide to the application layer. Typical examples are properties such as Status, Power, Init or Version. Site-specific properties may be coupled to site-specific data field types. The properties as well as their data fields are pre-defined in the site-specific template for new FESA3 equipment software. For operational FESA based equipment software to be used within the FAIR control system environment these properties must be implemented by the FESA equipment software developer. GSI's FESA development guideline outlines this and the other issues to be considered when developing productive FESA3 based equipment software for operation of the FAIR accelerator complex.

FAIR Timing Integration

For FAIR a newly build timing system will be used [4].

To be able to react to timing events sent via the new timing network FESA3 provides a prototype integration of the FAIR timing system. This is achieved by providing an additional timing event source within the site-specific part of the FESA core framework. The timing event source analyses the incoming FAIR timing events and triggers the execution of FESA real-time actions. FESA real-time actions are basically threads that can be run in “real-time” and do not directly interact with a client.

Multiplexing: GSI's Beam Process Concept

For operating the FAIR accelerator complex, when several experiments are supported with different beams on a pulse to pulse basis, a staged multiplexing concept will be used. The basic element for describing the actions in the machines is the Beam Process. A Beam Process is an uninterruptible activity like a beam injection, acceleration or beam transfer. Beam Processes are combined into Sequences. A Sequence may represent, for example, a synchrotron accelerator cycle, a set of beam manipulations in a storage ring or the transfer of a beam from one machine to another.

Depending on its functionality, a device may either have different settings for each Beam Process or have one setting for each Sequence, which means it behaves identically for each Beam Process in the Sequence.

This concept has an impact on the existing FESA3 implementation of multiplexing that is tailored to what was needed so far. Since there was no huge difference between CERN's and GSI's requirements the implementation was located in the core part of the FESA framework. To realize GSI's concept of multiplexing using Beam Processes or Sequences in accordance to CERN's multiplexing concept the implementation of the cycle selectors for both involved sites must be moved to the site-specific part.

FESA3 INSTALLATION

One of the specific requirements of GSI is to support installations of FESA3 on multiple Scientific Linux machines. Examples are the servers of GSI's local development cluster, the control system installation for the Proton Linac Source that will be constructed at CEA[5] in Saclay, France or the development environment for the Slovenian in-kind contribution for FAIR. This is in contrast to CERN's need to provide a FESA3 installation within a single development environment for local FESA developers only.

To easily support multiple FESA3 installations on various Linux machines it was decided at GSI to provide FESA3 as well as the required CMW middleware and other site-specific 3rd-party libraries as an RPM (RPM Package Manager) based installation. This allows distributing FESA3 in a comparably simple way. The

main advantage is that RPM packaging enables consistent and repeatable FESA3 installations on different Linux machines in varying environments.

This involves the realization of an installation directory structure that allows more than one FESA3 version in parallel. The chosen installation directory structure for FESA3 is to the greatest possible extent in accordance to a standard Linux directory structure.

EXTENSION OF THE USER INTERFACE: FESA3 ECLIPSE PLUG-IN

Since 2010 the graphical user interface for FESA3 is directly integrated in the development environment Eclipse as a plug-in. This allows to outline the basic development workflow of FESA3 based equipment software. The FESA3 development workflow for FESA equipment software developers involves steps such as

- creating / editing the FESA class design which is formally an XML document to outline the interfaces to the client and the equipment
- implementing automatically generated C++ source code frames which are based on the class design
- compiling the source code and linking the libraries
- creating device instances which are formally described in XML documents
- synchronizing the source code with a software repository such as SVN
- deploying the resulting software, configuration files and start scripts locally and remote on front-end computers
- testing the results locally and remote.

Figure 2 gives a basic overview of the typical development workflow for FESA equipment software.

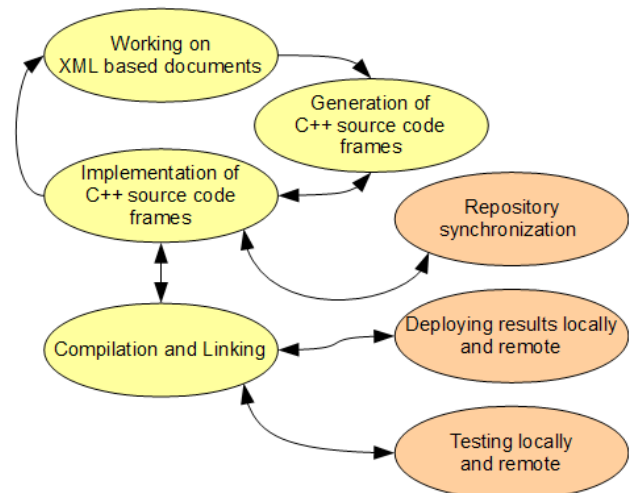


Figure 2: FESA Development Workflow.

The integrated development environment Eclipse provides a solid base for such differing steps including the possibility for custom enhancements and extensions.

So far site-specific characteristics that affect the graphical user interface such as

- different FESA framework installation locations
- support of 32- and 64-bit CPU architectures
- an extensive and configurable delivery process and structure
- an extended and custom-tailored FESA test tool
- a differing SVN repository layout
- a different set of Eclipse cheat sheets to illustrate the recommended FESA software development workflow and provide direct access to further information
- varying automatically generated Make targets
- site-specific expert settings to help ease and speed up the development workflow itself
- a different FESA database environment

have been identified and integrated.

The design involved providing the possibility to extend the core parts by specific parts using software design concepts such as interface-based programming and inheritance. Wherever possible the varying site-specific parts are kept configurable. In addition, the missing flexibility concerning the SVN repository layout or the site-specific database integration is ensured by providing a set of specific JAVA packages.

TESTING IT ALL: THE CRYRING

The Swedish contribution to the FAIR project “CRYRING” serves as an early test system for the first assembly of FAIR's control system. The assembly is currently planned for the end of 2014. The intention is to couple the control system components LSA, CMW and FESA3 for operation in combination with a new FAIR timing system at GSI.

The commissioning of the CRYRING is the first time that the future control system components interact. The intention is to find out how the different parts add up with all the other components.

For this first assembly FESA3 based equipment software is developed at GSI by software developers and equipment specialists in different groups. As more and

more developers use the FESA3 framework to produce operational equipment software the issues that require improvement can be identified.

COLLABORATION

The collaborative approach of developing the FESA3 framework is beneficial for all participating sites. Collaboration has its advantages for all intents and purposes when it comes to the integration of new concepts and ideas, improvements or the implementation of bug fixes.

However software development in collaboration increases the complexity of the whole development workflow itself. Particular care must be taken when introducing changes that certainly will affect the requirements of the other site. Mutual coordination on a regular basis is essential to keep both participating sites informed and up to date.

CONCLUSION

Since starting the collaborative development of FESA3 in 2010 several challenges have been mastered to establish FESA3 on a site different than CERN. These efforts not only include the technical issues mentioned in this paper but also constituting a development environment that involves comprehensive information and documentation for FESA3 equipment software developers.

In the past few years it has been shown that the flexible and modular approach of FESA3 supports the adaptation to different sites and environments.

The essential parts of the FESA3 framework have been established up to now. For the future several issues remain to improve the stability and usability of the FESA3 software at the involved sites.

REFERENCES

- [1] FAIR website: <http://www.fair-center.de>
- [2] J. Fitzek et al., “Settings Management within the FAIR Control System based on the CERN LSA Framework”, WEPL008, PCaPAC’10, <http://www.jacow.org>
- [3] V. Rapp et al., “Controls middleware for FAIR”, WCO102, PCaPAC’14, <http://www.jacow.org>
- [4] M. Kreider et al., “Launching the FAIR Timing System with CRYRING”, TCO304, PCaPAC’14, <http://www.jacow.org>
- [5] CEA website: <http://www.cea.fr/english-portal>

THE FAIR R³B PROTOTYPE CRYOGENICS CONTROL SYSTEM

H. Simon, T. Hackler, C. Betz, E. Momper, M. Zaera, C. S. Schweizer, M. Stern,
D. Sánchez-Valdepeñas, GSI Helmholtz Centre for Heavy Ion Research, Darmstadt, Germany

Abstract

The superconducting GLAD (GSI Large Acceptance Dipole) [1] magnet is one of the major parts of the R³B (Reactions with Relativistic Radioactive Beams) experiment. The cryogenic operation will be ensured by a fully refurbished TCF 50 cold box and oil removal system. One of the major design goals for its control system is to operate as independent as possible from the magnet control system. The cold box control system is seen as a first prototype for the later cryogenic installations in the Facility for Antiproton and Ion Research (FAIR). The operation of the compressor, oil removal system, and the gas management was successfully tested in January 2014. Within late winter 2014 a first cool-down of the refurbished cold box is planned. Once the magnet will be delivered, the magnet and the cryogenics controls will be commissioned together. To do all these learning and realization steps can be seen as preparatory work for novel industrial control systems to be established at the FAIR facility [2].

INTRODUCTION

The FAIR R³B Prototype Cryogenics Control System comprises a fully refurbished TCF 50 cold box and an oil removal system from DESY (Deutsches Elektronen-Synchrotron).

The cold box tubing has been modified in order to meet the requirements of the cryogenic process being adapted to the GLAD magnet operation. In the refurbishing and upgrading process, an outlet was added to the shield cooling of the superconducting coil. Also all sensors and actors have been especially chosen in order to test and select possible equipment for the later FAIR installation. This included two versions of actors for the valves and passive pressure sensors which are apparently more radiation resistant. The full instrumentation of the cryo plant has been renewed and a new control cabinet replacing the previous controls has been installed.

The control system development process is following a staged implementation. First step was to understand, to design and to implement all process functionality for compressor and oil removal system inside a S7-319F with PROFIBUS and PROFINET I/O modules using WinCC OA as SCADA platform. In the second step the program logic designed in step 1 has been successfully migrated to a new version based on the CERN Unified Industrial Control System (UNICOS) framework [3]. This was the first time at GSI, that UNICOS has been used. As next steps there is the design and implementation of all algorithms and control parameters needed for the cold box processes (cool-down, shield/magnet supply and warm up) foreseen.

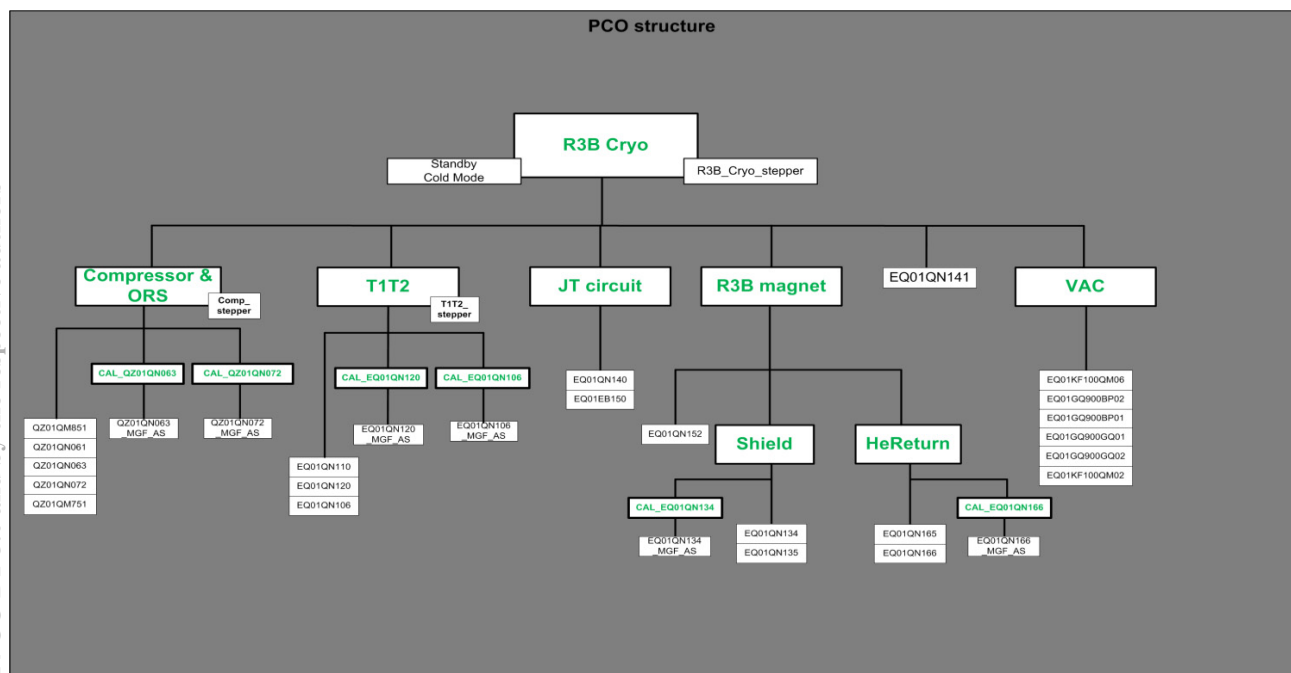


Figure 1: UNICOS unit (PCO) structure of the R³B facility.

METHODS

Programming Tools

To develop the R³B cryogenics three-layer control system, UNICOS' Continuous Process Control (UCPC) framework is chosen for FAIR. The framework is needed additionally to the development tools of the PLC and SCADA suppliers, but the framework offers a lot of additional opportunities, which are necessary in order to deal with huge and complex installations through a small team.

For that purpose UCPC provides a set of standard objects. Those objects cover all process items, from I/O-channels to high-level compound objects like sub parts of the facility and equip them with further logic.

Next to that it is also an approach to model a process in units (IEC-61512) as well as a generic environment to develop the applications specific control logic. To develop code UNICOS provides fully object oriented tools to generate code for all basic and project specific functionalities automatically.

The code generation is based on a particular project related xml database. Inside this xml database all object relations have to be defined, together with the general features and behaviours of all needed objects, combined with the desired look and feel at the SCADA level.

The basic functionalities, which are fully implemented directly after the first automatic code generation as well for the PLC as also for the SCADA system, allow an easier commissioning phase because all devices can directly be operated from the SCADA without need of a final panel design and implementation.

After a successful commissioning of a certain unit, there is no problem to hand all code to units of the same type.

Next to automatically generated code user specific code can be added to the framework in an object orientated way, too. This code has to be inserted into allocated files, which are provided through the framework and coded in Python. These files foresee the possibility to add own PLC (e.g. SCL) code at certain places.

To generate the PLC source code and the SCADA tag database nothing else, than the project related xml database user specific code and the automatic generation tool UNICOS Application Builder (UAB) are needed.

Because there is the possibility to enlarge the system through adding additional objects or user code in several steps or iterations, the system design is exceedingly flexible.

System Network

The R³B cold box can be remote-controlled via the WinCC OA SCADA system. For this purpose the S7-300 PLC is equipped with an additional network card (CP343-1 Adv.) for all communications to WinCC OA. Besides the system also can be controlled by a touch panel (TP700C), mounted on the side of the cabinet, which is connected via Industrial Ethernet to the integrated Ethernet card of the PLC.

Regarding the implementation of the control system two fieldbus systems are used in order to connect all modules and sensors: PROFIBUS for valve controllers, the Gantner remote I/O station with the main part of analogue input/output modules, the vacuum pump controller and the Janitza power analyser; and PROFINET for the Beckhoff remote I/O station, the Festo pneumatic controller station, the compressor control unit.

The large diversity of different suppliers and modules for remote I/O hardware used in the R³B cold box system comes from the idea to test them all as prospective for the FAIR cryogenic system. It is planned to focus for these systems only to PROFINET as fieldbus system and not more than two different types of remote I/O stations with a small number of different module types.

System Structure

Figure 1 shows the tree structure of the UNICOS objects of the R³B cooling system. Outgoing of the root (R³B_Cryo), are several logic units/objects needed, which manage an individual task each. These units are the compressor and oil removal system, the turbines, the Joule-Thomson valve, the R³B magnet as well as the vacuum system. Related to these knots are subsidiary objects like related sub parts (heat shield/ Helium return) or valves. It is foreseen to implement at the SCADA level panels for each units of the object structure a panel which shows all related devices/objects, like it is exemplarily shown in figure 2.

UNICOS itself already provides for WINCC OA and WinCC flexible libraries in the supervision layer an interface for: import/export the configuration, all widgets (a summarized view of the object), faceplates (detailed view of the object), the trend configuration panel etc. In figure 2 is shown the control panel in WinCC OA including several widgets and as well one controller device faceplate as also the faceplate for the compressor itself.

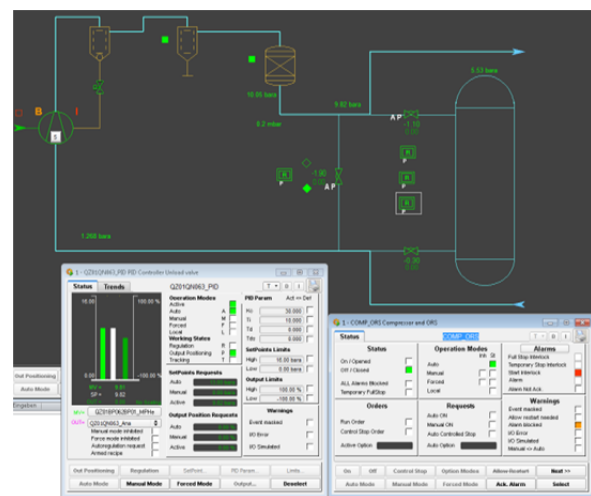


Figure 2: Control panel view developed for the R3B.

Hardware

For data acquisition the Beckhoff PROFINET remote I/O station, especially with the analogue modules 3602 and 3612 are used in order to be tested. They have a 24 bits resolution channel, which let us have accuracy measurements with low cost, as well as to test the sensor precision.

The Festo precise proportional pressure regulators are used to control regulated valves, which are foreseen to work in environments with radiation at the FAIR facility. The idea is to replace the valve positioners through this type of subsystem. The feedback information for the position of the valve is the resistance of a potentiometer coupled to the valve axis. A limit calibration for the possible max./min. values of the resistance is made every time the system will be started to ensure the accuracy of the measurement.

Because of the low working temperatures to get liquid helium, the accuracy of the temperature measurement causes several problems. To solve them the Gantner remote I/O station with their analogue modules are used [4], the station offers the possibility to program a polynomial function in order to calculate the temperature. The sensor used to measure the temperature is a diode with a characteristic curve shown in figure 3. The calculation of the temperature is divided in two functions dependent to the input signal.

In case the input signal is bigger than 1.1135 V, in relation to temperatures equal or bigger than 30 K. The corresponding polynomial is:

$$t = f(x) = 434.17 - 363.74x$$

In case the input signal is smaller than 1.1135 V, in relation to temperatures smaller than 30 K:

$$t = f(x) = 948.02 - 2527x + 2729x^2 - 1452.3x^3 + 378.93x^4 - 38.844x^5$$

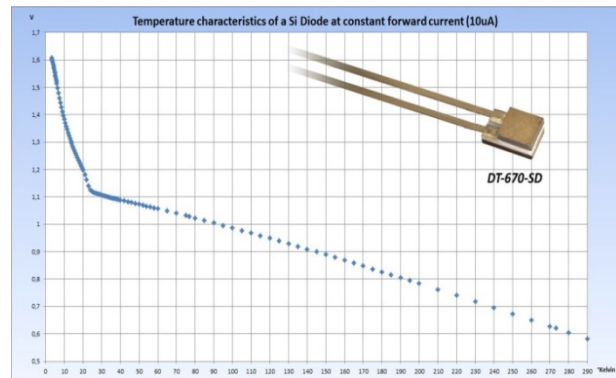


Figure 3: Diode sensor DT 670-SD characteristic curve [5].

The power analyser Janitza is to measure the performance (cooling capacity) of the complete cold box system. As soon as the system is completely cooled down, the needed electrical power for the heater of the liquid He tank inside the cold box correlates to the produced cooling capacity of the cold box system, in case the liquid helium level is stable. So the regulation of the heater is critical for the accuracy of the measurement. The signal send to the heater is monitored, the control signal is pulse-width modulated and transferred via a solid-state relay.

CONCLUSION

From the controls point of view for several problems is to show how to solve them: the accuracy the measurements needed to the process; the selection of devices according to the environment conditions at the FAIR facility; the compatibility requirements between the components of different suppliers and their availability in the market for the next years.

REFERENCES

- [1] Institute of research into the fundamental laws of the universe: <http://irfu.cea.fr/en/index.php>
- [2] H. Simon, "Status of the R³B GLAD magnet cryo plant", GSI, Germany (2014).
- [3] Cern.ch, Engineering department website: <https://j2eeps.cern.ch/wikis/display/EN/UNICOS>
- [4] Gantner-Instruments, DB.Qbloxx Modul A107-09, <http://www.gantner-instruments.de/>.
- [5] E. Momper, Figure 3

AN EXTENSIBLE EQUIPMENT CONTROL LIBRARY FOR HARDWARE INTERFACING IN THE FAIR CONTROL SYSTEM

M. Wiebel, GSI, Darmstadt, Germany

Abstract

In the FAIR control system the SCU (Scalable Control Unit, an industry PC with a bus system for interfacing electronics) is the standard front-end controller for power supplies. The FESA-framework is used to implement front-end software in a standardized way, to give the user a unified look on the installed equipment. As we were dealing with different power converters and thus with different SCU slave card configurations, we had two main things in mind: First, we wanted to be able to use common FESA classes for different types of power supplies, regardless of how they are operated or which interfacing hardware they use. Second, code dealing with the equipment specifics should not be buried in the FESA-classes but instead be reusable for the implementation of other programs. To achieve this we built up a set of libraries which interface the whole SCU functionality as well as the different types of power supplies in the field. Thus it is now possible to easily integrate new power converters and the SCU slave cards controlling them in the existing equipment software and to build up test programs quickly.

INTRODUCTION

As GSI is building up the FAIR [1] project and thus doing renovations all over the facility, it was decided to build up a new control system for the accelerator. This is done as a collaboration project with CERN. As part of the new system, the FESA (Frontend Software Architecture [2]) framework deals with all frontend related tasks.

To integrate our numerous hardware designs with the framework, we decided to build up the FESL (Front-End Support Library) as a lightweight approach to implement flexible equipment interfacing.

This paper sketches the workflow developing a FESA class and describes the challenges resulting from it. As a consequence the requirements to the FESL are depicted, followed by the description of the resulting structure of our library. As a last part we discuss the usage of FESL in the context of the FESA framework und give a short outlook to the future development of the library.

USING THE FRONTEND CONTROLLER

As described in [2] the development of a FESA class follows a specific workflow leading to a ready to use equipment software. After designing the class in an XML

based document, one can automatically generate a set of C++ source code frames. These frames are filled with specific implementation and are compiled to a ready to use FESA class. In the deploy unit one or more FESA classes are linked with the run-time core to build an x86-Linux executable. This executable can then be delivered to a front-end computer of choice.

The FESA framework is designed to be flexibly tailored to the broad range of equipment in the accelerator, but due to its rather long development cycles, it lacks the flexibility needed during an early development phase. Especially in our case, as we often have several variants of the devices. Writing test software for different power supplies forced us to walk through the aforementioned development process over and over again. Several only slight differences in the equipment behavior, led to an unwanted overhead of work and time.

REQUIREMENTS TO FESL

To cope with the above described limitations, we tried to decouple the equipment specifics from the implementation of the control system specific parts. Thus we came up with several requirements we had for our Front-end Support Library.

First of all it should unify and simplify the usage of the

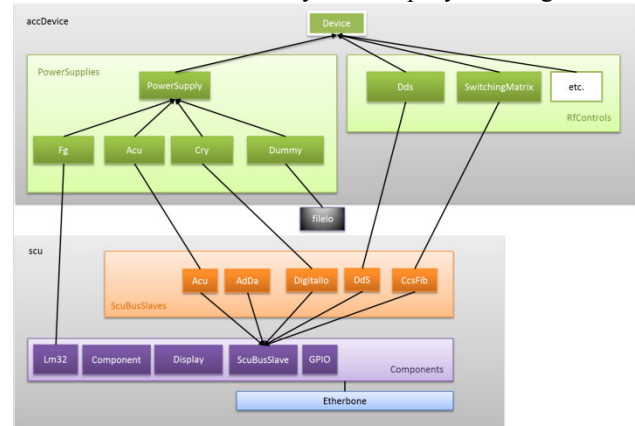


Figure 1: Overall structure of the FESL.

different power converters we are addressing through our front-end computers, namely the SCU (Scalable Control Unit [3]). Being an industry PC with a bus system for interfacing electronics, the SCU is used as the standard equipment interfacing in the FAIR project. Power supplies and other equipment are connected to it using slaves of an internal bus.

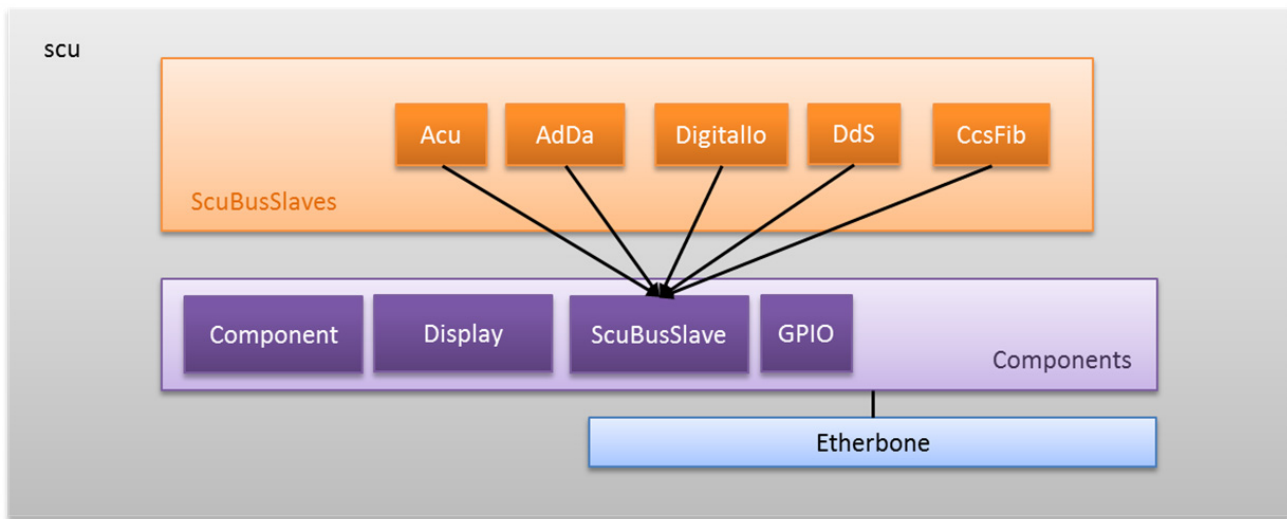


Figure 2: The SCU part of the FESL.

Easy Testing

To ease the testing of the connected devices and to decouple it from the need of embedding the equipment into the control system, we split up the library into two different parts (figure 1): One part to be used from inside the FESA source code frame (*accDevice*) and the other part to be used for dealing with the internals and the slaves of the SCU directly (*scu*). The *accDevice* part builds upon the *scu* part in a way, that it combines parts of it to provide a more unified look on it from the FESA class.

No Need to Know About Initialization Details

As a second point, we wanted to hide all the SCU internal communication mechanics from the user of the FESL. As a consequence, we decided to do all the initialization of the different SCU bus slaves as far as possible in an automatized way.

Follow the Structure of the Hardware

Furthermore it was important for us, that the library could be used in a very intuitive way. Thus the overall design of the FESL reflects the structure of the hardware found in the context of the SCU.

STRUCTURE OF FESL

Guided by our requirements, we designed the library along the hardware to be represented. This led to an intuitive and straight-forward model of our front-ends. As mentioned before, the library is split up into two parts, the *scu* part and a more FESA oriented *accDevice* part.

SCU Part of the Library

Figure 2 shows the SCU part of the FESL. The blue box represents the etherbone bus [4], which is used to communicate with all components of a SCU. On top of that comes the part of all SCU internal components,

starting with the base class for all of them (*Components*). By creating a *Components*-derived object all needed initializations are done. This makes it easy to create new component classes without deeper knowledge of the SCU. Components which can be created so far are

- Component (base class)
- Display (internal display of the SCU)
- ScuBusSlave (base class for all SCU bus slaves)
- GPIO (general purpose IO)

Inheriting from *Component*, *ScuBusSlave* is the base class for all cards which can be connected to the SCU via the backplane. Again, the base class does all the initialization needed to establish the connection.

AccDev Part of the Library

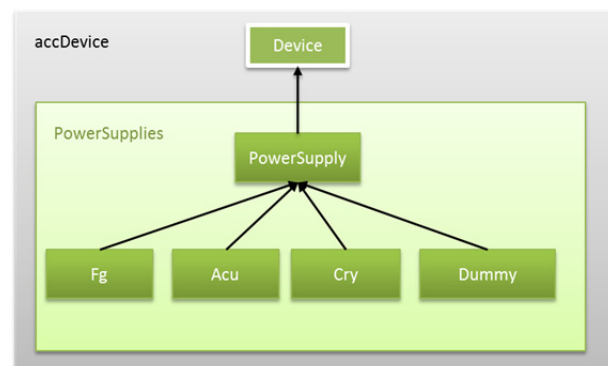


Figure 3: Device part of the FESL.

The *accDev* part builds on the SCU part and combines the different bus slaves to real devices located in the control system. Up to now, two types are supported:

- PowerSupplies
- RFcontrol

According to the idea that we wanted to be able to control different variants of power supplies with one FESA class (figure 3), they are all derived from the same base class *PowerSupply*. This base class can be used throughout the whole equipment software as an abstract interface. During FESA instantiation it is decided which power supply is actually connected and which implementation should be used.

Fg implements the interface for power supplies driven by a function generator, while *Acu* interfaces the so called Adaptive Control Unit for ramped power supplies. *Cry* again handles a different variant.

The *Dummy* class is used when there is no real power supply to control. It reads from and writes to the hard disk.

INTEGRATION WITH FESA

With the *accDev* part of the FESL at hand, we are now able to easily exchange the different device variants used in a FESA class. This allows us to operate multiple variants of power supplies from within one and the same equipment class.

Instantiating the Power Supply

During start-up, the FESA class reads the type of the power supply from the so called instance file (an XML file containing all the configuration details for a specific device instance). With this information it instantiates the according power supply from the FESL, which is then used in the whole class via the *PowerSupply* interface. So, what we did is injecting different equipment variants using the configuration file.

CONCLUSION

With the introduction of the FESL we were able to enhance our way of developing new equipment software to a level that is now as flexible and lightweight as we wanted. FESA as a framework for equipment interfacing and integration into the control system, is still the way to go, but can now be used with the presented flexibility in the development process.

It's now much easier to quickly write programs to interact with the hardware. Furthermore we can extend our library and thus can use new power supply variants with one and the same existing FESA classes, which saves a lot of time.

REFERENCES

- [1] FAIR website: <http://www.fair-center.de>
- [2] Solveigh Matthies et al., "FESA 3 Integration in GSI for FAIR", WPO006, Proc. PCaPAC'14, <http://jacow.org/>.
- [3] S. Rauch et al., "Facility-wide Synchronization of Standard FAIR Equipment Controllers", WEPD48, Proc. PCaPAC'12, <http://jacow.org/>.
- [4] M. Kreider et al., "Etherbone – A Network Layer for the Wishbone SoC Bus", WEBHMULT03, Proc. ICALEPCS'11, <http://jacow.org/>.

AN OPTICS-SUITE AND -SERVER FOR THE EUROPEAN XFEL

S. Meykopff, DESY, Hamburg, Germany

Abstract

For the European XFEL and the upgraded FLASH facility we require a tool for beam optics calculations. A newly developed software library manages accelerator parameters and performs beam dynamics calculations. In addition a server offers an interface between the library and the control system. A MATLAB interface allows convenient access to the optics server. This framework provides an online model which is integrated in the control system. The online model is used for a simulated European XFEL environment with realistic control interfaces. We use this environment for extensive software developments and tests.

At the FLASH facility a MATLAB toolbox is in use for the calculation of optics parameters [1, 2]. The FLASH2 update adds a second undulator beam line to the facility. The second beam line is not supported in the MATLAB toolbox and must be considered in a successor. Currently the European XFEL is in construction which has up to 5 beam lines [3]. A beam optics tool will be required for commissioning and routine operation. We decided to develop a code for both facilities. This code will provide the accelerator layout in conjunction with the different beam lines. The code calculates transport and response matrices and allows fitting of optics parameters. The matching of beam parameters will be an important feature of the new code. To provide clearly arranged software the new code is developed as a library.

OPTICS LIBRARY

The description of the accelerator is stored in the optics library and will be initialised from a SQLite database [4]. In the case of the European XFEL the SQLite database is generated from an Excel sheet which is distributed by the machine layout coordinators [5]. The sources of the FLASH description are some MAD8 output files [6]. In both cases the generation of the database will be done with MATLAB. The optics library provides these descriptions as static data. The static data includes element names, type names, positions, and covers all columns of the Excel sheet.

The design of the optics library provides multiple setups in the same time. These setups cover variable parameters like steerer magnet angles or k values of quadrupole magnets. The setups are kept separated with a unique setup name. Each setup is independent.

An external code is called for the beam dynamics calculation. Currently the code ELEGANT is in use [7]. To start an ELEGANT run a lattice file and a command is required. The static information and the current setup information are used to generate both files. The output format of elegant is SDDS. The SDDS files were read from the library and stored internally. A dispatch queue

collects all ELEGANT runs and executes them parallel. The code of the optics library is written in C. The design of the interface allows one to use it as a shared library.

OPTICS SERVER

The optics server offers a connection to the control system. The TINE interface was chosen as control system because it's in use at both facilities. The optics server provides all functions of the optics library. Every optics server call includes the setup name. This constraint allows a multi-user run with independent setup parameters.

The layer between the control system and the optics library is thin (see figure 1 for details). Mostly parameter checks and conversion is implemented in this layer. A major addition is the observe module. The observe module checks for updates of the RF system or changes of power supply currents. On demand a defined setup will be updated with these values.

We provide a virtual beam position monitor server for tests with simulated orbits. The orbit will be set by a call from the control system. In the optics server is a small push orbit module. This module delivers the simulated orbit to the virtual beam position monitor.

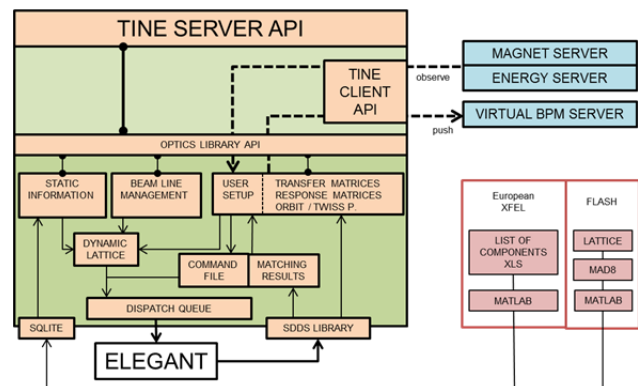


Figure 1: Optics server internals: TINE interface to clients and connections to other servers. Optics library internals with information databases, file generator, job dispatcher, access to SDDS output, and SQLite database as source.

INTERFACE TO MATLAB

In the DESY control room MATLAB is an important tool. The XOptics MATLAB object is a convenient interface to the optics server. The lower interface will be provided by XCOMM. XCOMM is a unified interface for TINE and DOOCS control systems (overview in figure 3) [8]. All functions of the optics server are covered by the XOptics object. MATLAB offers a tabulator expansion for object methods and properties. This feature is a notable simplification for the user of XOptics. Figure 2 shows a code example.

```
% get XOptics object:
x = XOptics();
% get all members of beamline I1D:
ild_member =
    x.ListOfBeamlineMembers('I1D');
% fetch all informations:
info = x.ElementSetupInformation(
    'DEFAULT','I1D', ild_member);
% plot beta x:
plot([info.z],[ info.betax]);
```

Figure 2: How to plot optics data in MATLAB.

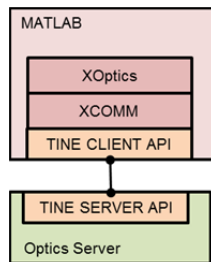


Figure 3: MATLAB interface to the optics server.

USE CASE: VIRTUAL XFEL

At DESY a large environment was built to develop and test software for the European XFEL. This environment includes elements of low-level, middle-layer and high-level controls software, and some hardware. It covers the timing system, the toroid server, the data acquisition server, the orbit server, the sequencer, the magnet server and the power supply servers. The test environment includes also software which runs in the control room like the display panels, the orbit bump tool and the emittance measurement and matching tool (see figure 4). Most of the software components are used with little or no modifications with respect to the production versions. Our environment allows us to test software components before the commissioning of the European XFEL.

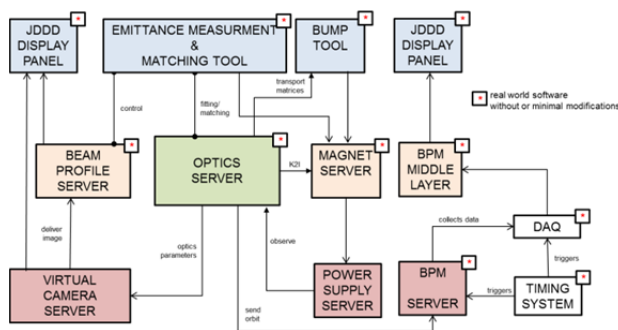


Figure 4: Components of the Virtual XFEL. High level applications on top, middle layer server and on the bottom front end server.

Test of Orbit Bump Tool

One test case is the orbit bump tool. The tool queries transport matrices from the optics server. The calculated bump angles are written to the magnet server. The magnet server uses the optics server to convert the angles into currents. The magnet server delivers the new currents to the power supply server. The new currents are read by observe module of the optics server. The optics server recalculates the orbit and pushes the new orbit data into the virtual beam position monitor server. The DAQ collects the new positions with the help of a trigger signal of the timing system [9]. The collected data will be read from the beam position monitor middle layer server. The standard tool JDDD displays the new beam positions, showing a closed bump if everything tested successfully [10].

USE CASE: EMITTANCE MEASUREMENT AND MATCHING TOOL

At the FLASH facility we developed a beam profile server. This middle layer server offers emittance measurements. The server handles all different interfaces of the cameras and wire-scanners. If required the server switches the laser on or off. There are routines to handle the background noise and to calculate some beam parameters. The “Emittance Measurement and Matching Tool” was developed to operate the middle layer server. The tool was developed in MATLAB and allows starting a measurement with live view of the cameras. The tool plots the measurement results and displays some beam parameters like the emittance and BMAG (see figure 5). The fitting of the beam parameters are done by the optics server. The tool offers the matching of the twiss parameters. The matching will be done by the optics server. The tool displays the advised magnet currents and pushes the values to the power supplies one demand. A new measurement with the updated magnet currents should yield optimized beam parameters.

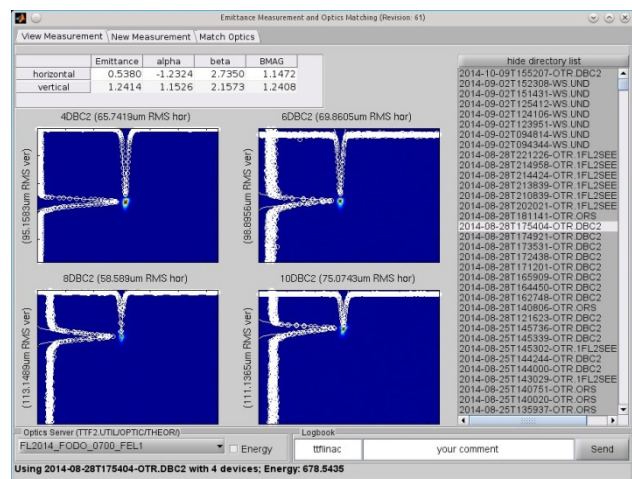


Figure 5: Screenshot of the “Emittance Measurement and Optics Matching Tool”.

CONCLUSION

The optics library is an easy to use and powerful interface to do beam optics calculations. The optics server offers all library functions in the control system. The full-featured interface in MATLAB is a valuable tool in the control room allows the development of nice applications. The integration in the control system enables the installation of an unprecedented development and test environment. The developers are independent of a real accelerator. The time schedule of the commissioning of the European XFEL will be tight. Currently we are able to develop and test software before the commissioning. We expect a smoother commissioning phase with our already tested software.

REFERENCES

- [1] Free-electron laser FLASH <http://flash.desy.de/>.
- [2] V. Balandin, N. Golubeva, Matlab, "Functions for Calculations of the Linear Beam Optics of FLASH Linac", <http://ttfinfo.desy.de/TTFelog/data/doc/Physics/Optics/2008-02-15T13:59:48-00.pdf>
- [3] European X-Ray Free-Electron Laser Facility GmbH, <http://www.xfel.eu/>.
- [4] SQLite, <http://www.sqlite.org/>.
- [5] European XFEL – list of components, <http://www.desy.de/fel-beam/>
- [6] MAD8, <http://cern.ch/mad8>
- [7] M. Borland, "elegant: A Flexible SDDS-Compliant Code for Accelerator Simulation", Advanced Photon Source LS-287, September 2000.
- [8] J. Wilgen, S. Meykopff, "XCOMM A Unified MATLAB API for TINE and DOOCS", Proc. PCaPAC2014, <http://jacow.org/>.
- [9] V. Rybnikov et al., "FLASH DAQ DATA MANAGEMENT AND ACCESS TOOLS", Proc. PCaPAC2010, <http://jacow.org/>.
- [10] JDDD, A Java DOOCS Data Display, <http://jddd.desy.de/>.

A UNIFIED MATLAB API FOR TINE AND DOOCS CONTROL SYSTEMS AT DESY

J. Wilgen, S. Meykopff, DESY, Hamburg, Germany

Abstract

At the European XFEL, MATLAB will play an important role as a programming language for high level controls. We present xcomm, a standard MATLAB API which provides a unified interface for TINE and DOOCS control systems. It supports a wide variety of data types as well as synchronous and asynchronous communication modes.

DOOCS AND TINE

The two main control systems at DESY, DOOCS [1] and TINE [2], have been in operation for decades at different accelerators at DESY. The TINE control system, originally created for the HERA collider, is in operation at PETRA, DESY2/LINAC2, and REGAE [3]. The DOOCS control system operates the FLASH accelerator [4]. For the European XFEL, ongoing efforts are being made to integrate both control systems [5].

MATLAB IN CONTROLS

During the past years, MATLAB has become increasingly popular for application development in DESY Control Systems. MATLAB programs are in operation at FLASH, PETRA, and REGAE. At the European XFEL, MATLAB will be used as a major programming language for high level control applications.

Several MATLAB APIs for DOOCS and TINE already exist at DESY, with different scopes and interfaces. Each of them supports only subsets of the data types and the features of the control systems. Some are platform dependent. For the high level controls at European XFEL and FLASH, it was required to have a common, complete and well-tested API with an intuitive and robust interface. Therefore we decided to build a new, unified MATLAB API.

XCOMM FEATURES

Communication Protocol

Xcomm uses the TINE protocol to communicate with both TINE and DOOCS servers. DOOCS servers have already integrated TINE by means of a built-in adaptor [6]. When the adaptor is enabled, a DOOCS server automatically becomes visible in the TINE name space and can be accessed by any TINE client as shown in Fig. 1. Relying on TINE, xcomm can therefore serve as a common MATLAB API for accelerator control systems at DESY. An exception is the Karabo control system [7] of the European XFEL undulator beamlines, which is currently not supported, but is planned to be accessed through a TINE gateway, as far as needed.

Simple Interface

Xcomm is implemented as a MEX (MATLAB Executable) function in the C programming language. It is called as a function which can be controlled by a variable list of parameters, as common in MATLAB. Using a single function interface, programs can send data to or receive data from any control system address. The complexity of the underlying client API is hidden from the user.

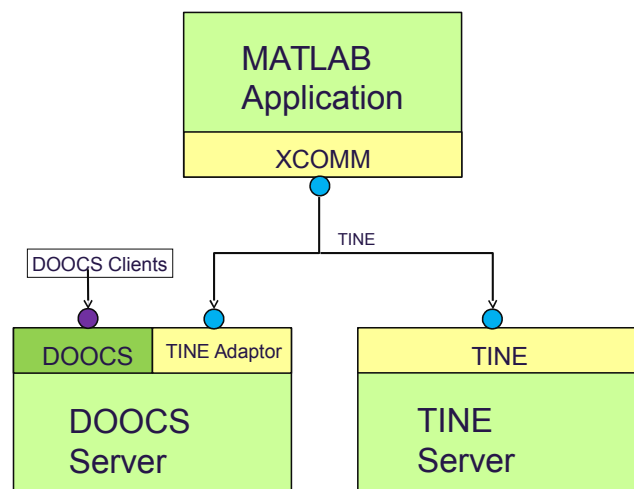


Figure 1: Communication with different servers.

```
result = xcomm('<target address>'
    [, <outputData>]
    [, 'optionName', optionValue, ...])
```

Both the DOOCS and TINE address styles are supported. The programmer does not have to know if the connected server is a DOOCS or TINE server.

The result of an xcomm call is a structure array which includes an error string, a time stamp in both MATLAB and text formats, and the received data. Since the actual content of the data depends on the request, the data may contain a scalar value, an array, a structure array or a cell array of any of these, dependent on the data type and the number of data units returned. Data sent to the server must be formatted in the same way.

Multiple Data Types

TINE and DOOCS have many pre-defined structure types which xcomm supports. A structure type is mapped to a corresponding structure array in MATLAB.

TINE servers can also have properties with user-defined data types. Xcomm can handle these data types

and maps them in the same way as pre-defined structure types.

Special types like images or multi-dimensional arrays have specific raw formats which need to be converted for further processing. An image, for instance, should be represented as a matrix in MATLAB. Xcomm converts such data automatically so that the results can be used directly. Still the raw data remain available.

Asynchronous Communication

In addition to synchronous communication, TINE offers different communication schemes [2], and makes extensive use of asynchronous communication in order to reduce the network load and the CPU load of servers. On the client side, this means that asynchronous events need to be processed when data arrive. Since MATLAB programs are single-threaded and typically have a synchronous data flow, xcomm uses a feature of the TINE client library to handle incoming data in a background thread. An xcomm call which uses an asynchronous connection reads data from a local buffer which is updated automatically. As shown in Fig. 2, this makes synchronous and asynchronous communication simple and transparent to the programmer, since it does not require different programming models. Still, callbacks may be added in a future version as an extension for special cases where either latency is important or to reduce unnecessary local polling.

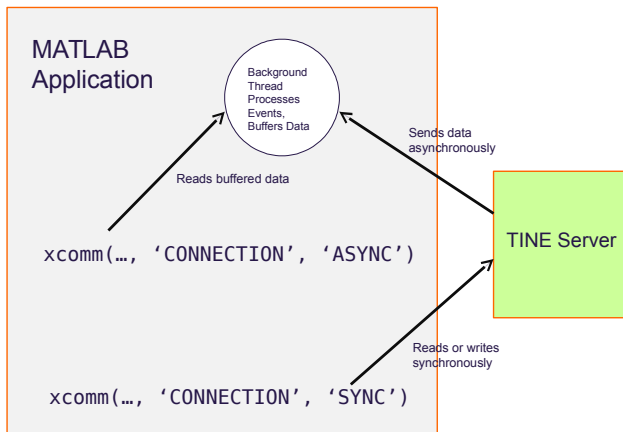


Figure 2: Synchronous and Asynchronous Communication.

Bulk Operations

Xcomm also allows to bundle multiple requests into one call, returning a set of results.

```
result = xcomm({'<address1>', 'address2'},
               [, 'optionName', optionValue, ...]);
```

where result is a struct array of partial results. Bulk operations can be useful in cases when multiple properties are needed at once.

CONCLUSION

Xcomm is a new MATLAB API to access TINE and DOOCS control systems. First applications have already been developed for FLASH and the European XFEL. We expect that a common standard API will improve the quality and maintainability of programs and make more control system features available to MATLAB programmers. High level control applications developed for the European XFEL will use xcomm as the standard interface.

REFERENCES

- [1] DOOCS. <http://doocs.desy.de>
- [2] TINE. <http://tine.desy.de>
- [3] R. Bacher, P. Bartkiewicz et al, "TINE: The Control System for PETRA3 and the DESY Preaccelerators", ICFA Beam Dynamics Newsletter, No.47, 2008.
- [4] A. Aghababian et al., "The Accelerator Control Systems at DESY", ICFA Beam Dynamics Newsletter No. 47, 2008.
- [5] P. Duval, A. Aghababian, O.Hensler, K.Rehlich, "Control System Interoperability, an Extreme Case: Merging DOOCS and TINE, PCaPAC, 2012
- [6] K. Rehlich, Control System Interfaces, XFEL Collaboration Meeting, 2013
- [7] B. C. Heisen et al., "Karabo: an integrated software framework combining control, data management, and scientific computing tasks, ICALEPCS, 2013

VACUUM INTERLOCK CONTROL SYSTEM FOR EMBL BEAMLINES AT PETRA III

A. Kolozhvari, U. Ristau and S. Fiedler

European Molecular Biology Laboratory EMBL, Hamburg Unit, Germany

Abstract

A vacuum interlock control system has been developed for EMBL structural biology beamlines at PETRA III synchrotron. It runs on Beckhoff PLCs and protects instruments by closing corresponding vacuum valves and beam shutters when pressure exceeds a safety threshold. Communication with PETRA III interlock system is implemented via digital I/O connections. The system is integrated in the EMBL beamlines control via TINE and supplies data to archive- and alarm subsystems. A LabVIEW client, operating in TINE environment, provides a graphical user interface for the vacuum interlock system control and data representation.

INTRODUCTION

Three EMBL beamlines for structural biology are in user operation at PETRA III synchrotron on the DESY site in Hamburg. Most of the beamline instruments operate in ultra-high vacuum, and therefore they need an efficient vacuum control system for protection against vacuum incidents. In this paper we describe the second version of the system that is fully integrated into the beamline control environment [1].

CONCEPTS

Controlled Object and Elements

The Vacuum Interlock Control System of EMBL at PETRA III, as any control system, evaluates and changes states of elements constituting the controlled object. The object is a vacuum system of a beamline divided into vacuum sectors, and the elements are vacuum gauges, ion getter pumps, vacuum valves, a front-end and a secondary beam shutter and vacuum sectors.

A sector usually includes X-ray optical instruments operating in ultra-high vacuum, pumps and pressure gauges. It is separated by vacuum valves from the adjacent sectors. A value of pressure in the sector may be obtained from the pressure gauges. Also, if the ion getter pumps are used, the pressure may be calculated from the pump current.

The vacuum valves are the main elements which can be actuated. The control system "by itself" is allowed only to close the valves; they may be opened only by an operator command.

Besides the valves, the control system supplies two permission signals to the Interlock Control System (ICS) of PETRA III, that commands the front-end and the secondary beam shutters of a beamline. In case of withdrawal of the permission signal the ICS must immediately close the corresponding shutter.

The system does not control the ion getter pumps. It only uses data from their controllers for evaluation of states of the sectors.

States of the Elements

A state of an element of the system is a combination of its "physical state" and its control mode. The control mode defines how the system should interact with the element. Three control modes are defined for the system elements: "automatic", "disabled" and "intervention".

The physical states defined for the vacuum gauges and ion getter pumps are: "OK", "Error" and "HV off".

The physical states of valves and beam shutters are: "open", "closed", "undefined" and "error".

The defined physical states of a sector are: "OK", "Warning", "Bad vacuum" - the pressure is above lower threshold, "Very bad vacuum" - the pressure is above ion getter pump shutdown threshold, and "Unknown".

ALGORITHM

The main task of the vacuum interlock control system is protection of the equipment in case of vacuum incidents. The system continuously monitors the pressure in all sectors of a beamline. If its value exceeds a predefined threshold, the system closes the corresponding valves and isolates the sector.

At each PLC cycle the following actions are performed:

- Reading of the vacuum gauge values and currents and high voltage states of the ion getter pumps. Update of their states and calculation of absolute pressure values.
- Permissions to open beam shutters are set to true. Their values will go to the ICS at the end of the PLC cycle.
- For each sector the system updates its state taking the worst pressure value reading from the gauges, and if they are not available then the one from the ion getter pumps. The system changes a beam shutter permission according to the sector state.
- For each valve, depending on its state and on the states of the connected sectors, the system decides whether the valve must be closed or not and closes the valve if necessary. After that, it checks for an operator command to open/close the valve. If it is possible, the system executes the command and acknowledges it. Otherwise, it sends a negative acknowledgement. If the valve participates in permission to open a beam shutter and if it is either open or disabled, the system keeps the previous value of a shutter permission. Otherwise, it sets the permission to false.

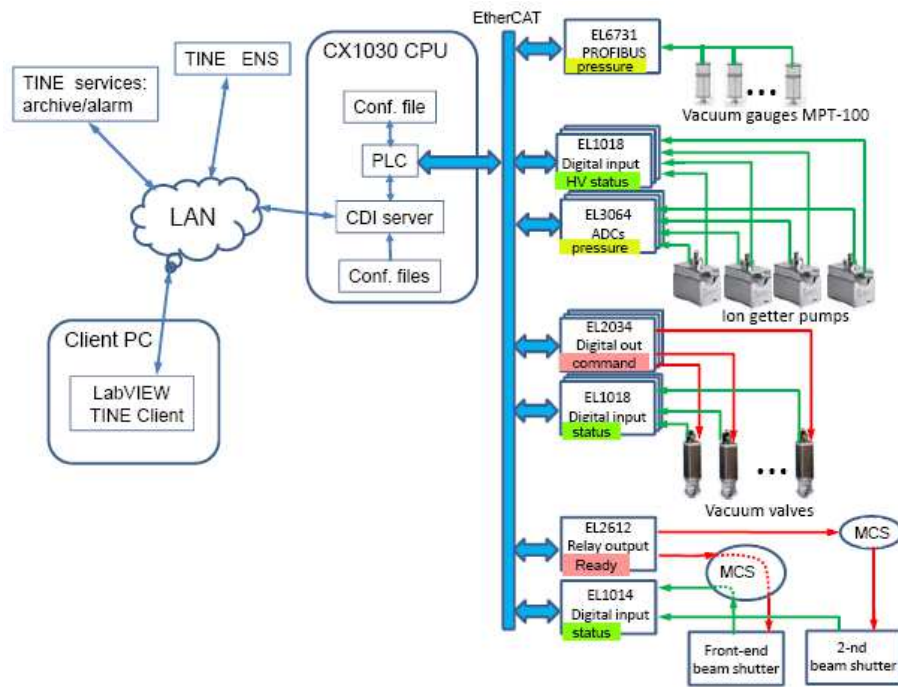


Figure 1: The general scheme of the vacuum interlock control system.

IMPLEMENTATION

The system runs on Beckhoff [2] embedded PC CX1030 CPU module with 32-bits 1.8 GHz Intel® Pentium® M processor, 992 MB of RAM, 7.6 GB compact flash card non-volatile memory, Microsoft Windows Embedded Standard 2009 OS and Beckhoff TwinCAT-2 software.

Basic Hardware Components of the System

- Pfeiffer vacuum gauges MPT-100 with range $5 \cdot 10^{-9}$ – 1000 mbar. They are connected via profibus interface to the Beckhoff master unit EL6731 for data read-out.
- Varian and Agilent ion getter pumps. The pump controllers provide pump current analog signals to Beckhoff 12-bits ADC EL3064 and logical signals HV on to Beckhoff digital input modules EL1018. The data are used by the system for the vacuum state evaluation.
- Different types of VAT vacuum gate valves with pneumatic actuators. They are controlled by 24 V/2 A EL2034 Beckhoff digital output modules. The end switches provide valve status information via the EL1018 digital input modules.

PLC Software and Interfaces

The PLC software is written in Structured Text language for Beckhoff TwinCAT-2 project manager. Two TwinCAT tasks are configured.

The main task performs vacuum monitoring and sector isolation in emergency cases. Also, it executes operator commands for “normal” opening and closing of the vacuum valves.

The second, “slave” task performs data exchange with the system configuration file stored in the local file system.

The general scheme of the system is presented in Fig. 1. The PLC code runs in CX1030 embedded PC and communicates via EtherCAT protocol with the Beckhoff hardware modules connected to the devices. Configuration of the system and parameters of the controlled objects, such as vacuum threshold and status, are defined in the configuration file. External communications are implemented via the chain: PLC code – TwinCAT – ADS protocol [2] – CDI server – TINE [3] protocol.

The CDI (Common Device Interface) server is used as an interface between ADS and TINE protocols. It is a component of the TINE toolkit. The server runs in the same PC as the PLC which simplifies the ADS communications. It sends data to TINE clients in asynchronous mode. The correspondence between TINE properties and the TwinCAT PLC variables is defined in one of the server’s configuration files.

Software Client

A TINE client is written in LabVIEW [4] to provide a GUI to control the system. The XControl technology is used. It allows easy adaptation of the GUI to the configuration of a specific system. The main code of the client is single-threaded. It is relatively simple and uses polling to obtain data from the server. The real asynchronous communications between the client and the CDI server are hidden from a programmer by the underlying TINE layer.

Several instances of the client may run concurrently to display the system status but only one may be enabled to control the system.

The client has 4 access levels:

1. monitoring: client performs only status display.

2. user: client allows the user to perform certain commands needed for normal operation (like venting and evacuating of certain sectors) provided that the action is safe.
3. operator: operations on all vacuum valves are permitted if the system considers them safe.
4. expert: commands changing the control modes of the system objects are permitted. It is potentially dangerous since it gives full access to the system.

Additional windows to display detailed information of each of the sectors and to change status of the objects can be opened with mouse clicks.

SUMMARY AND OUTLOOK

Currently, the vacuum interlock control system is deployed at one of the EMBL beamlines and is operational. Later it will be deployed at two other EMBL beamlines. It has more simple and convenient GUI and has more flexibility in control of the elements in comparison to the first

version of the system. Due to the used algorithms, the system minimizes the possibility of operator errors.

During the further operation, as more experience is gained, extension of the system functionalities like extrapolation of the pressure behaviour will be implemented. The structure of the software and the configuration procedure will allow to perform the modifications easily.

REFERENCES

- [1] U. Ristau et al., "The Cioncept of EMBL Beamline Control at PETRA III", MOZ02, Proceedings of PCaPAC08, Ljubljana, Slovenia (2008), p.22-24.
- [2] Site of the Beckhoff company:
<http://www.beckhoff.com/>
- [3] TINE web page at DESY:
<http://adweb.desy.de/mcs/tine/>
- [4] National Instruments Corporation:
<http://www.ni.com/labview/>

THE EMBL BEAMLINE CONTROL FRAMEWORK BICFROK

U. Ristau, A. Kolozhvari and S. Fiedler

European Molecular Biology Laboratory, EMBL, Hamburg Unit, Hamburg, Germany

Abstract

The EMBL operates three beamlines at the PETRA III synchrotron at DESY. The principal hard and software components of the beamline control will be described. A new framework, 'BICFROK' (BeamLine Control FRamework), that connects the various instrument control applications and is based on a LabVIEW-TINE environment, will be introduced. Examples for the implementation of the system will be described.

INTRODUCTION

For the control of the EMBL beamlines for structural biology at the PETRA III synchrotron on the DESY site in Hamburg, Germany, a hierarchically layered control software and industrial control electronics [1] have been implemented and are operational on all three beamlines. Key elements of the beamline control are the TINE control system suite 4.0 [2], the control electronic with EtherCAT fieldbus [3], the Beckhoff TwinCAT NC and PLC, and the LabVIEW software package [4].

The beamline instrument control is integrated with the BICFROK framework (BeamLineControlFRamework) by EMBL-Hamburg. Its purpose is to provide a graphical representation of the entire beamline enabling the beamline operator to access the instrument control applications in a well arranged manner and displaying monitor values of parameters relevant for the synchrotron beamline operation.

MASTER CONTROL UNITS

The instrument control of nearly all instruments at the beamlines is performed by a modular control software package and by standardized control electronics forming a master unit installation for each major component like monochromator, mirror or robotic sample changer.

An example of such a master control unit for the control of an X-ray focusing mirror is shown in Figure 1. One master unit comprises the control electronics modules and an embedded PC (CX1030/8GBRAM Pentium PC by Beckhoff). This PC hosts the control system installation, the device servers and the TwinCAT system including NC (Numeric Control) and PLC (Programmable Logic Controller) programs.

The amount and type of EtherCAT electronic modules in different control units varies with respect to the functionality needed for different instruments. Additional external, non-EtherCAT devices like the amplifiers selected for the beam position monitoring system and other TCP/IP based instruments can be integrated into the system if necessary.

Major tasks of the control units are encoded closed loop motion control, digital and analog signal processing, temperature monitoring and the timing of the beamline processes. With this design, the control of an entire beamline component can be integrated in one single unit.

At present, there are 27 master control units distributed in vicinity to the instruments inside and outside the beamline hutches. The local installation reduces the length of the cables from the instrument to the control unit and the problems associated with this. The distributed units are installed into closed and water-cooled housings to avoid the transfer of the heat produced by the electronics to the X-ray optical instruments - which are very sensitive to drifts caused by temperature variations.

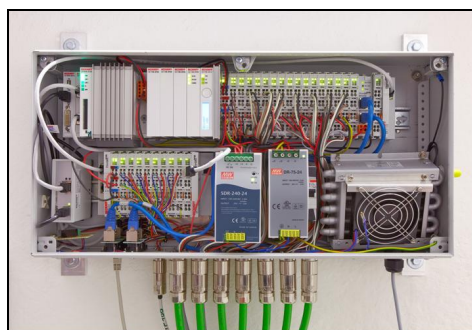


Figure 1: Example of a master control unit for an X-ray mirror instrument.

EtherCAT Master Installation

The soft real time fieldbus system EtherCAT is Ethernet based. For our applications, EtherCAT allows a cycle jitter of less than 1µs. The cycle frequency for PLCs varies between 100 Hz and 20 kHz. The Beckhoff EtherCAT module EL-6692 is installed for time synchronization between master control units. Master-to-master synchronization can be performed at the moment with a few milliseconds precision. With this concept, all beamline elements and signals like motors and encoder positions, intensities, vacuum pressure and temperatures monitors are synchronized with respect to each other. The synchronization speed is dependent on the EL-6692 electronics. The new generation of synchronization modules EL-6695 allows synchronization in the 100µs range and will replace the existing EL-6692 modules.

In order to achieve the best possible synchronization all programming that is critical for timing is performed at the real time PLC level of the control units. The read out of the amplifiers of the beam position signal, the control of the beamline undulator and the control of the mono-

chromator Bragg axis is also PLC based. This makes also fast on-the-fly scans and fast step scans possible.

Motion Control Electronics and DAQ

Currently, 150 motorized axes in operation at the beamlines are controlled by EL7041 Beckhoff EtherCAT modules for stepper motor control EL-7041. 150 EL-5101 counter modules are used to read out the encoder signal of the motor axes. Two servo motor axes are controlled by AX-5125 synchronous-asynchronous motor drivers. 24 piezo motor based translations stages (attocube [5]) and 12 piezo actuators (PI [6]) are controlled via EtherCAT.

DIO and AIO as well as counters and Profibus fieldbus communications are integrated into the system. Time stamp and oversampling EtherCAT modules are working in parallel as needed.

CONTROL SOFTWARE

The architecture of the beamline control software is shown in Figure 2. It allows access from all high level layers to all lower software layers and vice versa if sufficient access rights are granted for clients and servers. This design gives the necessary flexibility to ensure access to beamline control by different users. The TINE control system is the connecting element between all layers.

The TINE Common Device Interface CDI and the TINE Motor/Scan server are part of the TINE suite. CDI is the main low level device server. CDI is at the same time the common interface to devices and a server which exports the device functionalities as TINE properties.

The high-level generic TINE Motor/Scan server communicates with the low level CDI TINE server. It exports motion control and scan features as TINE properties.

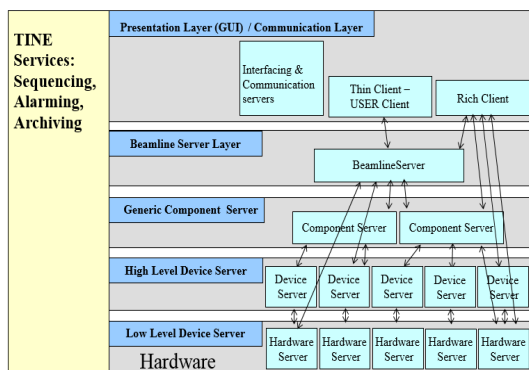


Figure 2: Layered control Software Architecture.

External communications are implemented via the chain: PLC code TwinCAT – ADS protocol [3] (Automation Device Specification) – TINE CDI server – TINE protocol. In this case the CDI server is used as an interface between TwinCAT and TINE protocol. This is a fast communication process because PLC, TwinCAT, CDI server and Motor/Scan server run all on the same PC and are seen to the outside as TINE servers.

Motor Server Architecture

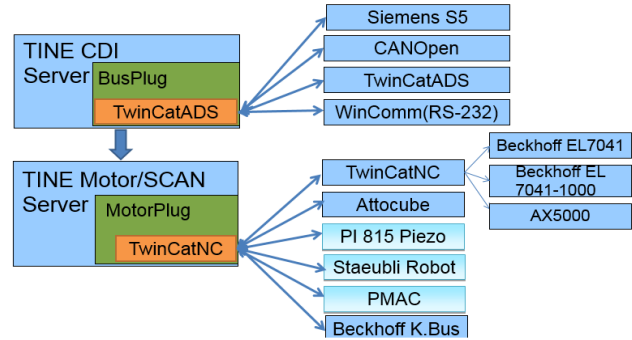


Figure 3: Motor/Scan Server architecture.

Figure 3 describes the interaction between CDI TINE server and the TINE generic Motor/Scan server – which, in this case, uses the TwinCatADS BusPlug and offers MotorPlugs for TwinCatNC motion control, attocube motion control and the support of PI[6] piezo drives. The integration of a MotorPlug for an industry robot (by company STÄUBLI) and a MotorPlug for PMAC controllers (by company DELTA TAU) is ongoing.

The generic Motor/Scan server supports all necessary features for initialization, axes definition (single or logical axes) different options for scanning, saving of scan data and scan read back for step scanning and continuous scans.

Closed loop motion control is implemented as a standard. Motors and actuators can be freely defined. The relative motion encoders are processed by digital counter inputs. The axis configuration and initialization on the client and server side is implemented by csv files.

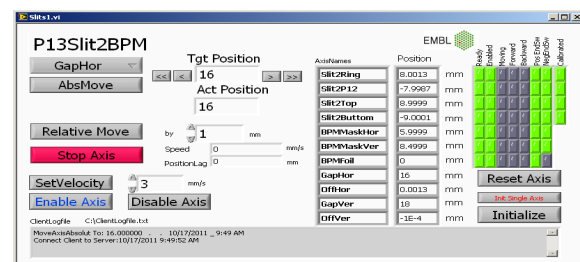


Figure 4: Generic standard motor client.

The standard expert generic motion control client (Figure 4) controls up to 15 motor axes. The client offers functionalities for the initialization of axis, command logging, error display, axis status display, relative and absolute positioning, display of velocity and position lag error and other basic features.

Beamline Control Framework BICFROK

The BICFROK framework is a tool set which supports the TINE features for data transport and central services like archiving of data and the alarm handling which are part of the TINE Suite 4.0. The BICFROK Container, shown in Figure 5, is the main user GUI for the control of the beamline instruments. The container grants access via

password protection for different user access levels. It acts as a window manager allowing access to the main beamline clients and offers a beamline overview and status display.

In the status display, the main beamline parameters and the relevant parameters from the PETRA III storage ring are displayed. Warnings and errors are presented and stored in a logfile. Device and component access is granted by selecting the element to be controlled on the beamline sketch with a mouse click.

Tab controls allow access to dedicated applications like the scan client, the beamline configuration tool or the X-ray beam viewers.

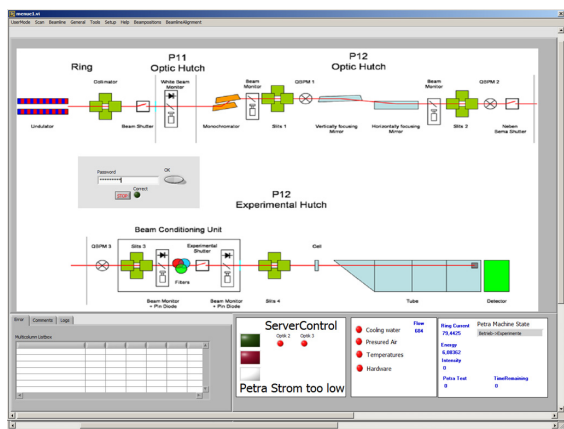


Figure 5: BICFROK Container.

Care has been taken to implement a modular software design following the example of the generic motion client.

Three different access modes are available for beamline operation: monitoring, operator and expert mode. In monitoring mode, the occasional beamline user can only view the main beamline parameters and status. In operator mode, the beamline operator has additionally access to commands to align and steer the beamline. Only in expert mode configuration files can be overwritten. Likewise, the generic motion client is available for motion control only in expert mode.

Software Environment

Servers are written in C/C++ and in G-Code using the LabVIEW framework. Clients are programmed with LabVIEW.

All servers have csv configuration and log files located in the Server directory of a component and are accessible via TINE calls. Configuration files define the access rights to servers, define the data to be archived and alarm levels etc.

The client log file stores the client command and access level history as well as status, warning and alarm logging. The SVN subversion system is used as code repository. Also the PLC code and files related to it is deposited in

SVN as well as the LabVIEW code with the help of the CVI tool, an element of the LabVIEW suite.

Hardware Environment

The TINE Suite 4.0 central services and the TINE name server are installed on two redundant Linux HP Proliant Rack PCs with redundant power supplies and mirrored hard discs offering failover functionality. The high and middle layer beamline control software runs on several PCs per beamline with raid1 mirrored hard discs. All data of the instrumentation PCs and PLCs are integrated into a backup system.

The standard operating system of the control computers is Windows 7 Professional. On all of the PLC computers Windows XP embedded is installed.

SUMMARY & OUTLOOK

The EMBL beamlines at PETRA III are in user service and the beamline control software for them is operational. However, since the beginning of 2014 a revision of the existing software is under way. The concept of EtherCAT based electronics, TINE as control system, LabVIEW and C/C++ for server/client programming will be continued and complemented by the comprehensive BICFROK framework. The system is still in the development phase and it is planned to be in operation in spring 2015.

REFERENCES

- [1] U. Ristau et al., "The Concept of EMBL beamline Control at PETRA III", MOZ02, Proceedings of PCaPAC08, Ljubljana, Slovenia (2008), p.22-24.
- [2] TINE web page at DESY:
<http://adweb.desy.de/mcs/tine/>
- [3] Site of the Beckhoff Company:
<http://www.beckhoff.com/>
- [4] National Instruments Corporation:
<http://www.ni.com/labview/>
- [5] attocube systems AG
www.attocube.com
- [6] Site of the PI Company:
www.physikinstrumente.com

STATUS OF THE FLUTE CONTROL SYSTEM

S. Marsching*, N. Hiller, V. Judin, A.-S. Müller, M.J. Nasse, R. Ruprecht, M. Schuh
Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract

The accelerator test facility FLUTE (Ferninfrarot, Linac- Und Test-Experiment) is being under construction nearby ANKA at the Karlsruhe Institute of Technology (KIT). FLUTE is a linac-based accelerator facility for generating coherent THz radiation. One of the goals of the FLUTE project is the development and fundamental examination of new concepts and technologies for the generation of intensive and ultra-broad-band THz pulses fed by femtosecond electron-bunches. In order to study the various mechanisms influencing the final THz pulses, data-acquisition and storage systems are required that allow for the correlation of beam parameters on a per-pulse basis. In parallel to the construction of the accelerator and the THz beam-line, a modern, EPICS-based control system is being developed. This control system combines well-established techniques (like S7 PLCs, Ethernet, and EPICS) with rather new components (like MicroTCA, Control System Studio, and NoSQL databases) in order to provide a robust, stable system, that meets the performance requirements. We present the design concept behind the FLUTE control system and report on the status of the commissioning process.

INTRODUCTION

FLUTE (Ferninfrarot, Linac- Und Test-Experiment) is a new accelerator facility [1,2] being under construction at the Karlsruhe Institute of Technology (KIT). FLUTE is designed as an accelerator test facility, aiming at studying and improving techniques for producing very short electron bunches and studying the mechanisms involved in the generation of THz radiation from these electron bunches.

In the first stage, the accelerator will consist of a 7 MeV photo injector followed by a 40 MeV linac and a magnetic chicane used as a bunch compressor, as depicted in Fig. 1. Finally, synchrotron radiation generated in the final magnet of the bunch compressor or by an optionally inserted foil after the magnet is coupled out into a THz beamline for use by experiments. This THz radiation can then be analyzed in order to investigate properties of the electron bunch. However, it can also be used for independent experiments.

In order to make systematic studies of the parameters and mechanisms affecting the bunch compression and the generation of THz radiation, a pulse-synchronous data-acquisition system is needed. Such a system has to record key parameters (e.g. accelerator settings, RF pulses, laser pulses, beam charge and profile) for every single electron bunch, so that they can be correlated. At the same time, the control system has to provide live information, that is needed in order to optimize the accelerator operation.

* sebastian<dot>marsching<at>partner.kit.edu

CONTROL SYSTEM DESIGN

The design of the FLUTE control system was driven by the aforementioned demands as well as by the experience with the control system used for the ANKA accelerators.

We chose EPICS [3] as the core control-system framework for several reasons: First, EPICS is already used at many accelerators and thus there is a large number of already existing device drivers, reducing the development costs. Second, the underlying concepts are simple and thus easy to understand. This allows for a short training of new team members, so that they can quickly start working on control-system related tasks. Finally, EPICS is already in use at ANKA, so that we can benefit from our experience.

We use standard off-the-shelf components wherever feasible. This means that most computers are x86_64 systems running a Linux operating system. All systems are connected to a private IP/Ethernet network. The individual network connections use Gigabit Ethernet, however the backbone is already designed for 10 Gigabit Ethernet, allowing for higher data rates in the future.

We use the MicroTCA [4] platform for fast data-acquisition and feedback systems (e.g. the low-level RF (LLRF) system and beam-position monitor readout). This allows us to use the x86_64 platform while having the input/output (I/O) capabilities required for those applications.

For slow control tasks, we use devices with embedded IP/Ethernet controllers or serial interfaces where possible. Other devices are integrated using SIEMENS S7 programmable-logic controllers (PLCs) [5]. We use GigE Vision [6] cameras, allowing for a direct connection to the control system network.

We use Control System Studio (CSS) [7] as the main operator's interface to the control system. CSS is already in use at ANKA and provides an integrated user interface with tools for designing operator's panels, plotting archived data and displaying the alarm status.

DATA ACQUISITION

While the accelerator will initially operate with a repetition rate not exceeding 10 Hz, the data acquisition system is designed for repetition rates of up to 50 Hz to allow for upgrades in the future. This does not mean, that all diagnostics components can operate at this frequency, however the data-acquisition framework is supposed to handle this rate.

In EPICS, each process-variable (PV) sample has a time stamp with nanosecond precision. However, it is hard to synchronize the clocks of different components so accurately, that the time stamp can be used to correlate samples. There are systems like White Rabbit [8] that can provide a sufficiently synchronized clock across different computer

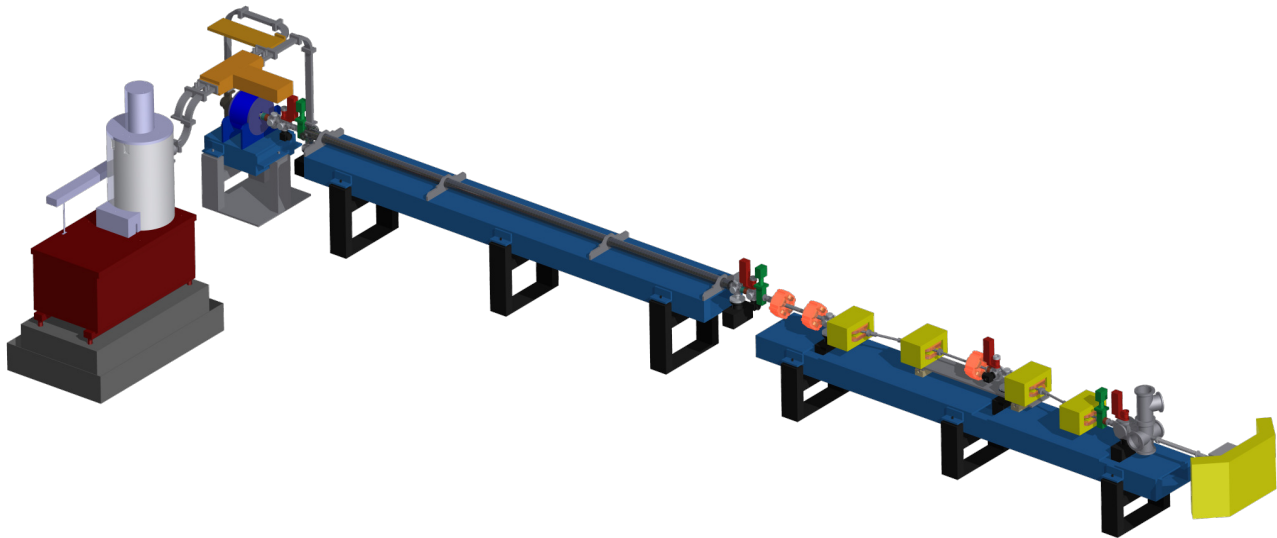


Figure 1: Layout of the FLUTE accelerator: Electron bunches are generated in a photo-RF injector (left), accelerated in an S-band Linac structure, and longitudinally compressed in a magnetic chicane, before being used to generate THz radiation. Finally, the electron bunches are destroyed in a beam dump (right). The injector and the linac are both powered by the same klystron (lower left).

systems, however these systems typically rely on special hardware. This would undermine the objective of using standard off-the-shelf hardware so that a different scheme is used.

Instead of trying to use consistent time stamps across the whole control system, the consistency of time stamps is only guaranteed within a single component. In addition to the PVs representing the actual measured quantities, each component has an artificial PV that represents the pulse counter. The time stamp of the pulse counter PV is synchronized with the time stamps of the measured PVs. Each time new measured data arrives, the pulse counter is incremented. This way, each PV sample can be correlated with a pulse sequence number. The measurement equipment can be synchronized using an external trigger signal that is independent of the control system. This solution has the advantage that it works with any measurement hardware (even hardware that is connected using an asynchronous interface like RS-232) as long as the measured data is transferred using a FIFO (so that no samples are lost) and the hardware supports some kind of external trigger.

ARCHIVING AND PROCESSING

We use an archiving solution [9, 10] based on Apache Cassandra [11] for archiving the PV values. The data is automatically distributed on a cluster of servers (see Fig. 2) allowing for both reliability and scalability. The performance scales linearly [12] with the number of cluster nodes, thus we can easily match future demands by just adding more computers.

The use of Apache Cassandra also allows us to use a MapReduce [13] based algorithm for aggregating the PV values belonging to the same pulse and running further analysis.

This way, we can benefit from the fact that the data is distributed in the cluster and run the analysis code on the same node that stores the corresponding data, thus improving the performance. We are currently evaluating Apache Spark [14] as the software framework for running this analysis code.

PROJECT STATUS

The development and commissioning of the control system is ongoing. Two virtual machine hosts, each running several virtual machines (e.g. for DHCP and DNS servers), have been prepared and are running in a test lab together with a PC designated as the operator's console. Other control-system components (e.g. network switches) have already been delivered and have been tested.

The software development is progressing. We have already created device drivers for some components (e.g. motor controllers). In close collaboration with DESY, we are working on integrating the LLRF electronics (developed and supplied by DESY) into our EPICS control system [15].

We expect the building infrastructure for FLUTE to be finished in early 2015, so that we can move the equipment from the test lab to its final location until spring 2015 and subsequently connect everything to the actual accelerator components. Thus, we expect to have first beam operation in 2015.

SUMMARY

For FLUTE, an EPICS-based control system has been designed and is currently under development. The control system uses standard off-the-shelf components where possible and leverages modern NoSQL-database technologies for data archiving and analysis. The control system is due to be

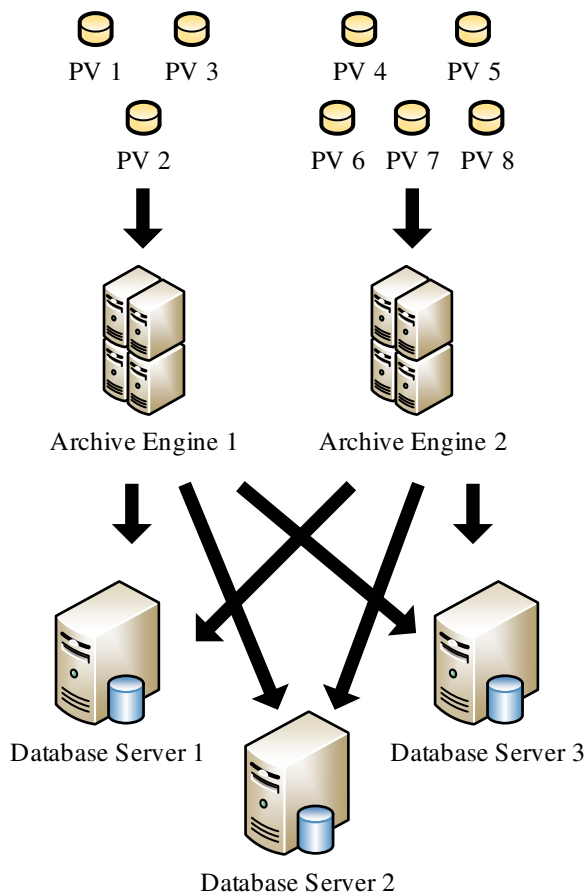


Figure 2: Different process variables can be handled by an indefinite number of archive engines. The archive engines automatically distribute the data on a cluster that can range from a few to a few hundred nodes and can be extended while running.

commissioned in early 2015 or and first beam operation is expected in 2015.

REFERENCES

- [1] M.J. Nasse et al., "FLUTE: A versatile linac-based THz source", Rev. Sci. Instrum. 84, 022705 (2013).
- [2] M. Schuh et al., "Status of FLUTE", IPAC'14, Dresden, June 2014; <http://www.jacow.org/>

- [3] "EPICS - Experimental Physics and Industrial Control System" website: <http://www.aps.anl.gov/epics/>
- [4] S. Jamieson, "Micro Telecommunications Computing Architecture Short Form Specification", PICMG, September 2006; http://www.picmg.org/pdf/MicroTCA_Short_Form_Sept_2006.pdf
- [5] Siemens, "Modular PLC controllers SIMATIC S7" website: <http://www.automation.siemens.com/mcms/programmable-logic-controller/en/simatic-s7-controller/>
- [6] AIA, "GigE Vision - Video Streaming and Device Control Over Ethernet Standard - version 2.0", Ann Arbor, November 2011; <http://www.visiononline.org/>
- [7] K. Kasemir, "Control System Studio Applications", ICALEPCS'07, Knoxville, October 2007; <http://www.jacow.org/>
- [8] J. Serrano et al., "The White Rabbit Project", IBIC'13, Oxford, September 2013; <http://www.jacow.org/>
- [9] S. Marsching, "Scalable Archiving with the Cassandra Archiver for CSS", ICALEPCS'13, San Francisco, October 2013; <http://www.jacow.org/>
- [10] aquenos GmbH, "Cassandra Archiver for CSS" website: <http://oss.aquenos.com/epics/cassandra-archiver/>
- [11] "Apache Cassandra" website: <http://cassandra.apache.org/>
- [12] J. Ellis, "2012 in review: Performance", DataStax Developer Blog, January 2013; <http://www.datastax.com/dev/blog/2012-in-review-performance>
- [13] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", OSDI'04, December 2004; http://usenix.org/publications/library/proceedings/osdi04/tech/full_papers/dean/dean.pdf
- [14] "Apache Spark - Lightning-Fast Cluster Computing" website: <http://spark.apache.org/>
- [15] M. Killenberg et al., "Drivers and Software for MicroTCA.4", these proceedings, PCaPAC'14, Karlsruhe, October 2014; <http://www.jacow.org/>

MAGNET POWER SUPPLY CONTROL MOCKUP FOR THE SPES PROJECT

M. Giacchini, M. Montis, M. Contran, INFN – Legnaro, Padova, Italy
M.A. Bellato, INFN- Padova, Padova, Italy

Abstract

The Legnaro National Laboratories employs about 100 Magnet Power Supplies (MPSs). The existing control infrastructure is a star architecture with a central coordinator and ethernet/serial multiplexers. In the context of the ongoing SPES project, a new magnet control system is being designed with EPICS[1][2] based software and low cost embedded hardware. A mockup has been setup as a test stand for validation. The paper reports a description of the prototype, together with first results.

INTRODUCTION

To improve the reproducibility and stability of the magnet control system used at LNL to the accelerators, the previous control system should be upgraded to a new one. The control system chosen for this task and to all site was EPICS. The previous system was based on terminal servers. The power supplies were connected to the terminal server which exports the serial interfaces to the central control workstation server. On this one a C program runs to manage the communications, and makes available the Graphical User Interface (GUI) to the operators.

This start architecture force the user to use only one GUI operator interface per time. The communication interface itself realize sub-groups concentrates them on terminal servers which relay on a serial interface converters; this makes, suddenly, communication problems in case of faults of only one power supply on the chain.

Basing on this experience and the strong EPICS architecture, the new system should have a great granularity, where, the communication shouldn't be concentrate, either the GUIs could be duplicated and moved without effort.

HARDWARE CONFIGURATION

The Input Output Controller (IOC) selected to this application is a low cost embedded PC (Figure 1). The power requirements necessary to this object is low enough to use the Power Over Ethernet technology. This makes extremely cheap the cabling, but requires the necessary power on the Ethernet switch. We are evaluating various kind of embedded PC which could play this role, the final decision has not been done, but the tests already done seems we have made a good technology selection.

The power supply of the magnets can use two kind of serial communications: RS232, RS422. The previous system has used the second one. The pros of that choice were the reduction of the amount of the lines, the reduction of the cabling which means a final reduction of the total cost; but the cons was quite bad, if someone of the serial controller of the power supplies on the line goes on communication fault state all the controllers on that lines are not controllable until the fault was fixed.

The new system makes all the communication independently using the RS232 connection, which realize a point to point connection between the controller and the embedded PC. Later, if this should be the case, we move back to the RS422, which is more strong in front of noises, but in any case we would use the point to point connection, to make the system robust as much as we can, minimizing communication delay.

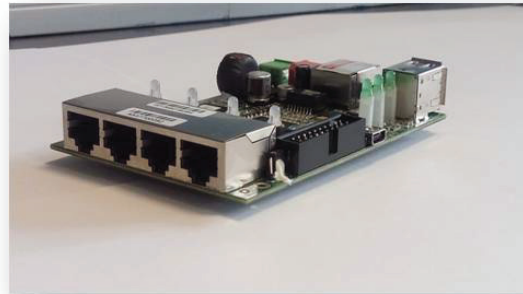


Figure 1: Example of embedded used as IOC for the magnet system.

SOFTWARE

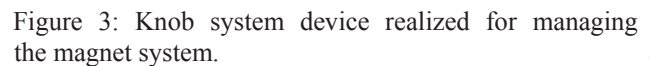
The core of the software, the device communication was accomplished by using ASYN[3]. A basic version used on other site has been modified and customized to fit the requirements of our site. More works has requested the development of the GUI operator interface, based on Control system Studio (CSS), as visible in Figure 2.

After 20 years of operation on the old magnet system the operator itself, for a first time in the history of our laboratories, has been trained to design and realize the skeleton of that GUI. Themselves feels comfortable using a GUI designed by them, that is going to make the change of the control system more smooth.



Extremely useful during these test is the EPICS archiving system[4] which makes surveillance on the various IOCs and the relative variables.

As soon as possible, matching with the agenda of shifts and maintenance stages, a consistent large installation will be placed like a proof of concept of the software and hardware selection choice.



The collaboration with the operators team is the first tentative to this kind but has already made great result on the handling of the control system and the related GUIs.

ACKNOWLEDGMENTS

We are grateful for their help and collaboration to G. Martin (PSI) and to A. C. Mezger (PSI) for sharing their knowledge related to knob devices developed in EPICS environment. This work leveraged years of experience on EPICS use from good engineers of other laboratories around the world: great acknowledgments to them.

REFERENCES

- [1] EPICS website: <http://www.aps.anl.gov/epics>
- [2] EPICS at LNL website: <http://www.lnl.infn.it/~epics>
- [3] ASYN website:
<http://www.aps.anl.gov/epics/modules/soft/asyn/>.
- [4] M. Montis et al., “New data archive system for SPES project based on EPICS RDB Archiver with PostgreSQL backend”, FPO014, PCAPAC’14, Karlsruhe, Germany (2014)

IFMIF EVEDA RFQ LOCAL CONTROL SYSTEM TO POWER TESTS

M. Giacchini, M. Montis, L. Antoniazzi, INFN-LNL, Legnaro, Italy

Abstract

In the IFMIF EVEDA project, normal conducting Radio Frequency Quadrupole (RFQ) is used to bunch and accelerate a 130 mA steady beam to 5 MeV. RFQ cavity is divided into three structures, named super-modules. Each super-module is divided into 6 modules for a total of 18 modules for the overall structure. The final three modules have to be tested at high power to test and validate the most critical RF components of RFQ cavity and, on the other hand, to test performances of the main ancillaries that will be used for IFMIF EVEDA project (vacuum manifold system, tuning system and control system). The choice of the last three modules is due to the fact that they will operate in the most demanding conditions in terms of power density (100 kW/m) and surface electric field ($1.8 \cdot E_{kp}$). The Experimental Physics and Industrial Control System (EPICS) environment [1] provides the framework for monitoring any equipment connected to it. This paper reports the usage of this framework to the RFQ power tests at Legnaro National Laboratories [2,3].

INTRODUCTION

The RFQ Local Control System (LCS) Architecture approved by the IFMIF-EVEDA Collaboration is designed to optimize reliability, robustness, availability, safety and performance minimizing all the costs related to it (purchase and maintenance). Following this philosophy and the IFMIF-EVEDA Guidelines, we realized a control system network composed by two different kinds of hosts:

- Physical machines for critical control system tasks;
- Virtual hosts in machines where no particular functional task or hardware is required.

The architecture realizes the 3-layer structure described in the Guidelines and each layer defines a proper hosts group (equipment directly connected to the apparatus, control devices, Human-Machine Interface) while the EPICS framework provides the interface between them.

LCS CORE SYSTEM

The core workstation server provides capabilities that enable controls engineers to deploy customized environments for their application perfectly aligned with the “Common Software Guidelines”. All the regular EPICS services, are backed up regularly and can be moved and cloned easily. Key enabling technology for this is the virtualization and the provisioning; this approach allows the installation saves floor space, power and cooling.

In the IFMIF-EVEDA RFQ LCS, Logical Volumes are used to define main server's partition table and the virtual hard disks used to realize virtual hosts. In this way, it is possible to manage any resourceful saturation in according to the free resources provided by the main

server. Because of the role covered by the server, all the unnecessary services and ports are switched off following an hardening policies to keep safe as much as possible the control system.

Archiver

The Channel Archiver is an archiving tool-set for EPICS based control systems. It can archive any kind of record available through the EPICS Channel Access. The deployable Archiver prepared into the manager server is ready to use. The retrieving interface is based on Control System Studio (CSS) framework. After the commissioning, this archiver will be switched off and replaced from the central one available from the Central Control System (CCS).



Figure 1: RFQ LCS Racks for Power Tests.

Deploy and Backup

In any control network, cases of hardware failure or breakdown can be very dangerous, especially in the case of infrastructure similar to the IFMIF facility. The need to restore the controls functionality in the shortest time is therefore fundamental. The manager machine is designed to realize an automated management for new machinery's configuration inside the RFQ LCS. In particular, it is possible to connect a new device to the network, indicate which type of play rule it must realize and the network itself will auto-configure it. The manager host uses dedicated open source software for provisioning the entire control system network, while backup service saves

ISBN 978-3-95450-146-5

through proper customized scripts the principal files of interest associated with the main machineries (physicals and virtual) available in the LCS.

Surveillance

In a critical system like the RFQ LCS, administrators should be constantly updated on this state and must be able to intervene promptly if required. In the local control network an open source solution was chosen to perform this task: it provides computer system monitor and network monitoring software application. LNL developed a plugin that allows state supervision of EPICS Process Variables in a control network[4]; this involves the possibility to monitor both the EPICS control system network and the computer network. The entire network can be controlled through a proper web interface and, in case of warning or critical problem, email notifications are sent to the system network administrators.

Project Management Documentation Server

Software projects become larger and more complicated; set up of agile project management methods to realize an effective software configuration management (SCM) environment improve both quality and productivity. With this target the RFQ LCS development it's managed and documented using these tools, which are widely used and recognized in Open Source development.

- Bugzilla is a "Defect Tracking system" based on MySQL and managed by a regular web browser;
- Wiki service is provided by MediaWiki, the largest used wiki which provides the engine of the famous Wikipedia. MediaWiki uses MySQL and Apache;
- Subversion server will contain all software development usually, but can be used to manage the revision of document like this one. A nice server named WebSVN has been setup to have the access of SVN by web pages.

EPICS softIOCs Virtual Server

In subsystem where the acquisition rate is the most critical requirement or where dedicated devices must be integrated, the control system is developed directly in EPICS. To realize and manage the IOCs particular hardware is not required, a properly configured virtual machine equipped with the entire EPICS environment is developed; in this way system administrator have a centralized server to supervise. This machine is created for having two tasks:

- Realize a EPICS IOC for every device and system which require it
- Provide a Boot Server for the VME system in charge of realize the RF acquisition

LOCAL CONTROL SYSTEM

The RFQ system is complex apparatus composed by many kinds of subsystems (radio frequency, vacuum, water cooling, etc.) developed using different hardware solutions. As consequence, every part of this structure

must be properly integrated to obtain the desired degree of control. Following these criteria, the system has been designed and realized using these assumptions:

- PLC hardware is chosen in tasks where security is the most critical feature;
- VME system is used where the acquisition speed rate is crucial;
- Common hardware (such as embedded systems) is chosen when only integration is required.

In Figure 2, it is possible to observe that for every subsystem a particular solution based in one of the three assumptions mentioned above is adopted. The LCS could be resumed by the following functionalities:

- RF signal generator;
- Fast acquisition system for the RFQ cavity power;
- Surface Temperature Monitor (STM) system;
- Bunch length monitor system;
- Vacuum system;
- Cooling system;
- Machine Protection System (MPS)
- Personal Protection System (PPS).

To drive the 220kW RF Amplifier, used during the power test, a RF signal generator is necessary. For this reason an EPICS driver was written to integrate that.

The fast acquisition is based on VxWorks real time OS which run over a VME architecture. The most important signals about RF power are sampled with a maximum rate of 1MEvents/s. To perform the automatic calibration of the acquisition channels a useful tool was developed.

The surface working temperature of a RFQ has never been analyzed before on other RFQs; we will map the temperature distribution along the RFQ with a high defined mesh, saving economic and wiring efforts. To this purpose 1-Wire® devices [2] provide simple and cheap technology; a dedicated EPICS driver was developed to integrate them. We will use 96 sensors for the high power test and 432 for the entire RFQ.

IFMIF-EVEDA requires a system to measure the micro bunch length. A slim embedded solution with the needed ADCs/DACs is enough to this measure.

Vacuum, Cooling, MPS and PPS functionalities are realized by PLCs while the EPICS framework is in charge of graphic user interface (GUI), archiving and alarm managements.

The vacuum control system is already done; PLC software and GUI, developed in CSS, was fully tested on the high power test vacuum system, which is composed by 1 of the 10 RFQ's pumping stations.

The cooling system will be used on RFQ cavity with the double purpose of power removing and frequency tuning by means of an extremely precise water temperature control. Five different working modes have been implemented on PLC's SW to satisfy all the required combination of these control functions.

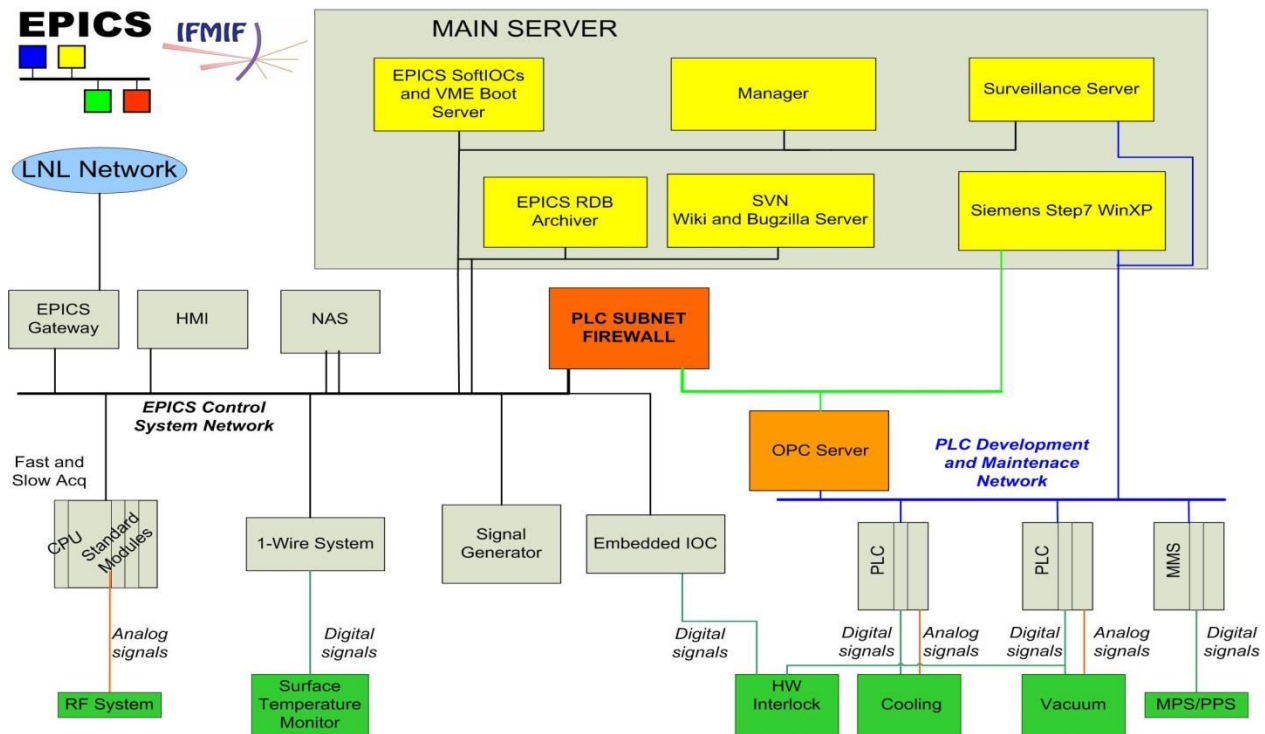


Figure 2: Control System Architecture for IFMIF EVEDA RFQ power tests.

PPS, for area access management, and the slowest part of MPS are in charge of a: SIEMENS® modular safety system (MSS) [5] which is SIL3 compliant. PPS and MPS main task is to stop the RF amplifier in case of any abnormal condition. PPS and MPS functionalities have been fully tested before the RF couplers high power test.

The Graphical User Interface (GUI), developed in CSS, is composed by a set of panels which let scientists to remote control the apparatus. Main panel give the global view of the experiment, while specific panels and pop-up windows, divided according the subsystems, give detailed information. A remote control using a mobile smart phone has been already tested. A dedicated machine host the EPICS Archiver which stores the data used by scientists. Under EPICS environment, there is almost one EPICS database for every subsystem, except for the mathematical side, which is composed by a set of database where each one performs a particular calculus.

The following table resumes the size of the whole EPICS application realized.

Table 1: Summary of EPICS Application Size

Objects	Numbers
Server EPICS	3
IOCs	8
Databases	24
Process Variables (PV)	2173
PV Archived	ca. 600
GUI Panels	27

CONCLUSIONS

Preliminary tests on the singular control system tasks are done with positive results. During RF Coupler High power test the main part of LCS, with exception of vacuum and cooling system, was utilized. Subsystems connected to RF signals generation and acquisition have been widely used and appreciated. Because of the importance of the power tests for the RFQ apparatus, this is a great test-bench for the entire control system architecture and a good feedback for the work realized.

REFERENCES

- [1] <http://www.aps.anl.gov/epics/>
- [2] <http://www.lnl.infn.it/>
- [3] <http://www.lnl.infn.it/~epics/joomla/>
- [4] M. Giacchini et al., "LivEPICS: an EPICS Linux Live CD Nagios Equipped", TPPA32, ICALEPCS2007, Oak Ridge, USA
- [5] <http://www.siemens.com>

UPGRADE OF BEAM DIAGNOSTICS SYSTEM OF ALPI-PIAVE ACCELERATOR'S COMPLEX AT LNL

M. Giacchini, G. Bassato, M. Poggi, M. Montis, INFN/LNL, Legnaro, Italy
B. Liu, CIAE, Beijing, China

Abstract

The beam diagnostics system of ALPI-PIAVE accelerators has been recently upgraded by migrating the control software to EPICS[1]. The system is based on 40 modules each one including a Faraday cup and a beam profiler made of a couple of wire grids. The device's insertion is controlled by stepper motors in ALPI and by pneumatic valves in PIAVE. To reduce the upgrade costs the existing VME hardware used for data acquisition has been left unchanged, while the motor controllers only have been replaced by new units developed in house. The control software has been rebuilt from scratch using EPICS tools. The operator interface is based on CSS; a Channel Archiver based on PostgreSQL[2] has been installed to support the analysis of transport setup during tests of new beams. The ALPI-PIAVE control system is also a bench test for the new beam diagnostics under development for the SPES facility, whose installation is foreseen in middle 2015.

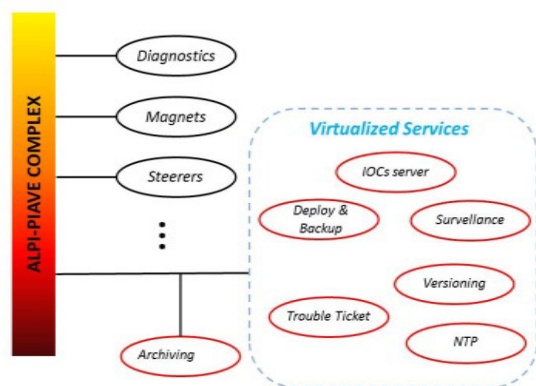


Figure 1: PIAVE-ALPI complex.

INTRODUCTION

Beam diagnostic system is a critical task for managing the accelerator complex, because it is the first stage where scientists and operators understand machine status and behaviour and execute the operations required to provide the desired beam.

Diagnostics system will be a part of the control system architecture under implementation at INFN/LNL Laboratory (Figure 1). One of the greatest improvement coming from the assumption of EPICS as main control system framework is the creation of a distributed environment where every sub-system is integrated and interconnected with others. As consequence, operators

and scientists can have an easier and more direct control to the entire apparatus. The upgrade related to the diagnostics system is realized focusing in two aspects: the reuse, as much as possible, of the hardware already installed in ALPI (to contain costs) and the integration into the EPICS environment. First development stage was realized in 2011[3].

DIAGNOSTICS SYSTEM ARCHITECTURE

In principle the ALPI-PIAVE diagnostics environment is composed by 48 diagnostics boxes installed along the apparatus containing, each one, a faraday cup and a couple of grids (horizontal and vertical) where every single grid is made of 40 wires. To avoid white noise induced by cables, current signals are converted to voltage next to the diagnostics box. After the conversion, grid signals are multiplexed and serialized before the transport to the acquisition system composed by an ADC card installed in VME crates. The multiplexer is driven by a counter whose clock is generated by the ADC itself, to have the signal transmission synchronized with the conversion. At the beginning, the software layer used for this sub-system was based on C programs developed on VxWorks OS for the VME systems and custom java application developed inside the INFN for the Human-Machine Interface (HMI).

With the focus of renovate and upgrade the control system software, EPICS was chosen as control system framework for the new environment, bringing all the advantages coming from this solution (performances, data distribution, etc.). As consequence, the actual set of programs and tools available to control the ALPI-PIAVE complex has been substituted piece by piece with the new EPICS application.

According to EPICS documentation, VxWorks is supported by EPICS since its origin; therefore VME systems are not completely changed: the VME processor implements an Input/Output Controller (IOC) providing the acquisition of grid signals and faraday cups. Databases loaded in the IOC are not very complex, due to simplify the maintenance, and provide minimal processing on raw data. Signal from beam profilers and faraday cups are acquired by XVME566 board produced by XYCOM: this kind of hardware provides 12 bit resolution and support a conversion rate of 100 KHz in streaming mode. As indicated in Figure 2, a dedicated device driver realized by M. Davidsaver realizes the software interface between the field and the EPICS server host.

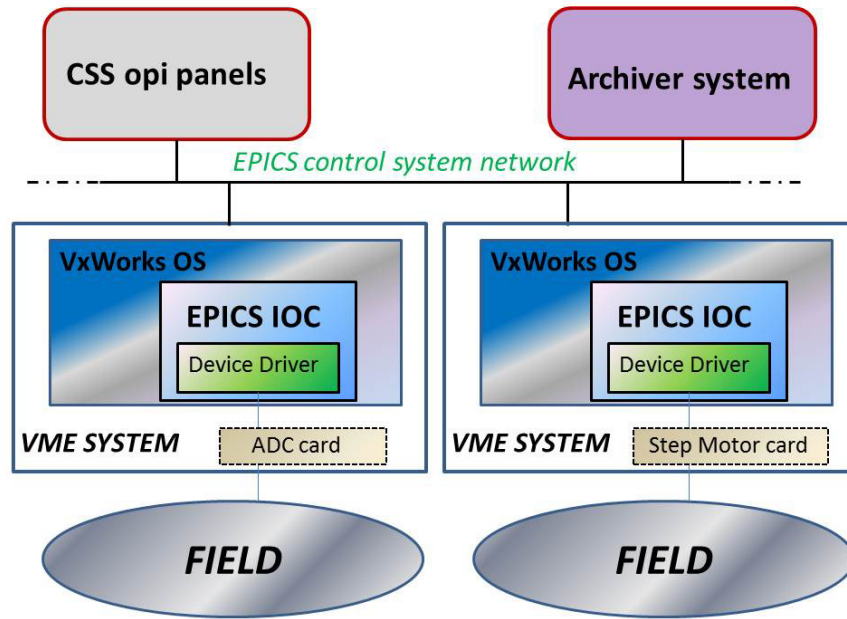


Figure 2: Software schema adopted for VME systems.

The steppers motor controllers for collimator position are VME cards built inside the LNL, together with the associated power drivers: every single controller is able to manage up to 8 motors. The crate controller is an Emerson MVME3100 based on the PowerPC8540 equipped with 256MB RAM. Also this host runs under VxWorks OS which provides an EPICS IOC properly configured to manage the devices connected and provide a set of basic functions:

- acquisition of collimator position from linear encoders
- control of step motors used to synchronize the collimator motion with the grid signal acquisition

Due to realize this functionality, a dedicated Device Driver has been developed by D. Chabot to control motor position and provide feedbacks.

The entire diagnostics system results composed by 1600 record, counting both records managing raw and elaborated data. For more details see Table 1.

Table 1: Records Implemented for Diagnostics System

Sub-sytem	Number of PVs
Grids and FC	1300
Step motors	300

HUMAN MACHINE INTERFACE AND ARCHIVE SERVICES

Different tools and applications are available in the EPICS environment to create dedicated control panels for managing the facility and the Linac apparatus, and every one provides particular features. Following the line defined for the SPES Project, Control System Studio (CSS) was chosen as common layer to realize human machine interfaces, in order to have an integrated environment where the user (scientist or operator) can use different tools (such as control panel, archiver viewer and alarm handler) in the same application.

As consequence, a dedicated set of control panels are developed in CSS inheriting as starting point the layout used until now by final users. The whole HMI for the ALPI diagnostics system is composed by a main panel representing the map of the linear accelerator (Figure 3) and a template where user can read information from a particular diagnostic box:

- in the main panel only the general information about diagnostic boxes are shown, such as grid and faraday cups' status and commands, due to simplify the reading
- selecting the interested diagnostic boxes from a dedicated table, it is possible to open the template window where all the information related to the desired devices is shown.

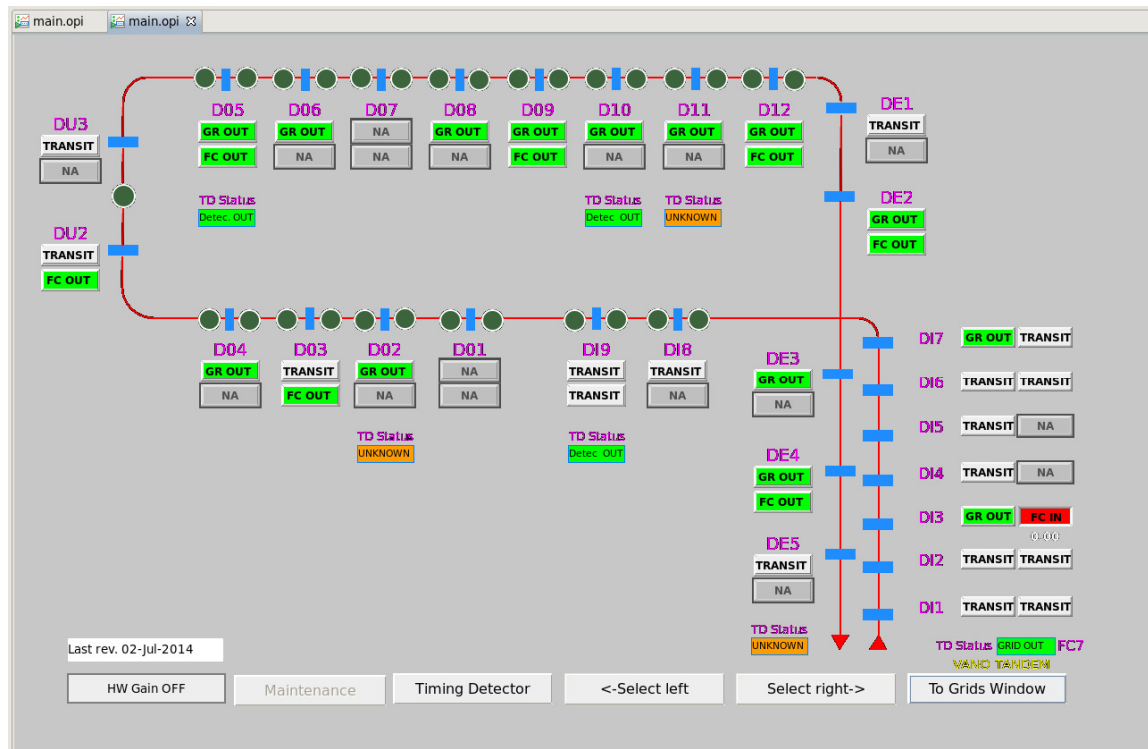


Figure 3: Main panel for ALPI Diagnostics system based on EPICS CSS framework.

The diagnostics system, as the first part of the apparatus involved in the control system upgrade was used as test bench for EPICS archiver service. Preliminary basic operation related to data storing and retrieving are performed between this sub-system and the archiver machine in order to verify the correct communication in both sides and define the common standard configuration which will be used for all the other sub-systems. The data acquisition performed now is used to estimate the hardware requirements (disk space, etc.) needed by the archiver service in production. In these tests both raw and elaborated data are stored, due to understand the best sampling configuration for archiver acquisition with different type of data.

For the ALPI diagnostics system, archiving all the information provided by this system requires about 70 GB/day; obviously this is the worst case and optimization is required, but this information can be used as upper bound for design data archiving for the diagnostics.

CONCLUSION

The diagnostics system for ALPI-PIAVE complex is migrated to the new EPICS control system framework in according to the guidelines defined in the SPES Project. The software migration to EPICS resulted in a significant increase of flexibility and performance of overall system.

All the diagnostics boxes provide EPICS IOC and the interconnection to the field through dedicate Driver Support developed in collaboration with BNL, while a dedicated HMI is developed in CSS. The functionality is

full tested and the system is ready for production. Minimal minor configurations will be required in order to

optimize services related to the control system, such as Archiver system.

ACKNOWLEDGMENTS

We are grateful for their valuable help to M. Davidsaver (BNL) and to D. Chabot (BNL) for the great help in developing all the interfaces required to integrate the hardware used to the EPICS environment.

This works leveraged of years of experience on EPICS use from good engineers of other laboratories around the world: great acknowledgments to them.

REFERENCES

- [1] EPICS website: <http://www.aps.anl.gov/epics/>
- [2] M. Montis et al., "New Data Archive System for SPES Project Based on EPICS RDB Archiver with PostgreSQL Backend", PCaPAC'14, Karlsruhe, Germany (2014)
- [3] G. Bassato et al., "A Beam Profiler and Emittance Meter for the SPES Project at INFN-LNL", MOPMS035, Proceedings of ICALEPCS2011, Grenoble, France (2011)

STARS: CURRENT DEVELOPMENT STATUS

T. Kosuge, Y. Nagatani, KEK, Tsukuba, Japan

Abstract

Simple Transmission and Retrieval System (STARS) is a simple and useful software for small-scale control systems running on various operating systems.

STARS is used by various systems at the KEK Photon Factory (e.g., the beamline control, experimental hall access control, and key handling systems), and the development of STARS (e.g., development for many kinds of STARS clients and for interconnection of Web2c and STARS) is ongoing. We describe the current development status of STARS.

STARS OVERVIEW

Simple Transmission and Retrieval System (STARS) [1] consists of client programs (STARS clients) and a server (STARS server) program. Figure 1 shows some STARS configuration examples. Each STARS client is connected to the STARS server via a TCP/IP socket and communicates using text-based messages. Every STARS client has its own node name, which is unique in the system; a message with a destination node name sent to the STARS server by a client is delivered to the corresponding STARS client by the STARS server. STARS uses this simple message transfer mechanism to provide control system functionality.

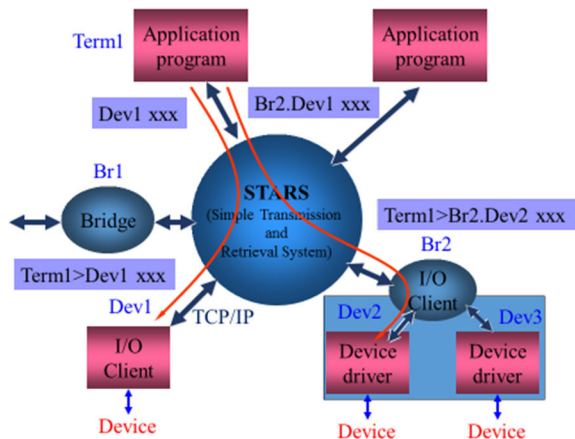


Figure 1: Layout example of STARS server and clients.

Multi-platform

The STARS server-side application is written in Perl and can run on various operating systems (e.g., Windows, Macintosh, and Linux). Development of a control system using STARS consists primarily in writing STARS client programs. The developer can choose his or her favorite operating system or programming language, if they support TCP/IP Socket and the handling of text string functions.

Simple Certification

The STARS server contains a simple certification procedure performed at STARS client connection time. STARS avoids client misconnection through three function steps, as follows:

- Host name certification
- Node name and keyword certification
- Node name and host name certification (optional)

Interface Libraries

STARS interface libraries (for Perl, .NET, Java, C, and ActiveX) assist in STARS client development. Developers need not be skilled in TCP/IP Socket or in the Node name and keyword certification procedure to use a STARS interface library.

INSTALLATION STATUS

STARS is used as a common beamline control system at the Photon Factory for introducing various systems. Table 1 lists the recent installation status of STARS at KEK.

Table 1: STARS Installation Status (as of September 2014).

Category	Beamline or Control System
PF-2.5GeV Ring X-ray	BL-1A, BL-3A, BL-4B2, BL-5A, BL-6A, BL-6C, BL-7C, BL-8A, BL-8B, BL-9A, BL-9C, BL-10A, BL-10C, BL-12C, BL-14A, BL-15A, BL-17A, BL-18B
PF-2.5GeV Ring VUV and Soft X-ray	BL-2A, BL-2B, BL-11A, BL-11B, BL-13A, BL-16A, BL-19B, BL-20A
PF-AR	NW-1A, NE-3A, NW-2A, NW-10A, NW-12A, NW-14A
Slow Positron Facility	SPF-A, SPF-B
Other Systems	Beamline Interlock Central Control System, Experimental Hall Access Control System, Key Handling System, Front-End Monitoring System, Radiation Monitoring System

CLIENT DEVELOPMENT

STARS contains three types of client programs (User, I/O, and Others). Users can add control system functions by adding STARS client programs without system stoppage.

A user client is a user interface program that includes a GUI. We have developed several types of common GUI, such as stepping motor, monochrometer, and mirror control panels.

An I/O client behaves in the manner of a device driver in STARS. We support commonly used hardware at the Photon Factory beamlines. If a user requires the use of new hardware, we discuss the general character of the hardware and begin development. The typically supported hardware is listed in Table 2.

Other types of client (e.g., a sysloger or breaker) work as background processes providing many functionality.

Table 2: Typically Supported Hardware at the Photon Factory.

Type	Product
Stepping Motor Controller	Tsujii PM16C series, Tsujii PM4C series, Tsujii NPM2C-01, Tsujii UPM2C-01, Kozu SC-200
Digital Multi Meter	KEITHLEY 2701 series
Pico Ammeter	KEITHLEY 6485, KEITHLEY 6487
Encoder	HEIDENHAIN ND261, Tsujii ER2C series, Tsujii ER4C series
Counter	ORTEC 974, Tsujii CT08 series, Tsujii NCT08 series

RECENT CHALLENGES

Web2c

Web2c [2] is a very powerful tool kit developed by DESY that provides web-based GUI functions. Currently, a test version of a Web2c/STARS interface has been



Figure 3: Snap shot of Web2c/STARS GUI panel on Android.

developed, and we have succeeded in connecting Web2c and STARS. Figure 2 shows a Web2c GUI panel with STARS on Windows 8.1, and Fig. 3 shows the GUI on Android. We obtained satisfactory results in this test. Development of a Web2c/STARS interface is ongoing.

STARS over RS232

Bridge clients are used for connecting a STARS server to other STARS servers (STARS bridge) or other protocols, such as TINE [3] (TINE bridge [4]). We have developed a new type of STARS bridge, the RS232 bridge, that works with RS232, connecting two STARS servers via RS232 with serial communication. STARS servers can communicate without a network using the RS232 bridge (Fig. 4). This performs efficiently in environments that do not have a network.

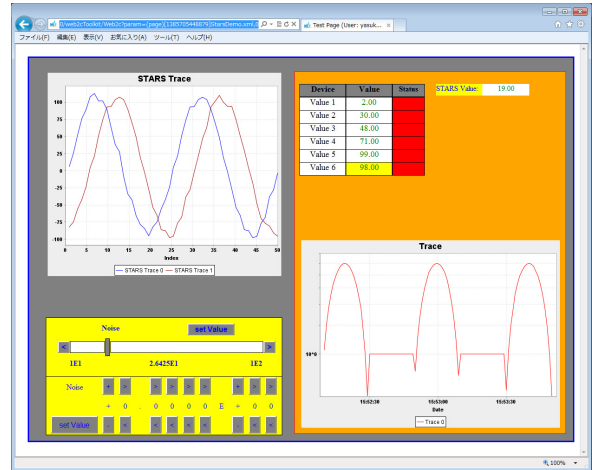


Figure 2: Snap shot of Web2c/STARS GUI panel on Windows 8.1.

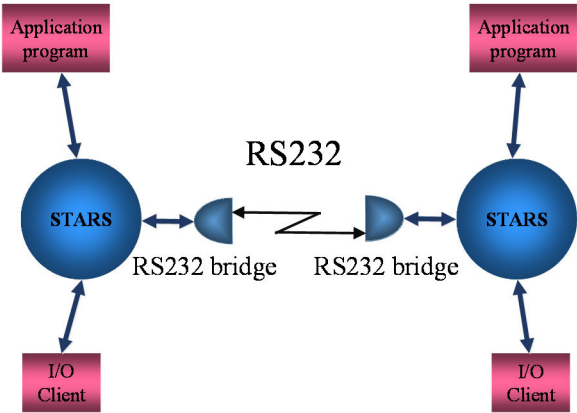


Figure 4: STARS RS232 bridge.

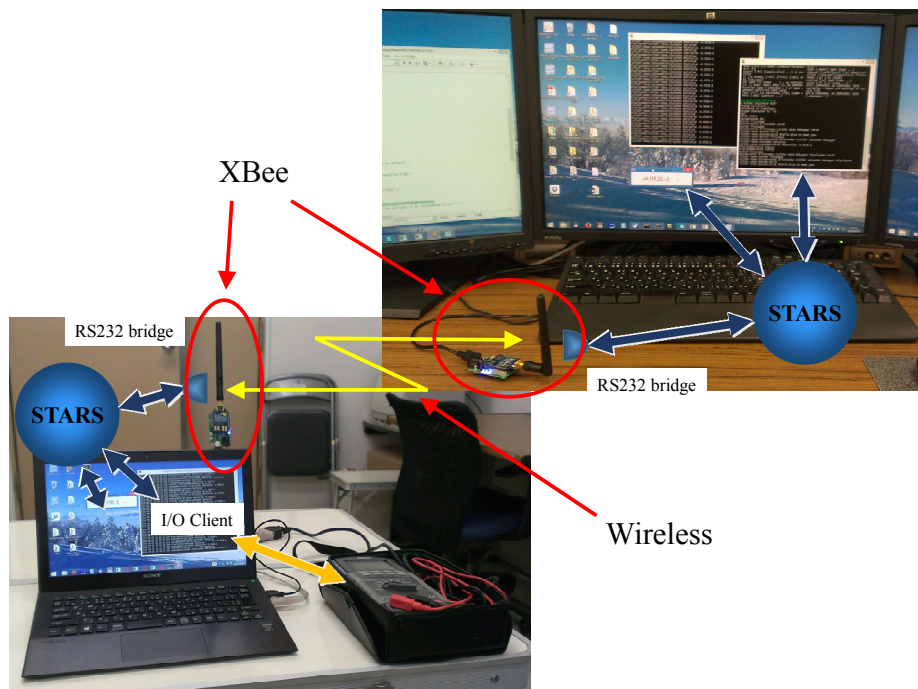


Figure 5: STARS RS232 bridge with XBee.

The RS232 bridge works well also with wireless modems (e.g., XBee), assisting development of control or data acquisition systems with slow wireless communication. Figure 5 shows a test bench of the RS232 bridge with XBee. We have succeeded in reading values of a digital multi meter from clients connected to another STARS server with wireless serial communication.

CONCLUSION

STARS is a simple and useful system installed in many kinds of system including beamline control systems. Development and updating of STARS (development clients and STARS functions) continues.

We have begun development of a STARS-Web2c practical application for our control system, and we are

planning to develop a standalone wireless monitoring system with the RS232 bridge.

REFERENCES

- [1] <http://stars.kek.jp/>
- [2] http://adweb.desy.de/mcs/web2cToolkit/web2c_home.htm
- [3] <http://adweb.desy.de/mcs/tine/>
- [4] T. Kosuge, et al., The Interconnection of TINE and STARS, PCaPAC2006, Newport News, 2006.

DEVELOPMENT AND APPLICATION OF THE STARS-BASED BEAMLINE CONTROL SYSTEM AND SOFTWARE AT THE KEK PHOTON FACTORY

Y. Nagatani, T. Kosuge, KEK, Ibaraki, Japan

Abstract

The Simple Transmission and Retrieval System (STARS) [1] is a message transferring software for small-scale control systems, originally developed at the Photon Factory (PF). It has a server-client architecture using TCP/IP sockets and can work on various types of operating systems. Since the PF adopted the STARS as a common beamline control software, we have developed a beamline control system that controls optical devices such as mirrors and monochrometers. We also developed various systems and software, such as the information delivery system of the PF ring status based on the STARS and Three-fold Integrated Networking Environment (TINE) [2], and measurement software based on the STARS, for the PF beamlines. Currently, many useful STARS applications are available, such as device clients, simple data acquisition, and user interfaces. We will describe the development and installation status of the STARS-based beamline system and software.

OVERVIEW OF STARS

The Simple Transmission and Retrieval System (STARS) consists of a server program, the *STARS server*, and client programs, the *STARS clients*. Each client is connected to the STARS server through a TCP/IP socket, and it communicates via text-based messages. Each client program has its own unique node name, and it sends text-based messages using the destination node name to the server, which then delivers the messages to the destination client. A STARS server, which is written in Perl, can run on various operating systems. STARS users can upgrade the system by writing client programs, and STARS clients can participate in the system at any time without system stoppage.

STARS MESSAGE STANDARDIZATION

Originally, the STARS had the following simple rules for messages:

- A message that starts with “@” (e.g., @message) is a reply to a command.
- A message that starts with “_” (e.g., _message) is an event.
- Any other type of message is a command.

For adopting the STARS as a common beamline control software, moreover, we standardize the STARS messages that control the device components including such as mirrors and monochrometers. Table 1 shows commonly used standardized STARS messages.

Table 1: Commonly used Standardized STARS Messages

“GetValue”	STARS command to get the target value.
“SetValue”	STARS command to change the target value.
“IsBusy”	STARS command to get the moving status of the target.
“_ChangedIsBusy”	STARS event to notify the change in moving status of the target.
“_ChangedValue”	STARS event to notify the change in the target value.

The standardization of STARS messages enables reusable beamline control programs and software to be developed. In addition, this standardization allows anyone using the same message format to control these baseline programs and software.

Further, the standardization of STARS messages reduces developmental costs associated with the creation of control programs; moreover, it enables beamline users to develop their own control programs.

BEAMLINE CONTROL USING STARS

Installation Status

Since the Photon Factory (PF) adopted the STARS, a common beamline control software, the development and installation of STARS-based beamline control systems have been increasing every year. Recently, more than 30 beamlines have introduced STARS-based beamline control systems (see Table 2).

Table 2: Installation Status of STARS-based Beamline Control Systems (October 2014)

Category	Installed beamline
2.5 GeV Ring X-ray	BL-1A, BL-3A, BL-4B2, BL-4C, BL-5A, BL-6A, BL-6C, BL-7C, BL-8A, BL-8B, BL-9A, BL-9C, BL-10C, BL-12C, BL-14A, BL-15, BL-17A, BL-18B, BL-18C
PF-2.5 GeV Ring VUV and Soft X-ray	BL-2, BL-11A, BL-11B, BL-11D, BL-13A/B, BL-16A, BL-20A, BL- 28
PF-AR	NE1A, NE3A, NE7A, NW2A,

	NW10A, NW12A, NW14A
Other	Slow Positron Facility (SPF-A, SPF-B)

In addition installation of STARS-based beamline control system for BL-3B and BL-3C are in progress.

System Configuration

Figure 1 is an example of an actual STARS-based beamline control system. The figure shows an operational STARS server, with several STARS clients connected to this server on the same computer. A STARS-based system can be distributed on multiple computers that are connected by a network; the system layout should be selected on a case-by-case basis, depending on a number of factors such as scale, requirements, and budget.

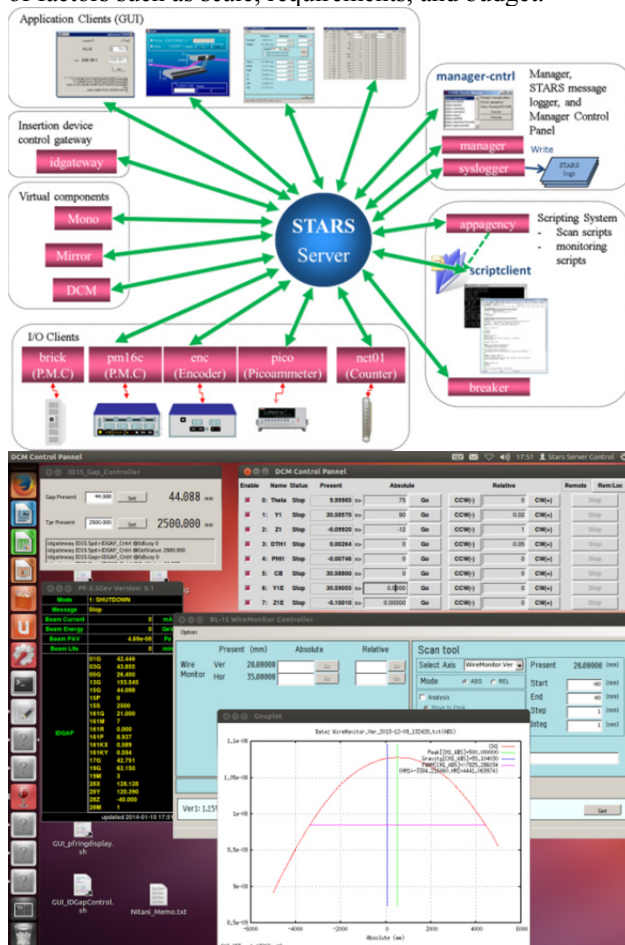


Figure 1: A beamline control system with a STARS and screenshots of the control GUIs.

Insertion Device Control Software

For each undulator beamline, the undulator gateway server is prepared by the PF Ring and PF-Advanced Ring (AR) control groups. By connecting the gateway server via a TCP socket, we can remotely control the modification of undulator parameters such as the gap and mode. In order to facilitate modification of the undulator parameters at each beamline control system, we provide

STARS-based insertion device control programs. (See Fig. 2)

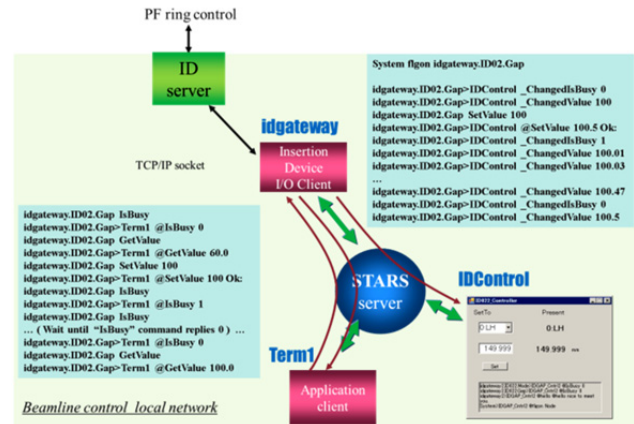


Figure 2: Insertion device control software. Any STARS clients connected to the STARS server, including the user programs, can obtain the values of the ID's properties.

PF Ring Information Delivery System

The PF Ring information delivery system was first developed in 2006 using TINE and STARS. [3] PF ring properties such as beam current, lifetime, operation mode, and gap of undulator can be referred on a STARS-based beamline control system. The system consists of a TINE server, which obtains the PF ring properties; a TINE and STARS bridge client, which receives the PF ring properties from the TINE server and provides these properties to the STARS; and a display client of the STARS.

The TINE server is on the KEK internal network, and the PC that the TINE and STARS bridge client runs has two NICs: one is connected to the KEK internal network and another is connected to the beamline control local network. The display client can run on any PC on the beamline control local network. The PF ring information can be delivered from the TINE server to the TINE client by using the multi-cast function of TINE. The PF has installed the system on several beamlines. (See Fig. 3)

Experiment and Measurement Software

As the use of the STARS-based beamline control system at KEK PF has increased, the software requirements for conducting experiments have also increased. For example, changes in the beamline component's properties, such as the energy and undulator gap values, then measure the experimental data.

As an example of experiment and measurement software, we developed the detector “PILATUS” (DECTRIS) control software and image monitoring software (See Fig. 4) for the KEK PF SAXS beamlines in 2012; it was successfully implemented. The programs are written in Perl and C# and the software requires a Perl and .NET 3.5 environment. The software can run on a single computer, and is portable. It has been installed on the other beamlines equipped with the “PILATUS” detector.

RENOVATION OF PC-BASED CONSOLE SYSTEM FOR J-PARC MAIN RING

S. Yamada*, KEK, Sokendai, J-PARC, Ibaraki, Japan

Abstract

Console system for J-PARC Main Ring (MR) was designed in 2007 and had been used for accelerator commissioning and operation since then. It was composed of 20 diskless thin clients and 10 blade servers. Both of them are PC-based computers running Scientific Linux (SL) 4 as their operating system. In 2013, migration to ordinary fat clients was triggered by update from SL4 to SL6. Intel NUC was selected based on result of preliminary investigation and use experiences of those thin clients. Its evaluation was carried successfully out during commissioning of MR. Thin clients were replaced by 22 NUCs that work as fat clients. Migration scenario and technique of managing console system are discussed.

INTRODUCTION

Performance improvement of computers and its price reduction are quite remarkable in recent years. The performance of a PC as small as palm-size surpass that of a server computer 5 years ago. In this paper, renovation of console computers for J-PARC MR which would withstand next 5 years is described.

The accelerator control system for MR was constructed in 2007 [1] using EPICS [2] and SL4 [3] as its control software framework and operating system, respectively. It has been operational since beam operation of MR started [4] in May 2008. GUI applications of the control system were designed and implemented using EDM and MEDM, which require X Window System. In order to achieve load balancing, the control system consists of thin clients which work as X terminals and blade servers to run those applications. HP Compaq t5720 and t5730 were introduced as thin client. They are diskless machines which boot from network using an image file common to all of them so that manageability would be advantageous. IBM BladeCenter HS20 and HS21 were introduced as blade server.

RENOVATION OF CONSOLE SYSTEM

Performance of MR has been steadily improving during 6 years of beam operation. At the same time, number of blade servers was increased [5] as functionality of applications for accelerator control became rich. However, the operation method was unchanged since 2007, which uses thin clients to display the GUI and blade server to the CPU.

Model Selection and Evaluation

In 2012 support for SL4 was ended. This triggered selection for new console computers that would withstand next



Figure 1: Photograph of HP Compaq t5730 (left), Intel NUC (right), and a D Battery for Size Comparison (center). T5720 is not shown but its appearance is almost same as t5730.

5 years. An Intel NUC DC54327HYE running 64-bit SL6 was evaluated in commissioning of MR from December 2013 to January 2014.

Following points are taken into account in the selection based on accelerator operational experience:

- Sufficient CPU power and amount of memory. Increase of demands to use heavy applications such as web browsers, PDF viewers, and office suites led to high load on blade servers. It is also foreseen that even more CPU power and memory are required in near future as GUI environment for EPICS will transfer from EDM and MEDM to Control System Studio (CSS) [6] which is becoming mainstream. Hence, it was considered that run applications locally on a console computer rather than remotely.
- Compatibility with SL6. Operating systems for blade servers have been migrated from SL4 to SL6 and EPICS I/O Controllers (IOCs) [7] underway. Using identical operating system for console will simplify its management and enables smooth transition.
- GPU support by the operating system. Matrox EpicA TC2 / TC4, which are graphics chips of previous thin clients, are not supported by SL. Proprietary device driver from the manufacture is available for SL4 and SL5, but update was stopped in 2009. No driver is available for SL6. Next model will be used for accelerator operation over next few years. Therefore a model was selected such that its GPU is supported by the operating system out of the box.

* shuei@post.kek.jp

Table 1: Specifications of Console Computers

Manufacture	Intel	HP Compaq	HP Compaq
Model	NUC DC53427HYE	t5730	t5720
CPU	Intel Core i5-3427U	AMD Sempron 2100+	AMD Geode NX 1500
CPU Frequency	1.8 - 2.8 GHz	1 GHz	1 GHz
Number of Cores	2 (4threads)	1	1
Amount of Memory	8 GB	512 MB	512 MB
GPU	Intel HD Graphics 4000	Matrox EpicA TC2 / TC4	Matrox EpicA TC2 / TC4
Maximum Number of Monitors	3	2 or 4	2 or 4
Boot Media	SSD	Network (PXE)	Network (PXE)

- Network booting and/or diskless not anymore mandatory.

Diskless SL6 needs more than twice as much RAM as the size of boot image when booting from Live-CD ISO file [8]. Another choice for diskless system is NFS-mounted root filesystem, but operation of a console running SL6 becomes extremely slow under high load condition with NFS root.

- Space-saving and multi-monitor output.

Configuration of previous consoles were dual display or quad display so that space in central control room is used effectively. At least two monitor outputs are required to the next generation console computer.

- Hang up problem.

There was a difficulty in thin clients that hangs up at a rate of one unit per month. This problem occurred as frequent as interfering accelerator operation. On the other hand it was as rare as unable to determine the cause.

Since both NUC and blade server use SL6 as their operating system and X Window System which is network transparent, there was no difference in operation whether an accelerator control application is running locally on NUC or remotely on a blade server. No hang up happened during evaluation.

The result of evaluation was satisfying and Intel NUC was selected as successor mode. Table 1 shows specifications of previous and next generation console computers. Figure 1 shows their appearance.

Migration to New Console

Replacing all the thin clients all at once would require careful preparation period. This method also may face unexpected problem which was not foreseen in the evaluation, and may interfere accelerator operation. Therefore console computers were replaced on the fly during accelerator operation. One or two units per week were replaced so that number of troubles to solve would be suppressed.

Replacement was started in January 2014 and 19 thin clients out of 20 were successfully replaced until summer shutdown of MR July 2014. Finally 20 thin clients were replaced by 22 NUCs. Number of console was increased in order to deal with shortage of consoles during accelerator commissioning. Figure 2 shows a screenshot of NUC in

operation during MR commissioning. No major problem such as hang up happened so far during operation of MR from January 2014 to June 2014.

Role of blade servers is changing. They had been platform for running applications. Importance of blade server is increasing as a host computer of virtual machines (VM) in which IOCs are running [9, 10]. More than 30 IOCs running on VMs are used for MR operation, and its number is increasing.

Installation of Operating System

Operating system are installed to new consoles by replicating disk image from master copy. An image of entire disk is occasionally extracted from a reference console and stored on disk server. When a brand-new console is booted from a USB flash drive, disk image is fetched from the server and restored into its SSD.

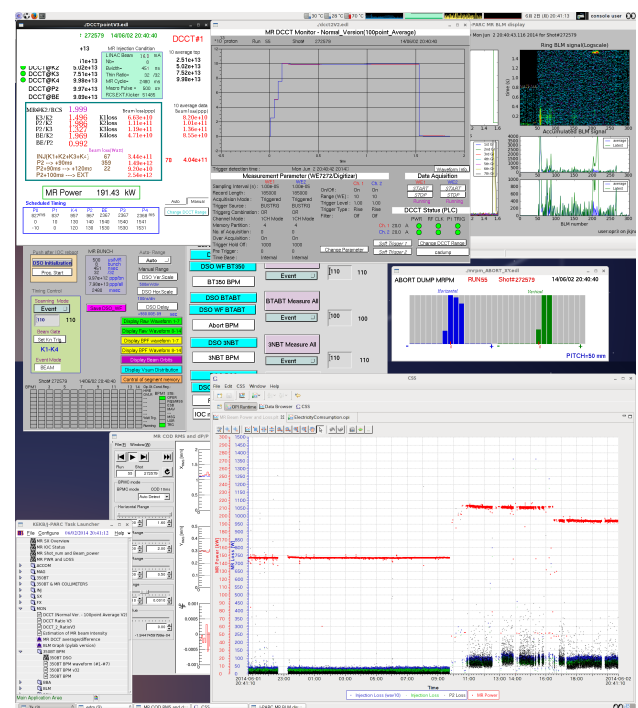


Figure 2: A Screenshot of Intel NUC During Commissioning of MR.

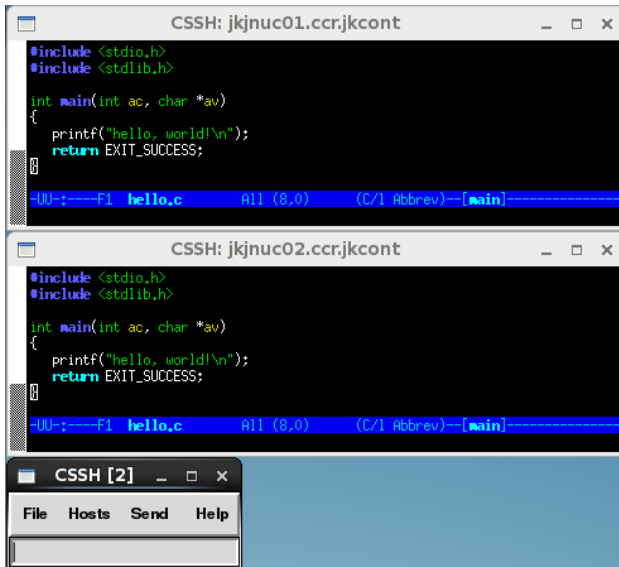


Figure 3: Screenshot of cssh. Two hosts are under control via ssh.

Management of New Consoles

New consoles are not managed collectively anymore, since they boot from their own SSD. Two distributed command execution programs are employed to manage new consoles.

One is Cluster SSH (cssh) [11], which runs interactive commands on multiple hosts over ssh. Cssh opens a control window and a xterm per each specified hosts. In each xterm user is automatically logged into specified host via ssh. Any keyboard input to the control window is replicated to all xterms. Thus inputs to all the xterms under control are kept in sync. Figure 3 shows screenshot of cssh controlling two hosts.

The other is Parallel SSH (pssh) [12], which runs the same command via ssh in parallel on number of hosts. Exit status of the command is shown in the terminal. Standard output and standard error of the command can be displayed in the terminal as each host completes, or they can be saved into a file for each host. Figure 4 shows screenshot of pssh running in a terminal.

SUMMARY

Console computers for J-PARC MR operation were renovated. Intel NUC was selected and its evaluation was carried successfully out during commissioning of MR. 20 thin clients were replaced by 22 NUCs on the fly during accelerator operation. New operation method was introduced so that applications run locally run on console rather than on

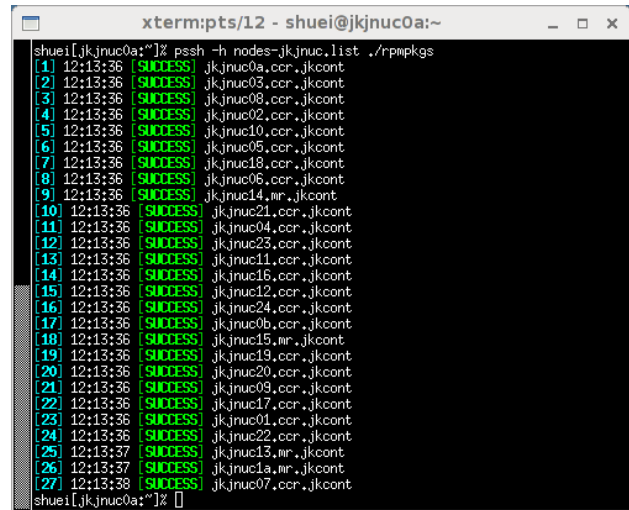


Figure 4: Screenshot of pssh. Exit status of the command is shown in the terminal.

blade servers. No major problem such as hang up happened so far during operation of MR from January to June 2014.

REFERENCES

- [1] S. Yoshida, *et al.*, “Console System Using Thin Client for the J-PARC Accelerators”, Proceedings of ICALEPCS 2007, p383, 2007.
- [2] EPICS - Experimental Physics and Industrial Control System, <http://www.aps.anl.gov/epics/>
- [3] <http://www.scientificlinux.org>
- [4] H. Kobayashi, “Beam Commissioning of the J-PARC Main Ring”, Proceedings of PAC09, p.1823, WE1GRI02, 2007.
- [5] N. Kamikubota, “Review of Control Resources for J-PARC Accelerators”, Presentation in PCaPAC 2012, WEIB02, 2012
- [6] CSS - Control System Studio, <http://controlsystemstudio.org/>
- [7] S. Yamada, “Upgrade of software toolkits for EPICS Input Output Controllers in J-PARC Main Ring”, Proceedings of the 10th Annual Meeting of Particle Accelerator Society of Japan, p.1106, SUP080, 2013.
- [8] <http://www.livecd.ethz.ch/diskless.html>
- [9] N. Kamikubota, *et al.*, “Virtual IO Controllers at J-PARC MR using Xen”, Proceedings of the ICALEPCS 2011, Grenoble, France, p.1165, 2011.
- [10] N. Kamikubota, “Experience of Virtual Machines in J-PARC MR Control”, Proceedings of the ICALEPCS 2013, San Francisco, USA, MOPPC131, 2013.
- [11] <http://sourceforge.net/projects/clusterssh/>
- [12] <https://code.google.com/p/parallel-ssh/>

CONTROL SYSTEM OF TWO SUPERCONDUCTING WIGGLERS AND COMPENSATION MAGNETS IN THE SAGA LIGHT SOURCE

Y. Iwasaki[#], SAGA Light Source, Tosu, Japan

Abstract

The SAGA Light Source (SAGA-LS) is a synchrotron radiation facility. Three insertion devices: a 4 T superconducting wiggler, an APPLE-II undulator, and a planar undulator, are used for synchrotron radiation experiments. For the further demand of hard x-ray experiments, we are investigating to install a second superconducting wiggler. To compensate strong tune shift caused by the superconducting wiggler, a part of the quadrupole magnets families will be excited by independent power supplies. We developed the control system of the storage ring power supplies for the superconducting wiggler. The PLC used for control of the new quadrupole magnets power supplies are linked by optical fiber cable to synchronize each main power supplies in the storage ring. We tested the synchronicity of the PLC sub unit for the new quadrupole magnets in the test bench.

SAGA LIGHT SOURCE

SAGA-LS is a synchrotron radiation facility consisting of a 255 MeV injector and a 1.4 GeV storage ring [1]. The electrons from the linac are injected into the storage ring, and ramped up to 1.4 GeV in 4 minutes. The user mode operation for synchrotron radiation experiments with a stored current of 100 mA has been performed since 2006 [2], and the further developments of the accelerator have been made [3]. To meet the needs of the hard x-ray experiments a 4 T superconducting wiggler was installed in 2010 [4,5]. We perform the accelerator operation at the maximum stored current of 300 mA at this stage. Fig. 1 shows the layout of the accelerator complex of the SAGA-LS and beam lines. For the further requirements of the hard x-ray experiments, we began to investigate a second 4 T superconducting wiggler. The control system of the second superconducting wiggler and compensation magnets power supplies system are developed.

CONFIGURATION OF THE POWER SUPPLIES

The configuration of the quadrupole magnet family (QF1 Family) is shown in Fig.2. To correct of the strong tune shift (Horizontal: -0.031, Vertical: 0.068 measurement result [5]) caused by the superconducting wiggler, a part of the quadrupole magnets are excited by the independent power supplies. The lattice of the SAGA-LS storage ring is 8 symmetry Double Bend type, and there are 3 quadrupole families (QF1, QD1, and QF2 families). A pair of quadrupole doublet (QF1 and QD1) is used for the tune correction. The new quadrupole magnets power supplies for the tune correction must work

simultaneously to the main power supplies of the storage ring magnets during the ramp up period, because the new quadrupole power supplies constitute a part of the main quadrupole magnets families of the storage ring.

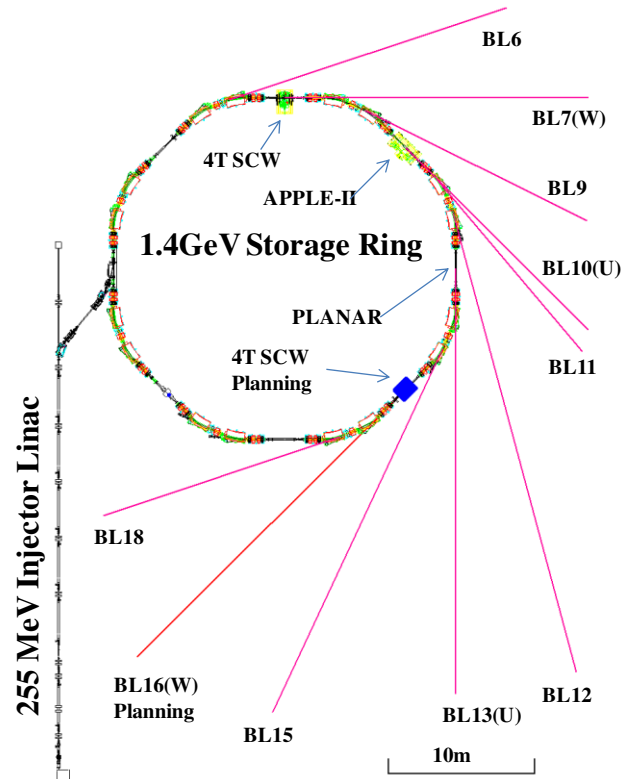


Figure 1: Layout of the SAGA Light Source.

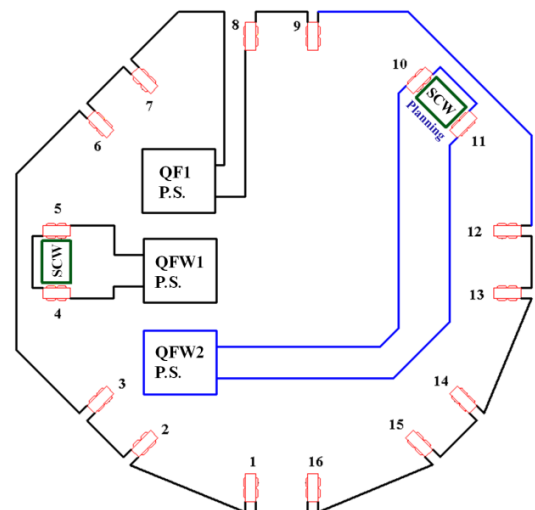


Figure 2: Configuration of QF1 Power Supplies Family. The blue lines indicate modifying section.

[#]iwasaki@saga-ls.jp

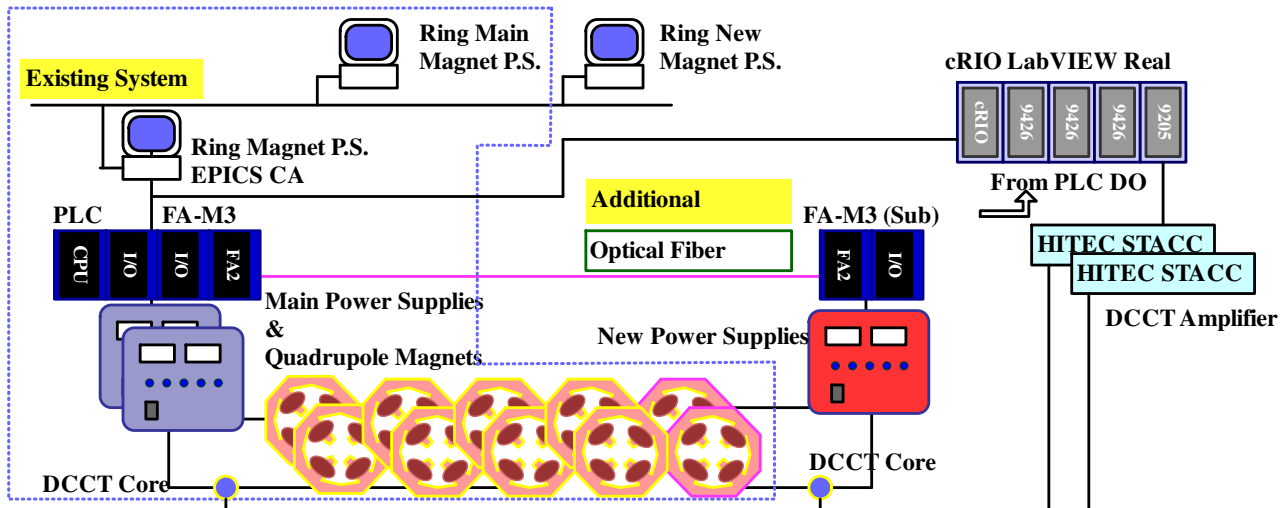


Figure 3: Existing power supplies control system and developing power supplies control system. Sub unit of the PLC is connected by the optical fiber cable to main PLC unit to synchronize to main PLC unit. The external DCCT of the power supplies are also shown. The external DCCT are used for feed-back current control system and accurate monitoring of the power supplies.

CONTROL SYSTEM

Superconducting wiggler

The 3 pole hybrid type superconducting wiggler [4,5] is demagnetized in the injection time, and raised to 4 T after ramp up. The excitation pattern files of the power supplies of the superconducting wiggler are stored in the PLC file memory. The pattern files of the side pole magnets were determined to minimize the closed orbit distortion. The control system of the second superconducting wiggler will be constructed same configuration to the existing 4 T superconducting wiggler.

Quadrupole Magnets

The quadrupole magnets and another power supplies are excited for injection to the storage ring. After the injection the power supplies are excited up to 1.4 GeV in 4 minutes. During the ramp up period, 10000 set points of the power supplies are outputted from the file memory of the PLC. The strobe signal for the set point of the power supplies are outputted in every 10 msec (the width of the strobe signal is 2 msec). The new quadrupole power supplies for the superconducting wiggler must work simultaneously to the main power supplies and used for correction of the strong tune shift caused by the superconducting wiggler. The requirement to the control system of the new quadrupole power supplies are synchronicity to the main power supplies and independent output current control. The PLC for the new quadrupole magnet power supplies are connected to the PLC unit by the optical fiber cable to constitute the sub unit of the main PLC as shown in Fig. 3. The pattern file of the new

quadrupole magnet power supplies for the ramp up is included in the main PLC file memory.

A PC server are used to control PLC and EPICS server [6,7]. Each client PC in the control room can communicate to the server PC by the EPICS Channel Access (ActiveXCA). PC based control system have been developed on National Instruments LabVIEW. Remote desktop service of The MS Windows is also used.

External DCCT Feed-back System

The output currents of the main power supplies in the storage ring are measured by external DCCT (HITEC STACC) for the accurate monitoring. Slow (1 Hz) feed-back the current control by external DCCT and National Instruments Field Point FP1601 AI110 (16 bit ADC) have been performed in the existing system (See Fig. 4). The same specification of the external DCCT for the new quadrupole magnet power supplies are prepared.

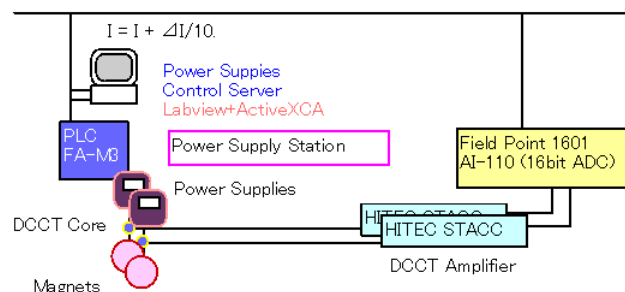


Figure 4: Feed-back Current Control System by External DCCT.

Chromaticity and Coupling Correction

The excitation of the superconducting wiggler causes chromaticity and betatron coupling modulation. To correct the aberrations, 12 pole multipole magnets are installed in vacant long straight section. The magnetic strength of the sextupole and skew quadrupole are 40.04 T/m^2 and 1.71 T/m respectively. The performance of the multipole magnets for the correction was examined with the existing superconducting wiggler. The power supplies of the multipole magnets equip independent PLC CPU, and directly controlled through Ethernet LAN.

TEST BENCH

We constructed the test bench of the PLC (YOKOGAWA CPU: SP76-7S \times 1, I/O:WD64-3P \times 5, FA Bus: LR02-02 \times 2) to confirm the synchronicity of the PLC sub unit. The ladder program of the PLC of the magnet power supplies was modified so as to be adopted with the I/O configuration of the test bench of the PLC. Fig. 5 shows the strobe signals from the Bending magnet power supply, QF1 power supply, and QFW2 (sub module) power supply measured by the YOKOGAWA Logic Analyzer [8]. As shown in Fig. 5, these signals are synchronized within 1 msec.

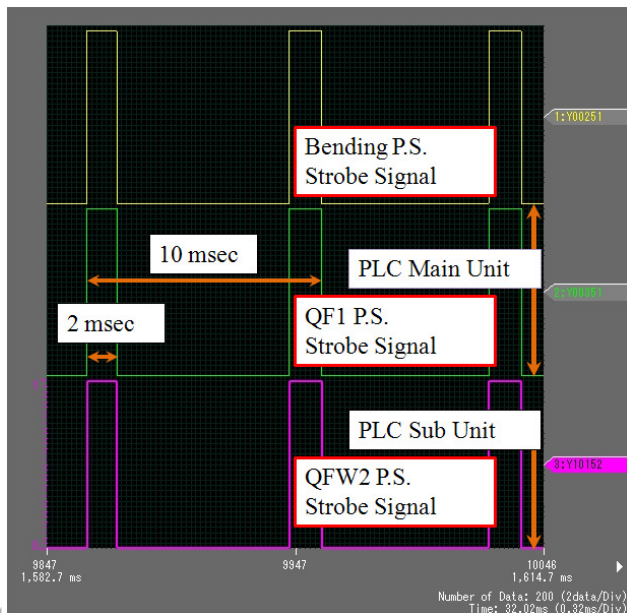


Figure 5: The strobe signals of bending magnet P.S., QF1 P.S., and QFW2 P.S. during the ramp up time.

CONCLUSION

To install the second superconducting wiggler, a part of the quadrupole magnet power supplies will be replaced by the independent power supplies. The power supplies of the quadrupole magnets families in the storage ring must work simultaneously during the ramp up period. We developed the control system of the storage ring power supplies for the second superconducting wiggler. The strobe signals for the ramp up from the PLC sub unit linked by the optical fiber cable were synchronized within 1 msec to the main unit. The 12 pole magnets will be used for the chromaticity and coupling correction.

REFERENCES

- [1] T. Tomimasu et al., "The SAGA Synchrotron Light Source in 2003", PAC'03, Portland, (2003), pp. 902-904.
- [2] K. Yoshida et al., "The SAGA Light Source", Synchrotron Radiation Instrumentation, AIP Conference Proceedings, vol. 879, (2006), pp. 179-183.
- [3] T. Kaneyasu et al., "Present Status of Synchrotron Radiation Facility SAGA-LS", IPAC'09, Vancouver (2009), pp. 2294-2296.
- [4] S. Koda et al., "Design of a Superconducting Wiggler for the SAGA Light Source Storage Ring", IEEE TRANSACTIONS ON APPLIED SUPERCONDUCTIVITY, 21, 32 (2011).
- [5] S. Koda, et al., "Effects of a hybrid superconducting three-pole wiggler on the stored beam at the SAGA-LS storage ring", Nucl. Instrum. Methods A682, 1(2012).
- [6] ActiveXCA Tools, <http://ics-web.sns.ornl.gov/kasemir/axca/ActiveXCA60.zip>
- [7] H. Ohgaki et al., "PC-LABVIEW BASED CONTROL SYSTEM IN SAGA-LS", PAC'05, Knoxville, (2005), pp.3976-3978.
- [8] Yokogawa Wide Field 3, <https://www.yokogawa.co.jp/itc/Recom/WF3/wf30000.htm>

PERSONNEL SAFETY SYSTEM IN SESAME

M. Mansouri Sharifabad, I. Saleh, A. Ismail, SESAME, Allan 19252, Jordan

Abstract

SESAME (Synchrotron-light for Experimental Science and Applications in the Middle East) is a “third-generation” synchrotron light source under construction in Allan, Jordan. Personnel Safety System (PSS) in SESAME restricts and controls the access to forbidden areas of radiation. The PSS is an independent system which is built on Safety PLCs. In order to achieve the desired Safety Integrity Level which is SIL-3, as defined in IEC 61508, several interlocks and access procedures have been implemented in the system fulfilling characteristics such as fail-safe, redundancy and diversity. Also a system meant for monitoring and diagnostics of PSS is built based on EPICS and HMI. PSS PLCs which implement interlock logic send all the input and output bits and PLC status information to EPICS IOC which is not an integral function of PSS operation. This IOC will be connected to other control system's IOCs to send informative signals describing the status of PSS to the main control system in SESAME. In addition, 5 combined Gamma-Neutron radiation monitors which are distributed around and over the booster area send interlock signals to personnel safety system.

INTRODUCTION

Personnel Safety System is an independent access control and interlock system which interlocks accelerator operation and controls access to shielded enclosure to prevent personnel from exposure to high level of radiation forbidden by the law. PSS is designed based on the implementation of IEC61508 standard for programmable safety systems. In SESAME the safety Integrity Level of PSS interlocks is SIL-3. In addition, redundancy and diversity techniques have been applied to increase the reliability of these safety interlocks. For example to inhibit the operation of Microtron, the safety permissions will be removed from two different parts of the system; Microtron Trigger and Microtron High Voltage Power Supply. The first phase of SESAME PSS controls the Booster tunnel which contains Microtron and Booster ring. Following the adoption of IEC 61508 Standard, SESAME decided to use Allen Bradley GuardLogix® controller which is a dual processor solution that uses a primary controller and a safety partner to achieve SIL-3 [1,2].

PSS DESIGN PRINCIPLES

Based on the experience from other synchrotrons the following design principles have been taken into account:

- PSS must be failsafe and PSS safety interlocks must dump the beam in case of any emergency status so that the safety integrity level equals SIL-3.

- Emergency stop and search buttons must be installed in the area covered by PSS, they need to be easily accessible, clearly labelled and distinguishable.
- Audible and visual warnings should be provided prior to accelerator operation and status indicators should be present to reflect the actual conditions of the machine.
- PSS needs to be a totally independent system and all PSS cables run in dedicated conduits and cable trays which are not shared with any other system [1,3].

PSS SAFETY FUNCTIONS

The PSS has four main states, OPEN; which means there is free access to Booster tunnel, INTERLOCKED; after the search patrol has been completed, RESTRICTED and SECURED.

Booster tunnel is considered cleared of personnel when the search patrol is completed. The search process starts upon generation of Start Search command in PSS cabinet in control room. Search buttons distributed all around the Booster tunnel should be pressed in sequence within certain time limit based on the search path designed by the safety officer. During the search patrol audio warning is broadcasted by the sound system, asking the personnel to leave the tunnel [1,3].

The restricted access function is foreseen to provide short visits of personnel to the tunnel without the need for a new search. Like the Search patrol, the restricted access needs to be permitted by control room. After taking one of the personal keys the person is authorized to enter the Booster tunnel during RESTRICTED access mode (up to two people). In order to return back to INTERLOCKED state all personal keys must be in place on PSS cabinet door, tunnel door locked and the finish RESTRICTED access permitted by control room. The SECURED state exists when PSS is in INTERLOCKED state and the keys on the PSS cabinet in control room are in the position that allows the trigger of Microtron [1,3].

The tunnel door is locked by an electromagnetic lock and its status is monitored by a magnetic switch (SIL-3) and the feedback from electromagnetic lock [1,3].

In emergency cases (e.g. an individual is trapped in the Booster tunnel) they should press one of the red Emergency Stop buttons, distributed around the tunnel and on Booster PSS cabinets. This will immediately trip the interlock system, unlocks the Booster tunnel and stops the injection process [1,3].

To monitor the radiation level in service area, Booster roof and the inner ring of the Booster (pool), five combined neutron-gamma radiation trolleys (by Thermo Fisher) are distributed in the mentioned zones. Three digital signals are continuously provided by each trolley to PSS, monitoring the status of radiation level and error signals. PSS will react accordingly to each of these signals [1,3,4].

PSS SAFETY INTERLOCKS

The interlock devices linked to PSS are as follows:

Microtron PSS Interlocks: Microtron Trigger Interlock and Microtron High Voltage Power Supply Interlock

Booster PSS Interlocks: Booster RF Interlock and Booster Vacuum Gate Valve Interlock

Any of the Microtron PSS interlock devices can cease the operation of Microtron and terminate the injection process while the Booster RF Interlock will stop energizing of the beam and the Vacuum Gate Valve Interlock eliminates the stored beam in Booster. Each of the PSS safety interlocks is applied through phoenix contact PSR-SCP Safety relay which is approved for SIL-3 high and low-demand applications. The configuration shown in the picture below can fulfil the failsafe requirement up to SIL-3 as per IEC 61508 [1,2].

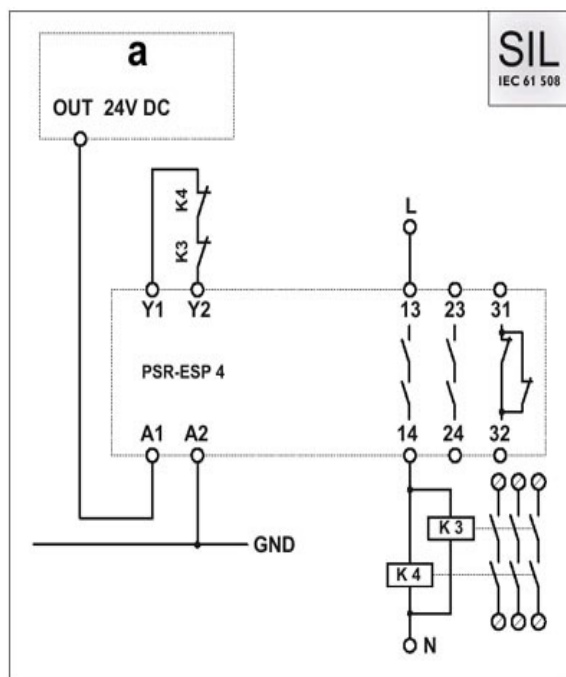


Figure 1: One-channel evaluation of a safety controller with automatic activation, suitable up to SIL-3.

OPERATOR INTERFACE

The operator interface and operation keys are installed on the PSS cabinet in the control room. FactoryTalk® View is the HMI software product used to build a variety of screens on HMI. This system is only designed for monitoring and diagnostics. It does not have write access to the safety data block of PLC. The most critical operation permits are given with physical keys. The status of PSS, radiation monitors interlock signals, beam-on, etc. can be monitored via HMI. To unify the PSS GUI (Graphical User Interface) with other GUIs in control system of SESAME, work is underway to produce PSS operator interface in CSS.

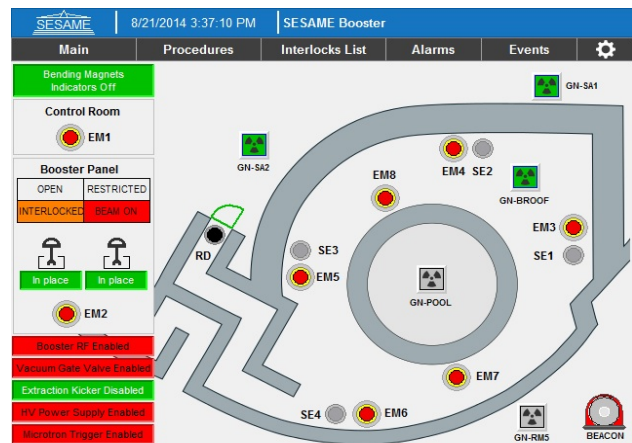


Figure 2: Main GUI OF Booster PSS.

ARCHITECTURE AND TECHNOLOGY

The architecture decided for PSS project is to use safety PLCs as the processing element. Safety PLCs are designed and certified for use as part of a safety function. They have specific design features intended to meet the failure rate, redundancy, and diagnostic requirements of an IEC61508 compliant system. The PSS in Booster phase is built on an Allen Bradley 1756-L72S CPU, a GuardLogix® controller that also provides safety control. The GuardLogix system is a dual processor solution that uses a primary controller and a safety partner to achieve SIL-3. A major benefit of this system is that it is a single project, with safety and standard control together.

Another basic design feature is to use a distributed remote I/O architecture. The Guard I/O modules implement the CIP-safety protocol extensions over EtherNet/IP networks and provide various features for the safety system. CIP Safety, the extension to the Common Industrial Protocol (CIP) for functional safety applications on the EtherNet/IP networks, provides fail-safe communication between nodes, such as safety I/O blocks, safety interlock switches and safety PLCs in safety applications up to Safety Integrity Level SIL-3. Safety application coverage in CIP provides the ability to mix safety devices and standard devices on the same network or wire for seamless integration and increased flexibility. Because the safety application layer extensions do not rely on the integrity of the underlying standard CIP services and data link layers, single channel (non-redundant) hardware can be used for the data link communication interface. This same partitioning of functionality allows standard routers to be used to route safety data. The routing of safety messages is possible, because the end device is responsible for ensuring the integrity of the data. If an error occurs in the transmission of data or in the intermediate router, the end device will detect the failure and take an appropriate action [5].

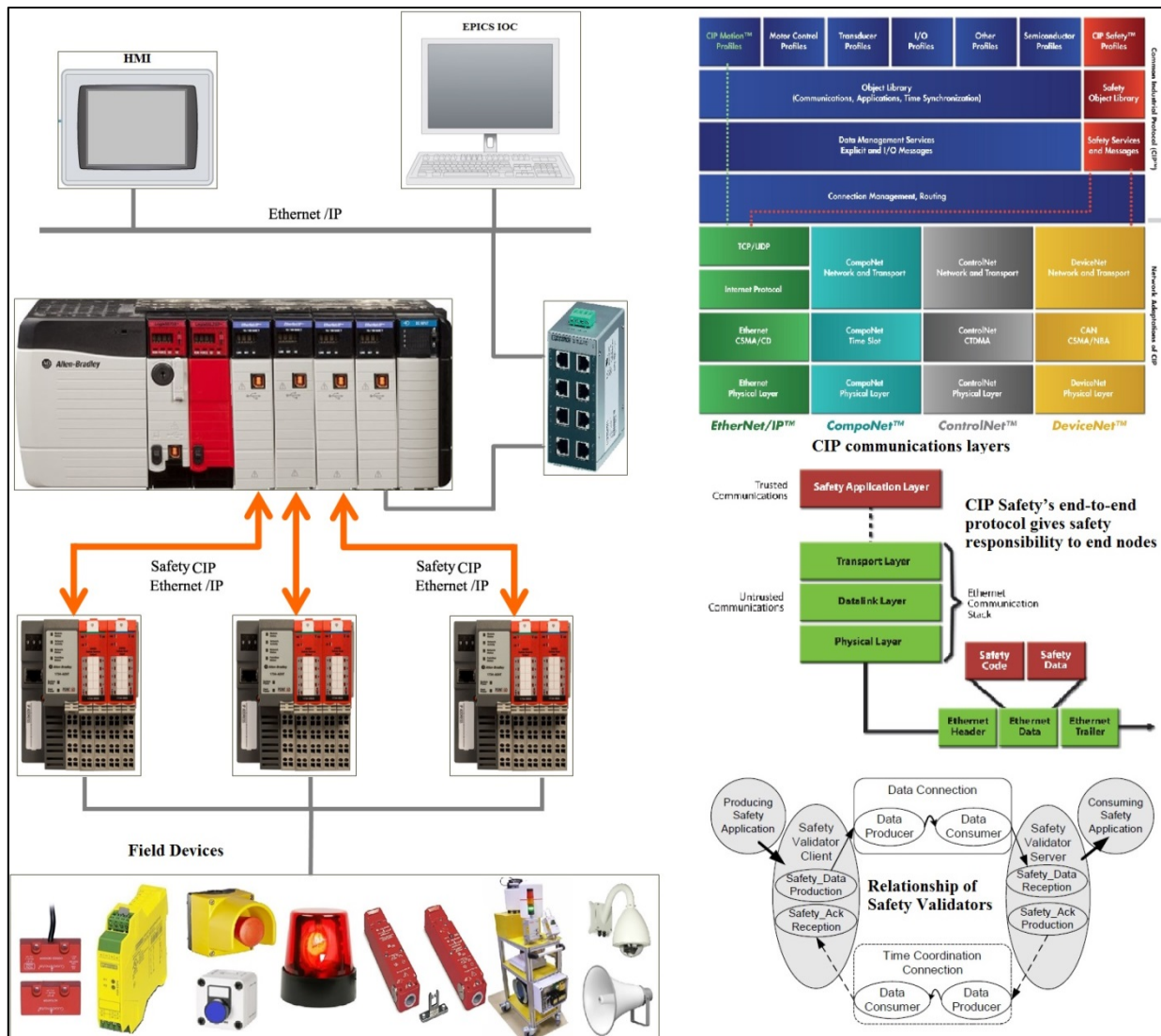


Figure 3: Network Layout of Booster PSS.

SUMMARY AND FUTURE WORK

The Personnel Safety System in SESAME is established on safety PLC based system to meet the requirements of IEC 61508 standard and the design logic of this system is fulfilled with the safety interlock principles. The design uses a distributed safety architecture that increases reliability and lowers costs. The successful implementation of this system has increased the operation safety of SESAME facility. Lots of work is still underway to develop other phases to cover Storage ring and 3 day-one beam lines in SESAME.

ACKNOWLEDGMENT

Many people have contributed in this project. I would like to thank Erhard Huttel, Technical Director, Adli Hamad, the safety officer and control team in SESAME.

Also I would like to thank D. Fernández and A. Robi in ALBA for their kind help and support.

REFERENCES

- [1] ALBA Synchrotron, CELLS, Bellaterra, Barcelona, Spain
- [2] International Electro-technical Commission, IEC 61508, "Functional Safety of Electronic/Programmable Electronic Safety-Related Systems" International Electro-technical Commission, Geneva, 1998
- [3] D. Fernández-Carreiras, "ALBA, THE PLC BASED PROTECTION SYSTEMS", Proc. ICALEPCS2009, <http://jacow.org/>.
- [4] <http://www.thermofisher.com>
- [5] <http://ab.rockwellautomation.com>

CLIENTS DEVELOPMENT OF SESAME'S CONTROL SYSTEM BASED ON CSS

A. Ismail, I. Saleh, M. Mansouri, Y. Dabain, SESAME, Allan, Jordan

Abstract

SESAME is a third generation synchrotron light source under construction near Amman (Jordan). It is expected to begin operation in 2016. SESAME's injector (Microtron) and pre-injector (Booster Ring) have been commissioned. Commissioning of the storage ring is expected in 2016. The control system at SESAME is based on EPICS. EPICS IOC's are used for the servers. Control System Studio (CSS) is used for the clients. CSS BEAST alarm handler is used to identify all the critical alarms of the machine including configuration and visualization. This paper presents the architecture and design of the CSS BOY graphical user interfaces (GUIs) and CSS BEAST alarm handler for the different subsystems. It presents the standards followed in the development of SESAME's clients. SESAME will use an archiving tool based on CSS to access process variable history.

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 are 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. This paper will focus on the design and implementation of SESAME's clients based on CSS.

CUSTOMISED CSS

Control System Studio (CSS) is a combined effort of several parties, including DESY (Hamburg, Germany) and SNS (Oak Ridge, TN). It provides a collection of control system tools in a common environment, based on Eclipse [1]. Control System Studio (CSS) is designed to serve as an integration platform for engineering and

operation of today's process controls as well as machine controls systems. CSS is a complete environment for the control room covering alarm management, archived data displays, diagnostic tools and last but not least operator interfaces [2].

A custom build of Control System Studio (CSS) based on V.3.16 has been implemented in SESAME (Fig. 1). Archive system and alarm handler plugins have been integrated to the CSS custom build as they were not included in the basic build. A thumbwheel function has been added to the input box widget, which provides a compact fine-tunable control to the set values and it replaces the old thumbwheel widget (Fig. 2). A digit of a set value is marked and changed by using the arrow keys of the keyboard. The custom build also includes a new feature for showing the CSS screens name on the title bar of each screen. Another feature has been added when multiple CSS windows are opened, only one instance of identical windows is allowed at a time.

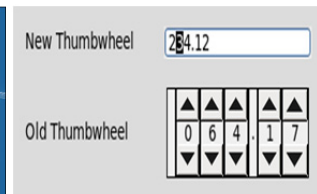
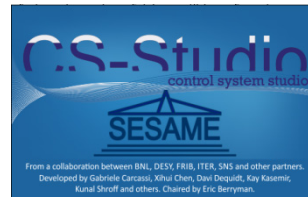


Figure 1: CSS Start Screen. Figure 2: New Thumbwheel.

OPERATOR INTERFACES

SESAME uses CSS BOY (Best OPI, Yet). It is an Operator Interface (OPI) development and runtime environment which enables monitoring and controlling of an EPICS system. It has a modern graphical editor and a modern web browser style runtime. It is dynamic via rules or scripts and it has comprehensive types of widgets. SESAME has two environments of CSS; one is used for development in which OPIs can be edited and modified, and the other is used for deployment which disables modification and hides all the development windows of eclipse. SESAME's client development is maintained and tracked using a version control system called Git. Git is a distributed revision control and source code management system [3]. Clients' repository contains CSS core build and CSS workspaces that contains all the developed OPIs held in main folders. SESAME OPIs are arranged and classified depending on the machine main parts; there are three main folders which contain all the required OPIs for the Microtron, TL1 and Booster control systems. The booster control system is divided into seven subsystems; each one has its own sub-folder and OPIs for the control

system. The Microtron and the TL1 also have their subsystems.

SESAME's main OPI (Fig. 3) represents a launcher which can access all of the subsystems of the machine each on a separate OPI window. Each subsystem can be monitored and controlled using its specific OPI. The main OPI has different action buttons which are connected to different OPIs specified for the subsystems.

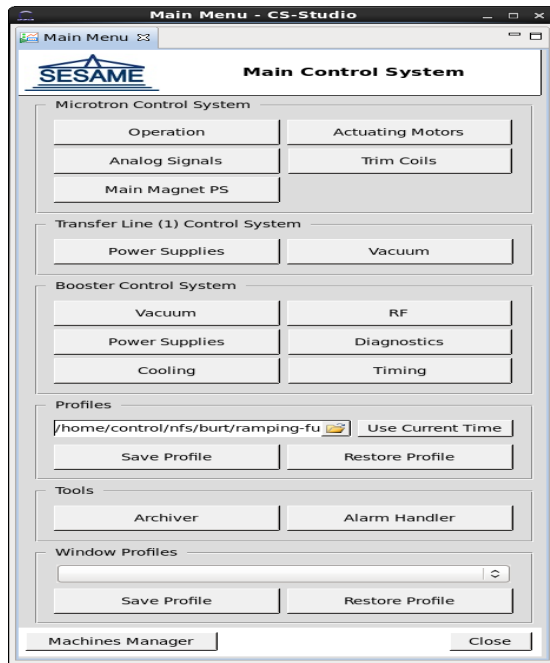


Figure 3: Main OPI.

The main OPI includes other action buttons connected to different functions such as:

- Profiles Save/Restore: Based on BURT (Back Up and Restore Tool) which is a tool for saving the current state of a list of Process Variables (PVs) at a given moment and restoring them at a later date [4]. It is regularly used by the operators.
- Window Profiles Manager: It is a python script, developed by SESAME, running under CSS which can open, move and close windows according to a previously saved windows workspace profile. A window profile contains the list of open windows and their OPI file paths, macros used in these windows and their screen positions.
- Machines Manager: It is an EPICS IOC developed by SESAME to manage different machines and IOCs. It can enable/disable IOCs, synchronize IOCs with the repository and monitor the uptime, free RAM, free space and the average load of the IOCs.
- Archive system and alarm handler can also be accessed from the main OPI.

SESAME OPI STANDARDS

SESAME started using standards for the design and the layout of different OPIs. These standards will give the operators simpler and clearer view to the OPIs. Standards

are used for windows, widgets and colors. Figure 4 shows an example how these standards are applied.

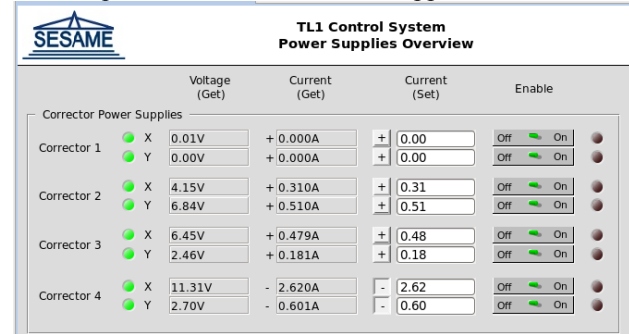


Figure 4: OPI for the TL1 Power Supplies.

Windows

- Name property must be set (which is used as a window's title).
- Same background color specified in color.def should be used.
- Header should be white color. It should contain SESAME's logo and a window's title.

Widgets

- Text Update: This widget is usually used for monitoring getter values. It should have a transparent background. Its border style should be Etched Style.
- Text Input: This widget is mainly used for setting PV values. It should have the system's default background color. Its style should be Native. Thumb Wheel Mode should be set if the value is numeric.
- LED: This widget is used to show devices on/off state, interlocks, limit switches and possibly other Boolean signals. Its color should convey the type of signal shown according to the color standards. Its size should be 20x20 pixels.
- Switch (custom widget): It is used mainly to turn devices on and off, although it can also be used for other stuff (e.g. valve open/close command).

Colors

- Color definitions are present in the file color.def within the workspace. This file contains mappings between names and colors, defined as rgb values.
- Current listing in color.def is shown in Table 1.

Table 1: SESAME Color Definitions

color.def listing				
Name	red	green	blue	color
Background	220	220	220	
Readonly Background	220	220	220	
Group Title	0	0	0	
Error Off	60	20	20	
Error On	240	5	5	
Start Off	20	60	20	
Start On	5	240	5	
Limit Switch Off	60	60	20	
Limit Switch On	240	240	5	

ALARM SYSTEM

SESAME uses BEAST (Best Ever Alarm System Toolkit). The alarm engine is installed on a virtual machine on the machine network. It monitors PVs for alarms and sends update messages to the connected clients using an Apache ActiveMQ server. The client shows alarms divided by systems and subsystems in a hierarchical way where the state of the division is equal to the state of the most severe alarm underneath it. A notable feature is that alarms stay latched onscreen even if the fault is no longer present, until they are acknowledged. In this way the operator is made aware of both transient faults and steady faults.

We are facing an issue about the choice of the alarms that we want to show on the client. If a large number of alarms are shown, the operators will likely miss the important ones. On the other hand, some low importance alarms may be the key to figuring out a problem in the operation. A balance is still to be found, however, one of the ideas to solve this issue is to show and hide different alarms at different states of the operation. In other words, when the machine is off, for a simple example, some alarms may be normal and hidden.

ARCHIVING SYSTEM

The archiving system is an important part of the control system in any accelerator. SESAME has integrated the BEAUTY (Best Ever Archive Toolset Yet) archiving system. The system is divided into two parts: the archive engine and the data browser. The archive engine is installed on a virtual machine and setup to log the changes of approximately all PVs to a local PostgreSQL database. The value, alarm state and severity are recorded on each change. The data browser, which is the client side, is integrated into CSS as a plugin. It can be used to search for PVs and to plot them on a graph in any specific time interval. Figure 5 shows the architecture of the archiving system with the data browser showing some vacuum readings.

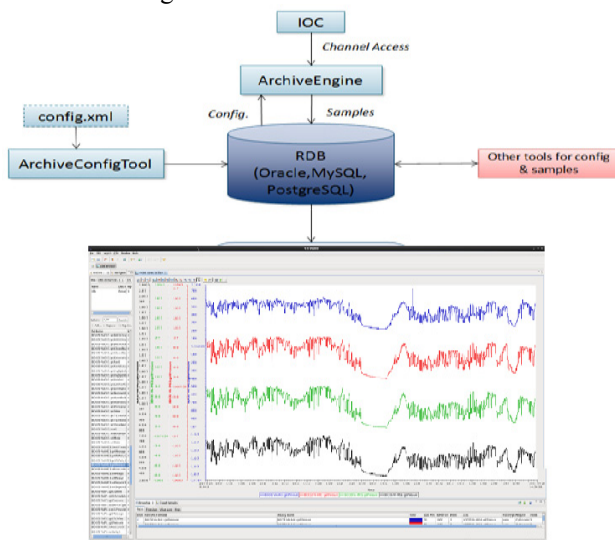


Figure 5: Archiving System Architecture.

CAMERA DRIVER AND CLIENT

One of the in-house developments on the client side is a camera epics client. Currently, SESAME has two Basler acA1300-30gm GigE cameras installed in the booster ring. A driver was developed to interface the Basler cameras with EPICS. 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. A feature that was requested by the operators to ease visual inspection of the beam is the ability to switch between different color maps. The camera client provides the normal grayscale color map and a hot-cold colour map. The client was developed using C, EPICS CA libraries for connecting with EPICS, OpenGL for image rendering, SDL for window creation and input handling and AntTweakBar for controlling the camera's various parameters.

CONCLUSION

SESAME uses a custom build of Control System Studio (CSS) based on V.3.16. Archive system and alarm handler plugins have been integrated to the CSS. CSS adds a complete environment for the control room covering alarm management, archived data displays and operator interfaces. Work will continue to provide compact simple OPI screens for the operators.

ACKNOWLEDGMENT

We would like to thank Pascale Betinelli (SOLEIL), Mark Heron (DLS) and Igor Kriznar (ANKA) for their support. We would like to thank also our technical director Erhard Huttel (SESAME) for his support.

REFERENCES

- [1] DESY: <http://css.desy.de/>
- [2] M. Clausen, J. Hatje, J. Penning, "THE CSS STORY", Proceedings of PCaPAC2012, Kolkata, India, 2012.
- [3] [http://en.wikipedia.org/wiki/Git_\(software\)](http://en.wikipedia.org/wiki/Git_(software))
- [4] <http://aps.anl.gov/epics/extensions/burt/index.php>

THE APPLICATIONS OF OPC UA TECHNOLOGY IN MOTION CONTROL SYSTEM

Min Wang[#], Hong Luo, Min Li, Jinmei Dong, Ruishi Mao, Tiecheng Zhao,
Institute of Modern Physics, Chinese Academy of Sciences,
Lanzhou 730000, China

Abstract

The establishment of data model is more abundant based on OPC UA (Unified Architecture) technology, which has good platform independence and high reliability. Thus it becomes a new direction in the field of data exchange of industrial control. In the paper, the motion control model based on redundant ring network is built by using NI 3110 industrial controller and servo motors. And the data structures used in parallel communication between the upper computer and multi-terminal motors are designed by using OPC UA technology. So the problem of inconvenient data exchange between the RT system of lower controller and the Windows system of upper computer may be solved.

INTRODUCTION

Motion control is widely used in modern industrial automation. The design is based on the application of the heavy ion accelerator in Lanzhou, China, which is firstly used in motion control of four beam diagnostic detectors. OPC UA (OPC Unified Architecture) technology with the advantages of good platform independent, security and so on, has become the main means to the data interaction in present industrial control, and is widely used in distributed control system.

THE ADVANTAGES OF OPC UA TECHNOLOGY

The traditional OPC specification is based on the COM/DCOM technology of Microsoft, which cannot meet the requirements of modern industrial automation in the aspect of interoperability, security, reliability and so on [1]. So OPC foundation released the newest unified method of data communication, namely OPC UA, which not only covers OPC DA, OPC HDA, OPC A&E, OPC security, but also expands many new functions [2].

Compared with the traditional OPC technology, OPC UA has the following advantages:

- Unified access approach, which means it can provide a unified address space and service model, but also has the function of semantic recognition, solves the problem that is the same information cannot be accessed in a unified way.
- Reliability, which means adjustable timeout can make the mistakes found and corrected, all this makes us deal with errors and failures of

communication more easily.

- Security, which means the technology of underlying communication used for message transfer between the applications based on OPC UA provides the functions of encryption and marking.
- Independency of platform, which means OPC UA technology based on the SOA (Service-Oriented Architecture) of Web Service makes the application development no longer dependent on any particular operating system.
- Relevance of data, which means OPC UA provides the correlation functions of data nodes, rather than puts the node as a single data point [3].

THE ARCHITECTURE OF MOTION CONTROL SYSTEM

The motion control system is used for the motion control of beam diagnostic detectors of the cyclotron. The beam diagnostic detectors contain one SS (scintillation screens), one FC (faraday cup) in LEBT (low energy beam line), and another FC, one scraper in MEBT (medium energy beam line), so five servo motors are needed for the motion control (see Fig. 1).

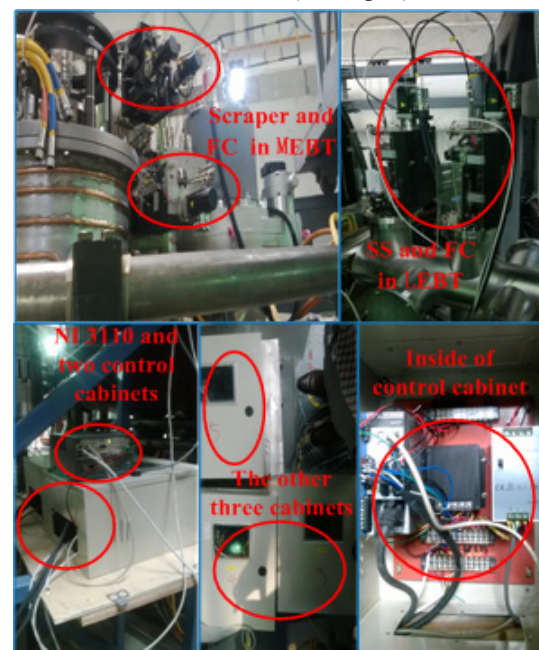


Figure 1: Photos of the field in motion control.

Because Kollmorgen servo drive supports EtherCAT communication protocol [4], which has obvious advantages in topological structure, clock

[#]wangmin@impcas.ac.cn

synchronization, data transmission speed, construction cost and so on compared with other communication protocols, and the NI company has developed many LabVIEW software toolkits based on Kollmorgen servo drive, which can greatly reduce the difficulty and period of the development of upper application software, so Kollmorgen servo motor and NI 3110 industrial controller is selected for the motion control system.

The motion control system is composed of a plurality of servo drives and motors, a NI 3110 industrial controller and a client computer (see Fig. 2). Each servo drive is connected to a servo motor, servo drive communicates with each other through the EtherCAT interface. In order to build the redundant ring network of EtherCAT and improve reliability of the system, the first drive and the last drive are connected to the output and input of the EtherCAT interface ports of NI 3110 industrial controller, which makes NI 3110 communicate with each drive [5,6]. Because of the good platform independence and security of OPC UA technology, NI 3110 has the RT system is used as the server to display the real-time data and receive control commands from the client has the different Windows systems in the form of OPC UA.

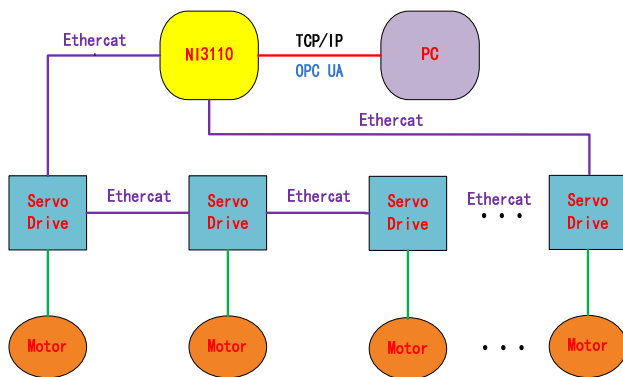


Figure 2: Architecture of motion control system.

The system can also be extended to a variety of topologies, such as star, tree and daisy chain type etc. Multiple NI 3110 can be connected through the EtherCAT protocol, and each NI 3110 also can be connected with any other hardware devices supporting the EtherCAT, not only the servo drives. The data from NI 3110 can be displayed by OPC UA, and the commands from multi-client computers are received by OPC UA too.

THE DATA STRUCTURE USED FOR MOTION CONTROL BASED ON OPC UA

Because OPC UA provides the correlation functions and the semantic characteristics of data nodes, so it is very necessary to design the data structure of the motion control system based on OPC UA, which will make the program easier to maintain, enhance the readability of the program, and facilitate the development of client program.

Based on the OPC UA application of LabVIEW, the data structure can be freely built according to the different

applications. The data structure based on OPC UA for the servo motor named “motor_1” is designed, which is similar to the other servo motors (see table 1). The *Folder* node is used to distinguish each servo motor; the *Item* node is used to sort the data for motion control based on OPC UA, each *Item* node can define the name, access method, data type and description; the *Property* node belongs to the corresponding *Item* node, and also contains the definition of name, access method, data type and description, which do not appear in table 1 as a result of length reason.

Table 1: Data Nodes of the System

Folder Name: motor_1	Item_1 Name: data Access: read only Datatype: float Description:	Property_1 Name: position Property_2 Name: scale_value
	Item_2 Name: command Access: read/write Datatype: int32 Description:	Property_1 Name: enable Property_2 Name: start Property_3 Name: stop Property_4 Name: home Property_5 Name: distance
	Item_3 Name: status Access: read only Datatype: boolean Description:	Property_1 Name: enable? Property_2 Name: complete? Property_3 Name: fwd_limit? Property_4 Name: rev_limit? Property_5 Name: scale_safe?

From the table 1, the *Item* of “motor_1” is divided into three categories, which are named with “data”, “command” and “status”. The “data” item includes two properties, which are named with “position” and “scale_value” – “position” means the position feedback from servo drive; “scale_value” means the value of displacement sensor acquired from the digital I/O of servo drive. Because the motor encoder is relative encoder, in order to improve the accuracy of position acquisition and bring convenience for calibration of detectors, resistance displacement sensor is used to get the correct absolute position of motion. The “command” item includes five properties, which are named with “enable”, “start”,

“stop”, “home” and “distance”. “enable” is used to enable the motor; “start” and “stop” is used to start and stop the motion tasks; “home” is used to find the motion origin; “distance” means the distance of motion tasks. The “status” item includes five properties, which are named with “enable?”, “complete?”, “fwd_limit?”, “rev_limit?” and “scale_safe?”. “enable?” means whether the motor is enable; “complete?” means whether the motion task has been completed; “fwd_limit?” means whether the positive limit switch is triggered; “rev_limit?” means whether the negative limit switch is triggered; “scale_safe?” means whether the value of displacement sensor has been out of the safe range.

The control interface of the client upper computer is shown in the Fig. 3. From the Fig.3, the motion control interface of SS in MEBT is divided into two parts. The left part named “Motor_1 Control” of the interface with the functions of enabling the Motor, finding the origin, starting and stopping the motion task, inputting the distance of motion, observing the feedback of motion, getting the value of displacement sensor and calculating the distance between the detector and the centre of chamber is used to complete motion tasks. The right part named “Motor_1 Status” is used to get the status containing “enable?”, “complete?”, “fwd_limit?”, “rev_limit?” and “scale_safe?”.

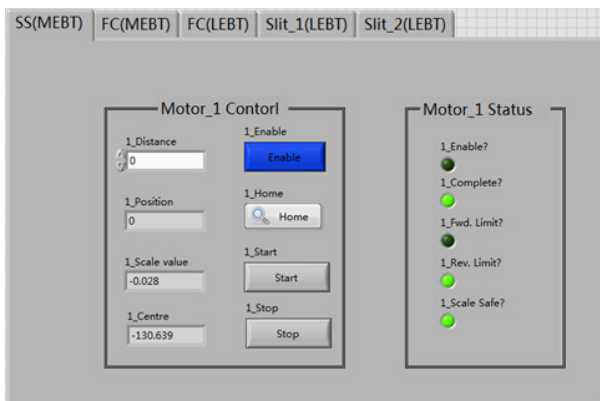


Figure 3: Interface of motion control system.

Thus it can be seen that the flexible and easy way of data organization of OPC UA can make the program of NI 3110 as the lower computer more easily to extend and maintain, but also make the development of program of the client as the upper computer more convenient.

CONCLUSION

With the development of industry control, the data communication needs the better reliability and safety, independency of platform and so on. In the paper OPC UA technology is selected for building the motion control system, the architecture of motion control system and the data structure based on OPC UA are designed, and the motion control system which can verify the advantages of OPC UA technology is being tested.

ACKNOWLEDGEMENT

We would like to thank my colleagues and Dianling Wang, Yao Li, Wenlong Jing from National Instrument for all of their help.

REFERENCES

- [1] M. Wolfgang, L.S. Helmut, “OPC Unified Architecture – the future standard for communication and information modeling in automation”, ABB Review, March 2009, p. 56-61(2009).
- [2] Wikipedia website: http://en.wikipedia.org/wiki/OPC_Unified_Architecture
- [3] L.S. Helmut, M. Wolfgang, “OPC UA Service-oriented Architecture for Industrial Applications”, Software Trends, November 2006, ISSN0720-8928 (2006).
- [4] J. Katzel, M.T. Hoske, “Industry Ethernet Protocol”, Control Engineering China, March 2007, p. 34-39 (2007).
- [5] National Instrument website: <http://www.ni.com/white-paper/11700/en/#toc2>
- [6] EtherCAT Technology Group, “EtherCAT Physics and Slave Controller Integration”, (2008).

THE MEASUREMENT AND MONITORING OF SPECTRUM AND WAVELENGTH OF COHERENT RADIATION AT NOVOSIBIRSK FREE ELECTRON LASER

V. V. Kubarev, S. S. Serebnyakov

Budker Institute of Nuclear Physics SB RAS, Novosibirsk, Russia

Abstract

This paper describes in detail the architecture and capabilities of the system for measurement of the free electron laser (FEL) radiation spectrum. The measurements are performed with a monochromator and a step-motor with a radiation power sensor. The measurements result in the transmission of the curve of the radiation spectrum to the control computer. As this subsystem is fully integrated into the common FEL control system, the results of the measurements – the spectrum graph, average wavelength, and radiation power calculated – can be transmitted to any other computer in the FEL control local area network, as well as to computers of the user stations.

INTRODUCTION

A high-power FEL based on a multiterm energy recovery linac (ERL) [1] is under construction now at Budker Institute of Nuclear Physics. The first and second phases of the project have already been commissioned and are currently in operation.

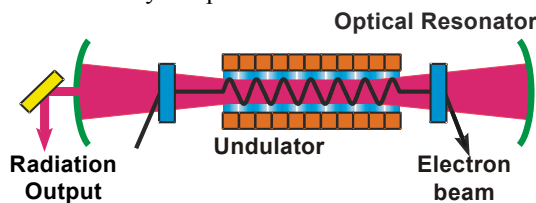


Figure 1: FEL Operation layout.

During its operation, the FEL generates coherent radiation (see Fig.1), which is used for various experiments. The wavelength of this radiation depends on some accelerator parameters and beam energy and is expressed with formula

$$\lambda = \frac{d}{2\gamma^2} \left(1 + \frac{k^2}{2} \right), \quad (1)$$

where λ is the radiation wavelength; d is the undulator period; γ is the relativistic factor of electrons; k is the undulator parameter; $k = k_0 \cdot I$, where I is the current in the coils of the electromagnetic undulator, and k_0 is the constant of proportionality.

As seen from Eq. (1), the radiation wavelength can be tuned via change in the beam energy or adjustment of the

undulator current. Besides, the FEL radiation has a rather narrow spectrum, which depends on different FEL parameters. Therefore, the real-time monitoring of the spectrum, power and average wavelength of the FEL radiation is necessary for operators and FEL users throughout FEL operation. For this purpose, we created a separate system with a monochromator.

HARDWARE AND STRUCTURE OF RADIATION MONITORING SYSTEM

The layout of the system is presented in Fig.2. Its main device is the monochromator, which is used for measurement of the FEL radiation spectrum. The FEL radiation enters the entry window of the monochromator, and the radiation with transverse amplitude distribution corresponding to the input radiation spectrum goes out through the exit window. Thus, to obtain the radiation spectrum, one has to read out the intensity distribution at the exit of the monochromator and transmit it to the computer. That is done with the radiation sensor installed on a support, which is moved horizontally by a stepper motor. The stepper motor controller is connected to the IBM-PC via the RS-485 interface.

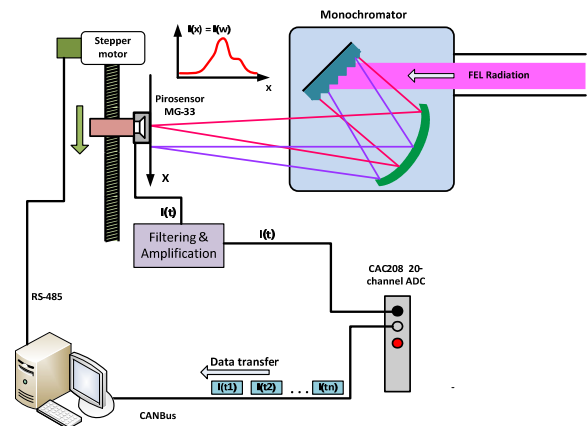


Figure 2: Layout of the system for measurement of the FEL radiation spectrum.

For proper measurement of the radiation power, the flux of measured radiation is modulated with a special rotating shutter. This procedure is necessary because the pyroelectric radiation sensor applied can measure only a variable signal. After some filtering and amplifying, the signal is measured with an ADC. The CAC208 device with CAN interface, developed at BINP [2], is used for

this purpose. The CAN bus line is connected directly to the IBM-PC.

Having the obtained spectrum curve and using some calibration coefficients, one can calculate the power, maximum and average wavelength of the FEL radiation.

SPECTRUM MEASUREMENT AND PROCESSING OF RESULTS

The control software fully governs the process of spectrum measurement. A user command starts the process, after which the program executes the following steps:

1. The support with the radiation sensor is moved to the extreme left position.
2. The command “MOVE” with a specified constant speed is sent to the stepper motor controller, for the radiation sensor to run the entire horizontal range of output radiation from the monochromator.
3. A command is sent to the ADC to start repetitive single-channel measurements of the channel the signal from the radiation sensor arrives at. The time of one measurement is specified as 40 milliseconds. After this command, the ADC device begins measurements of the current channel and immediately sends the measured data to the central PC.
4. The measurement data are received and added to a dynamic array, until the signal “STOP MOVEMENT” arrives from the stepper motor controller. During this process, the control software plots the waveform of the data (see Fig 3.) received before the actual moment.
5. The signal “STOP MOVEMENT” informs that the radiation sensor has reached the extreme right point of the scanning area, the stepper motor has stopped, and the scanning process has been finished. Upon receiving this signal, the control application stops the current ADC operation and begins the processing of the obtained waveform of the spectrum.

Since the speed of the support movement along horizontal axis can be determined easily, the speed of ADC measurements is specified by the user, and the constant of proportionality between the scale on the horizontal axis of the output radiation and the wavelength of the radiation is also well known, it is easy to apply the obtained waveform to the frequency axis.

After executing this sequence, the application changes to the idle state or, if the repetitive mode was set, the application starts a new scanning cycle, i.e. steps 2-4 are repeated. The only difference is that the direction of the movement of the support with the radiation sensor changes to the opposite one, so the control application has to change the initial point on the frequency axis from which the measurement starts and the order of adding of the incoming data from the ADC. The total time of the

movement of the support throughout the range of interest is approximately 50 seconds.

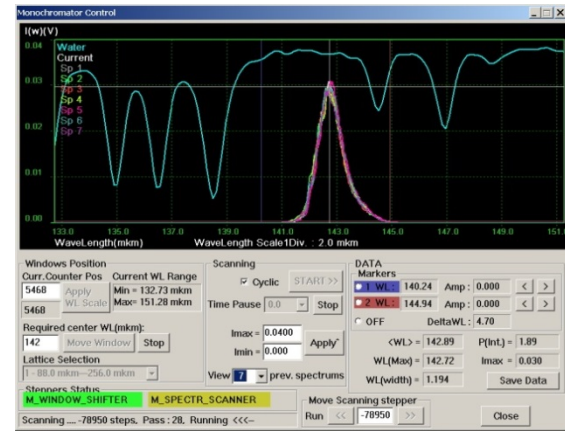


Figure 3: Plot of measured FEL radiation spectrum.

Thus, in the repetitive mode, the control application continuously sends commands for the support to move by turns in opposite directions, receives measurement data from the ADC, and processes and displays these data as the spectrum of the FEL radiation.

In this mode, the application also allows setting the delay time between the completion of the actual support pass (spectrum scan) and beginning of next one. It is possible to set a delay time of up to 120 seconds. This feature allows one to reduce the mechanical wear of support and is used during stable operation of the FEL, when the operator does not modify the radiation wavelength.

After receiving the entire spectrum waveform, the application performs the following processing:

1. Calculation of the total power of the FEL radiation from the spectrum curve with application of calibration coefficients.
2. Calculation of the “average” wavelength of the radiation using the following formula:

$$\langle \lambda \rangle = \frac{\sum_i (A_i - A_{\min}) \times \lambda_i}{\sum_i (A_i - A_{\min})}, \quad (2)$$

where λ_i is the wavelength corresponding to the i th element in the measured waveform; A_i is the value of the i th element in the measured waveform; A_{\min} - minimum value of the radiation intensity in the waveform of the obtained spectrum.

3. Calculation of the width of the spectral line of the radiation using the following formula:

$$\Delta \lambda = \frac{d\lambda \times \sum_i (A_i - A_{\min})}{A_{\max}}, \quad (3)$$

where $d\lambda$ is the step in the wavelength between two measurements with the ADC during the support

movement; A_{\max} - the maximum value of the radiation intensity in the waveform of the obtained spectrum. Retrieval of the wavelength corresponding to the maximum intensity.

TRANSFER AND DISPLAY OF DATA OBTAINED

Since the default communication protocol in all the applications of the FEL control system is the EPICS Channel Access, it is also used for the transfer of the obtained and calculated spectrum data to other FEL control PCs and user station computers. The spectrum curve itself, total radiation power, average and maximum wavelengths, and spectral line width are represented as process variables both in the FEL control network and user station network (see Fig.4). The computer with the FEL radiation control application is connected to both these networks.

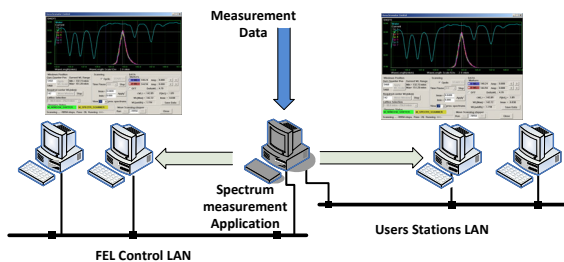


Figure 4: Connection of the control PCs and data transfer layout.

The application that controls the process of spectrum scanning is executed as a module inside the application that controls and monitors all other parameters of the FEL radiation system and described in detail in [3]. This application can run in two modes – client and server ones. The server-mode application runs on the computer connected to all the control and measurement hardware, while the client-mode application runs on other computers in the FEL control or user Local Area Networks. The client-mode application only displays data obtained by the server-mode application and transmitted by means of the Channel Access Server.

In case of the scanning spectrum application, this division into the client and server modes is also used. All operations regarding the scanning process control and interaction with the hardware (CAN ADC and stepper motor controller) are carried out in the server-mode application. After obtaining the new spectrum waveform and its processing, the Channel Access server updates all the corresponding PVs. The client-mode applications are only waiting for new values to appear, whereupon these values are displayed. Applications in both modes have the same user interface, therefore the FEL operators and users work with the same spectrum graph and window and are able to implement the same actions with the spectrum graph, its data and calculated values.

CONCLUSION

The system of real-time measurement of the FEL radiation spectrum has been successfully used for several years, during all time of the FEL operation, demonstrating high stability and reliability. The measurement data allows obtaining not only the radiation spectrum, but also the total power, average wavelength and spectral line width. With the EPICS Channel Access protocol, all the obtained data can be displayed on any computer both in the FEL control and user station local area networks.

REFERENCES

- [1] N.A. Vinokurov et al., Novosibirsk free electron laser facility: Two-orbit ERL with two FELs, Proceedings of FEL2009, TUOD01.
- [2] V.R. Kozak, Embedded device set for control systems. Implementation and applications, Proceedings of RuPAC 2006, Novosibirsk, Russia, THDO05.
- [3] V.V. Kubarev et al., Control and diagnostic system of Novosibirsk FEL radiation, Proceedings of FEL2007, Novosibirsk, Russia, MOPPH043.

EPICS BEAST ALARM SYSTEM HAPPILY PURRS AT ANKA SYNCHROTRON LIGHT SOURCE

I. Kriznar, Cosylab, Ljubljana

S. Marsching, Aquenos GmbH, Baden-Baden

E. Hertle, E. Huttel, W. Mexner, N. J. Smale, A.-S. Mueller, KIT, Eggenstein-Leopoldshafen

Abstract

The control system of the ANKA synchrotron radiation source at KIT (Karlsruhe Institute of Technology) is adopting new, and converting old, devices into an EPICS control system [1]. New GUI panels are developed in Control System Studio (CSS). EPICS alarming capabilities in connection with the BEAST alarm server tool-kit from the CSS bundle are used as an alarming solution. To accommodate ANKA future requirements as well as ANKA legacy solutions, we have decided to extend the basic functionality of BEAST with additional features in order to manage the alarming for different machine operation states. Since the database of alarm sources is been populated from scratch, we have been able to take a fresh approach in management and creation of alarm sources to build-up alarm trees. The new alarm system is being continuously used, tested and refined, and has been in the production environment since the end of 2013.

INTRODUCTION

ANKA is an electron synchrotron radiation light source located in Karlsruhe, Germany. The storage ring is generally operated at an energy of 2.5GeV with a typical beam current of 200 mA and a life time of 20 hours. The ANKA machine control system has been gradually moved towards EPICS based solutions. Since EPICS is relatively fresh at ANKA we tried to use the best that is available in the EPICS field.

Overview

An alarm system catches, transports, processes and visualizes error conditions (malfunctions) of hardware and software. An alarm system can report problems early, before serious consequences develop, and can provide data for later analysis. It propagates the alarm states from the producers (IOCs, drivers, services, applications) to the clients.

A complete alarm system includes the following components: an alarm source on IOC or services, an alarm aggregation service and a book-keeping service, an alarm archive, an alarm distribution service, a GUI viewer for new and archived alarms, a GUI configuration manager of the alarm service. All these building blocks are nicely covered by CSS alarm system [2], called “Best Ever Alarm System Toolkit” or BEAST [3].

Status Monitoring vs. Alarms

It is very important to distinguish between a status monitoring and an alarm event. Status monitoring system

and associated displays gives an overview of information about certain control system components or areas. Changes of status are potential sources of alarms if certain conditions are met. The Alarm view of a system is therefore a dynamic view of status changes, which might be dangerous to the operation.

A short example for illustration: If a server panel displays control system servers with little LED lights attached; Then, if a light is green, the server is running, if red, the server is down. This is status display. On the contrary when an alarm panel is blank, everything should be OK. When one of the servers goes down, the change of status triggers a notification, this is an alarm message, which appears in a table of active alarms until it is acknowledged by the operator. When an operator sees an alarm he/she can open a related status display for more information and consequently take appropriate action.

ALARM SOURCES

We can distinguish several kinds of EPICS records in relevance to alarming, thus several kinds of alarm sources.

Control point alarm sources are low level alarm sources. These are records, which are used to control or monitor devices and connected hardware. They generate alarm notifications directly during EPICS record processing and are connected to record intrinsic conditions: like value outside alarm limit or bus errors.

Context dependant alarm sources are the higher level alarms; they calculate alarm conditions from values obtained from several sources. For example, the machine operation status provides the context, which then defines if an alarm is raised or not for some condition.

Gateway alarms. These are software records which monitor alarm sources, which are not part of EPICS, and forwards their alarms as EPICS alarms.

Alarms which originate in an **interlock system** or any other safety systems, which are generally independent from the rest of control system.

The Context Dependent Alarms

Some machine or device status conditions present valid alarm source only under specific conditions. For example, a certain magnet (or its power supply) turned ON can present a reason for an alarm when the machine is in INJECTION mode, but the same status could be OK or irrelevant when the machine is in SR RAMPING mode.

Contextual alarms could be produced by implementing contextual logic directly as control point alarm source, such as device IOC, or as a separate server which intercepts and processes the alarm.

Device drivers at the IOC level need to be kept relatively simple as they might be very different in internal record set-up, so implementing contextual design on larger scales seems impractical.

However, this does not seem to be a problem if contextual alarm processing is implemented on a separate server, which listens to PVs and lets their alarms through if context of machine state or some other condition allows it. This way alarm triggering logic can be quite complex without complicating device drivers.

In addition the control system must provide all contextual relevant information of the machine in the form of PV channels. This means that at minimum the machine operation status is an EPICS enumeration PV.

ALARM OPERATION POLICY

In business alarming **quality comes before quantity!** An alarm panel full of alarm messages, in which no-one has explanation or knows the cause, especially while the machine seems to be operating without problems, contribute little to the operation. Such an alarm system is not functional. This means that simply adding all (primary) alarm sources or PVs into the alarming system does not provide a functional alarm system.

There are two main rules for a functional alarm system:

- System should provide only alarms to which operator **must react**. If alarm information is nice-to-know, but does not require reaction from operator, then this piece of information is not an alarm. Even if technically it is defined as such.
- Alarm system must not provide to operator more alarms that operator can handle. There are several strategies to prevent this: do not include too many alarms in the first place, integrate/group alarms together or provide ways to easily disable alarms which carry no relevance.

Bottom line is that if an operator is bombarded with alarms, he will just stop paying attention to the alarm system and we don't want that.

ANKA ALARM SYSTEM IMPLEMENTATION

As we were introducing the alarm system in the ANKA control room we took care that operators would treat them as a serious information source. We made sure that all the alarms added to the alarm system conformed to the mentioned "Operation policy". All alarms that were fired without obvious relevance to the operation were carefully examined and then fixed or removed.

Most low level alarms, coming from control point alarm sources and device drivers, do not comply with the

operation policy. In addition to EPICS naming convention we have added additional functional requirement in the form of device type convention. This convention prescribes that all devices should implement one additional record of `bi` with PV name `<device>:Status:ErrorSum`. The `Status:ErrorSum` record should be 0 when a device functions properly and have value raised to 1 and alarm state raised when anything goes wrong with the device or its operation is limited. This requirement reduces the number of PVs to be included per device and makes device behaviour more standard and easier to integrate into alarms, and other tools.

The higher level contextual alarm sources are those sources that contribute the most relevant alarms to the alarm system. This requires a strong intermediate alarm processing level of alarm sources: a conditioning level, which provides functionality between the BEAST alarm server and low level PVs.

This additional functionality was provided with the ANKA Alarm Conditioning Server. The alarm conditioning server hosts alarm dedicated records, which are triggered by processing events from other alarm sources or by monitoring vital signals of the control system, which are not necessarily integrated into EPICS. The BEAST alarm server connects to the alarm sources from the conditioning server as it would do to any other PV record. This way conditioned alarm PVs together with the PVs coming directly from low level alarm sources are building up the alarm hierarchy tree in a BEAST alarm configuration. By this no additional complexity is introduced into an already reliably working BEAST archiving, since it treats all alarms equally.

The ANKA alarm conditioning server is implemented on top of a Java CA server (CAS). It listens to other PVs and alarms. It filters alarms so particular alarms are generated or forwarded only when the machine is in an appropriate operation state. It also hosts alarm sources, which are generated by processing other control system parameters. The actual alarm processing is done by the configurable alarm processing units. Each of these alarm processing units provides one alarm dedicated PV into the BEAST alarm hierarchy tree and can be configured to listen to various parameters or PVs. The alarm processing units are loaded and configured from XML configuration files at the alarm conditioning server start-up. Additional scripts take care that BEAST alarm configuration is updated.

Table 1: An Incomplete List of Alarm Processors, Which Provide Various Alarm Sources at ANKA

Alarm Source Type	Applications
Host ping	Makes periodic network ping to all important servers or IP enabled devices in the control system.

System Process Watchdog	Runs on Windows or Linux computer, raises alarm when certain process appears in OS task list. Designed to intercept Java Errors on Windows servers.
Status Check	Monitors PVs with bitset value (such as mbbiDirect record value) and raises alarm if bits matches configured bit-masks for on or off states. Status PVs of all power supplies are checked this way.
State Watchdog	A PV value of this processor must be reset in regular intervals by some remote process. If this fails, then alarm is raised. Some crucial application are monitored this way, such as slow orbit correction and some archiving processes.
Summary Alarm	Listens to one or more PVs, sums their alarms, and forwards them further if machine operation state allows. It can also provide configurable alarm sums of a sub-tree in the BEAST alarm hierarchy three.
Value Diff Check	Check value difference between set-point and read-back of a power supply. All power supplies are included. This alarm check has two implementations in use: one as part of conditioning server and one as independent dedicated EPICS server.

In addition to the mentioned processing capability, all these alarm processing units (Table 1) have in addition the capability to filter their raised alarms by machine operation status, which is provided by a control system wide PV. This is again completely configurable within the main XML configuration file.

Because of the general nature of an alarm server it is also used for additional tasks such as a general EPICS application server. It hosts processing tasks for feedback loops, power supply ramping and cycling, and wiggler operation.

The alarm system at ANKA is a project still in progress. Further alarm sources are planned to be added into the alarming. As new device groups are integrated into EPICS they get also their own alarm sources. In addition we want to further add higher level alarm checks for control system health and operation problems.

ALARM CLIENTS

Besides the mentioned addition of alarm conditioning server, the rest of the Alarm solution is basic CSS BEAST installation. ANKA alarming panel is a standard CSS application with preconfigured perspective areas, as

shown in Figure 1. The ANKA Launcher opens this panel by a dedicated short-cut, which takes care that each time the alarm panel is opened it is organized in the same way.

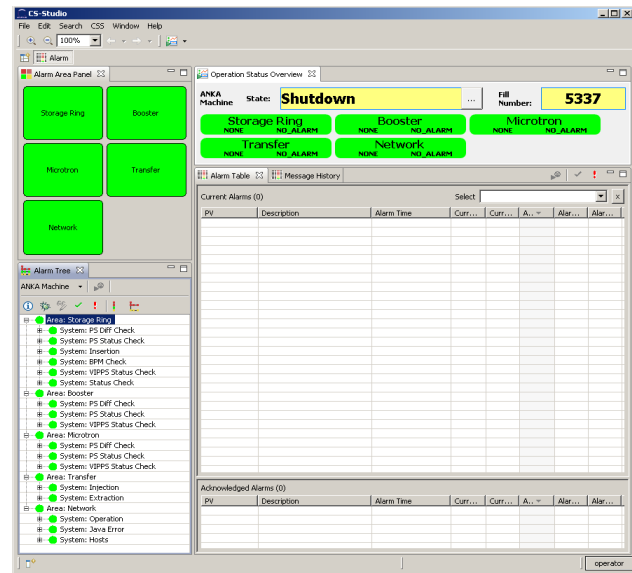


Figure 1: BEAST Alarm panel at ANKA as it is usually seen, without errors, even during a shut-down period, when operation related signals are in an undefined state and therefore their alarms are irrelevant and suppressed in context of shut-down machine status.

CONCLUSION

ANKA has introduced into operation a functioning and relevant alarm system. Operators have learned to trust information in the alarm panel and take alarms serious. This already proved very useful in day-to-day operation. The ANKA alarm system was carefully built with alarm sources that were giving information relevant for operation. The set of alarm sources might not be as wide as would be possible but the ones that are included are of high quality.

ACKNOWLEDGEMENT

Many thanks to wonderful community around CSS that provided solid solution, the BEAST, for delivering alarms to the operators.

REFERENCES

- [1] N.J. Smale et al., "The ANKA Control System: On a Path to the Future", MOPPC099, ICALEPCS'13.
- [2] K. Kasemir et al., "Control System Studio Guide - Alarm System", [Http://Cs-Studio.Sourceforge.Net/Docbook/Ch13.html](http://Cs-Studio.Sourceforge.Net/Docbook/Ch13.html)
- [3] K. Kasemir et al., "Control System Studio Guide", <http://cs-studio.sourceforge.net/docbook/>

IMPLEMENTATION OF THE DISTRIBUTED ALARM SYSTEM FOR THE PARTICLE ACCELERATOR FAIR USING AN ACTOR CONCURRENT PROGRAMMING MODEL AND THE CONCEPT OF AN AGENT.

D. Kumar, G. Gašperšič, M. Plesko, Cosylab, Ljubljana, Slovenia
R. Huhmann, S. Krepp, GSI, Darmstadt, Germany

Abstract

The Alarm System is a software system that enables operators to identify and locate conditions which indicate hardware and software components malfunctioning or nearby malfunctioning. The FAIR Alarm System is being constructed as a Slovenian in-kind contribution to the FAIR project. The purpose of this paper is to show how to simplify the development of a highly available distributed alarm system for the particle accelerator FAIR using a concurrent programming model based on actors and on the concept of an agent. The agents separate the distribution of the alarm status signals to the clients from the processing of the alarm signals. The logical communication between an alarm client and an agent is between an actor in the alarm client and an actor in the agent. These two remote actors exchange messages through Java MOM. The following will be addressed: the tree-like hierarchy of actors that are used for the fault tolerance communication between an agent and an alarm client; a custom message protocol used by the actors; the message system and corresponding technical implications; and details of software components that were developed using the Akka programming library.

INTRODUCTION

The FAIR Alarm System is composed of three major layers: a generation layer, a processing layer and a client layer. The connecting glue between the layers is the messaging system which allows the layers to communicate by passing messages into each other's queues and topics.

The Generation Layer

The alarm generators are the components that raise/lower alarm signals which are transported to the processing layer through a Java Message Oriented Middleware. The main purpose of the generators is to produce the alarm signals containing an alarm identification and state of the alarm that can be active or inactive. They are also responsible for handling the fast alarm oscillations. The alarm generators produce life-cycle messages notifying the processing layer about their health. The alarm generators must be registered with the processing layer before the alarm signals can be sent. This gives the processing layer a chance to prepare the environment for alarm generator monitoring and alarm

signal receiving. During the registration process the processing layer also checks that the alarm identifications are known to the system. If they are not known, the processing layer creates a default configuration for the unknown alarms.

The Processing Layer

The core of the alarm system is the alarm processor which is responsible for alarm signal processing and dispatching of the processed alarm signals to the client layer via an agent. The alarm processor also monitors the alarm sources. The alarm signal processing includes: matching the alarm signal with its configuration, updating the alarm state, alarm masking, and alarm archiving. The alarm processors are stateless and session-less, working in groups to share the load of the alarm processing. Processed alarm signals are not dispatched directly to the client layer but are sent to the agents which have active client sessions. The client layer accesses the alarm system only through an agent by opening a client session. All client requests are handled by the agents. The agent is responsible for handling alarm reduction, subscribing and filtering alarm state changes, acknowledgement of the alarms, sending filtered alarm state changes to the subscribed clients, and searching the alarm state and alarm archive.

The Client Layer

There are many types of alarm clients: the alarm monitoring viewer showing the state of the alarms, the alarm archive browser displaying the alarm history from a selected time range, the alarm configuration editor which issues CRUD operations on the alarm configuration. Common to all alarm clients is that they access the alarm system through an Alarm Client API. The alarm clients open many concurrent and independent sessions through which they issue requests to the alarm system and receive replies and alarm state changes. When a session is opened in the client layer, another session is also created on the agent. A hierarchy of actors [1] is created on both layers establishing a logical communication channel between a client session actor and an agent session actor. The physical communication is done through the actor hierarchy where individual actors take different roles: session management, session supervision, service worker, JMS message producer, and JMS message consumer.

The Technology

The alarm generator is written in Java SE 7 and C++. The native version of the generator works under Linux and Windows using a Boost portable C++ source library [2] and ZeroMQ [3] for a networking and concurrency framework. The underlying messaging system is ActiveMQ [4] in a “shared file system master slave” for a fail-over configuration. A custom ActiveMQ plugin was implemented bridging ZeroMQ and JMS messages. The alarm processor and the agent are written in Java SE 7 using the Spring framework [5], Bitronix standalone transaction manager for the distributed transactions [6], and JPA for the management of relational data. No application server is needed to run the alarm system.

The alarm processors work in a cluster using Hazelcast [7], an In-Memory Data Grid. The Alarm Client API is written in Java SE 7 and uses the Akka library [8] for the actor abstraction implementation in the client and processing layer. All layers of the alarm system use custom protocol messages encoded with Google Protocol Buffers [9].

ACTORS AND AGENTS

As already mentioned, there are two subsystems in the processing layer. The core task of the alarm signal processing and generator monitoring is assigned to one or more alarm processors running in a cluster. The task of managing the client layer is done by the agents. The window into the alarm system from the perspective of a client is a session in the Alarm Client API subsystem. While the session is being established on the client layer, another linking session is also opened in the processing layer on the agent. These two sessions communicate with each other through a logical communication channel exchanging regular and life-cycle messages. If the client session detects that the agent session is not responding, it will reregister with another agent and transfer its state to it. The applications using the Alarm Client API will not notice that the session was re-established on a different agent. The application using the Alarm Client API can open many sessions. These sessions are fully independent from each other, running concurrently with their own alarm subscriptions, their own event listeners and alarm reduction rules settings.

To ease the development of concurrent and fault-tolerant alarm clients and agents, we replaced the traditional model of shared state concurrency with the Actor Model, thus avoiding the pitfalls of controlling and manipulating the shared state with locks and threads.

In the Actor Model, all objects are modelled as independent, computational entities that only respond to the messages received. There is no shared state between actors. Actors change their state only when they receive a stimulus in the form of a message [10].

Error detection is an essential component of fault tolerance. That is, if you know an error has occurred, you might be able to tolerate it by replacing the offending

component, using an alternative means of computation, or raising an exception [11].

Each actor that performs a task is associated with a supervisor actor which monitors its actors for faults. If an error occurs in the supervised actor, the supervisor will initiate some error recovery procedure. This error recovery can restart or resume the subordinate actor, terminate it, or escalate the failure to its own supervisor which has the exact same options regarding the error handling. The supervisors and worker actors thus form a supervisor hierarchy. In our case, a session in the client layer is implemented as an actor that belongs to a hierarchy of supervisors and supporting actors that enable the session actor to communicate with its counterpart session actor on the processing layer. The same holds true for the session actor on the processing layer. It too belongs to a supervisor hierarchy with the supporting actors that enable communications with the client layer and service actors that execute requests on behalf of the alarm clients.

Figure 1 shows the outline of the alarm supervisor hierarchy in the client and processing layer and, more importantly, the roles that actors play to establish different types of communication channels between the client and the agent.

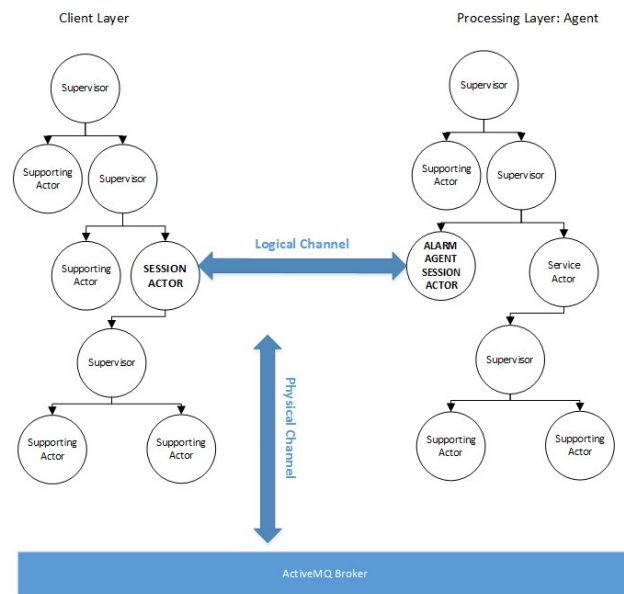


Figure 1: The Alarm Supervisor Hierarchy.

THE AGENT SUPERVISOR HIERARHY

The agent supervisor hierarchy is shown in Fig. 2 and has a root actor, AlarmAgentActor, which is responsible for bootstrapping the hierarchy. In the tree hierarchy we have three main branches of actors. The branch holding the AlarmAgentConsumerSupervisor and its supporting actors is responsible for receiving the register and unregister client protocol messages and alarm state changes from the alarm processors.

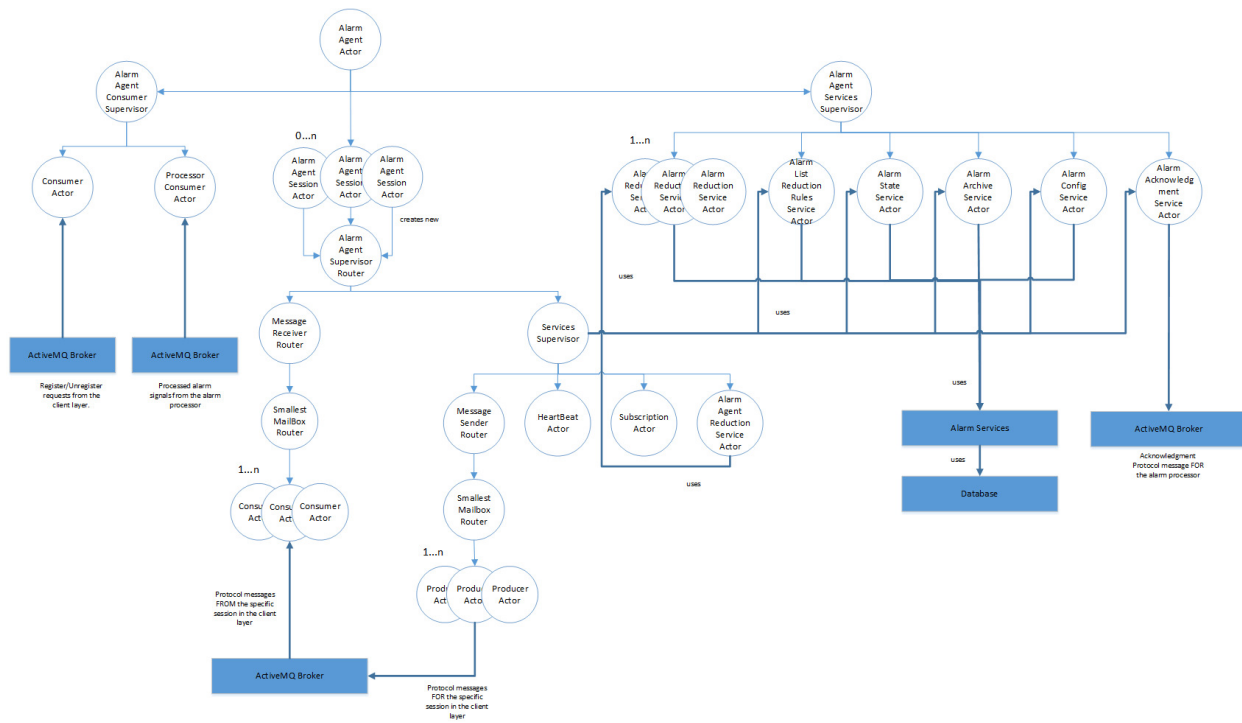


Figure 2: The Agent Supervisor Hierarchy.

The register protocol message will create a new agent session actor in the middle branch of the tree hierarchy. The newly created AlarmAgentSessionActor bootstraps its own sub-hierarchy of supporting actors. The session actor pushes all of its protocol messages, self-generated or received through the AlarmAgentConsumerSupervisor, to the AlarmAgentSupervisorRouter. This actor then routes the messages to the ServiceSupervisor where the client requests are handled by its task actors. The ConsumerActor in the session actor sub-hierarchy is responsible for receiving the client protocol messages for that session. These messages are routed to the session service supervisor by the AlarmAgentSupervisorRouter. The session service sub-hierarchy has one supervisor (ServiceSupervisor) and many task actors. The task actor HeartBeatActor is responsible for the session life-cycle management. The SubscriptionActor will filter the alarm state changes, the AlarmAgentReductionServiceActor reduces the alarms. All messages that are produced by the task actors are sent to the client layer through the ProducerActors. Lastly, we have the service layer of the agent represented by the AlarmAgentServiceSupervisor where we have all the services that are used by the session actors.

CONCLUSION

Implementing a distributed and a fault tolerant system is never an easy task. To simplify the development of a distributed and a fault tolerant alarm client layer and processing layer we avoided using the shared state concurrency model and went with the actor model. The system was made fault tolerant by organizing the actors

into a supervisor hierarchy containing the actor tasks and the supervisor actors responsible for fault monitoring and error recovery [11]. The alarm supervisor hierarchies were also built and modelled in the simulation package AnyLogic 6 [12] where we used agent based modelling. The results we obtained from the simulation were used to prove that the non-functional requirements of the FAIR Alarm System were satisfied.

REFERENCES

- [1] Wikipedia website: http://en.wikipedia.org/wiki/Actor_model
- [2] Boost website: <http://www.boost.org>
- [3] ZeroMQ website: <http://zeromq.org>
- [4] ActiveMQ website: <http://activemq.apache.org>
- [5] Spring framework website: <http://projects.spring.io/spring-framework>
- [6] Bitronix website: <http://docs.codehaus.org/display/BTM/Home>
- [7] Hazelcast website: <http://hazelcast.org>
- [8] Akka website: <http://akka.io>
- [9] Google Protocol Buffers website: <https://developers.google.com/protocol-buffers>
- [10] M. K. Gupta, Akka Essentials, ISBN: 978-1-84951-828-4, Packt Publishing, Birmingham, UK (2012), pp. 11.
- [11] J. Armstrong, "Making reliable distributed systems in the presence of software errors", Doctoral Dissertation, The Royal Institute of Technology, 2003, pp. 115-127.
- [12] AnyLogic website: <http://www.anylogic.com>

VACUUM PUMPING GROUP CONTROLS BASED ON PLC

S. Blanchard, F. Antoniotti, F. Bellorini, JP. Boivin, J. Gama,
P. Gomes, H. Pereira, G. Pigny, B. Rio, H. Vestergard
CERN, Geneva, Switzerland
L. Kopylov, S. Merker, M. Mikheev
IHEP, Protvino, Russia

Abstract

In particle accelerators, high vacuum is needed in the beam pipes and for thermal isolation of cryogenic equipment. The first element in the chain of vacuum production is the pumping group. It is composed by: primary pump, turbo-molecular pump, valves, gauges, process and interlocks devices. At CERN accelerators, the pumping groups may be found in several hardware configurations, depending on the environment and on the vacuum system used; the control is always based on Programmable Logical Controllers (PLC) communicating with the field equipment over a field bus; pumping groups are controlled by the same flexible and portable software. They are remotely accessed through a Supervisory Control and Data Acquisition (SCADA) application and can be locally controlled by a portable touch panel. More than 250 pumping groups are permanently installed in the Large Hadron Collider, Linacs and North Area Experiments.

INTRODUCTION

This paper describes the new control software for the turbo molecular pumping group of type “VPG_6A”. It is based on operational modes and phase sequencers. It includes sequential process for pumping, leak detection and venting modes. Optional features for automatic restart and automatic venting are also available.

The mechanical layout of the pumping group is illustrated by the Fig. 1.

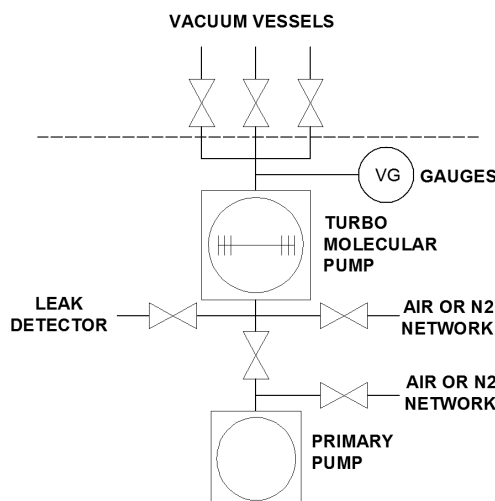


Figure 1: Pumping Group Layout.

CERN Accelerators have more than 120 km of beam and thermal isolation vacuum vessels. Turbo molecular vacuum pumping groups are used for rough pumping, for leak detection and to maintain the vessels under high vacuum. The primary pump performs initial pumping from atmospheric pressure to rough vacuum (between 10 and 0.1 Pascal) then the turbo molecular pump achieves high vacuum (between 1.10^{-4} and 1.10^{-7} Pascal).

HARDWARE

The new software is compatible with any hardware control crates based on Siemens® PLC S7-300 or S7-400 series with a minimum of 64Kb working memory. That includes the recently designed prototype shown in Fig. 2 but also with the 15 years old crates.

Pumping group devices are controlled using direct PLC Input/Output or Profibus® fieldbus.

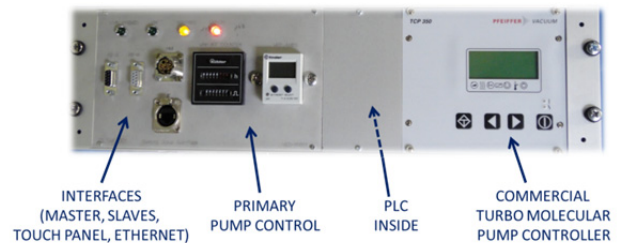


Figure 2: Prototype Pumping Group Controller Crate.

The controller crate commands the pumps and valves; and gets the feedback status from them: opened / closed from the valves, and a minimum of 3 status bits from the pumps.

For the primary pump: “ON” status is the feedback of the pump power supply actuator; “NOMINAL SPEED” status is given by the motor current monitor and “ERROR” status is the feedback of the circuit breaker.

For the turbo molecular pump: “ON” status is the pump rotation detected feedback; “NOMINAL SPEED” status is given when the pump rotor reaches nominal speed threshold (in the most of the cases, 80% of the maximum speed) and “ERROR” status occurs when load is too high, when no rotation is detected, or when the rotor acceleration time is too long.

The standard hardware architecture, as shown in Fig. 3, is composed of a local crate, a control crate, a touch panel (local Human Machine Interface), a gauge controller, a PLC master and a SCADA server.

A Profibus® network connects the touch panel, the control crate, the gauge controller and the PLC master.

The touch panel can be connected either to the local crate or to the control crate. It is easily removable, not to remain in radiation area and to limit the total needed number of touch panels.

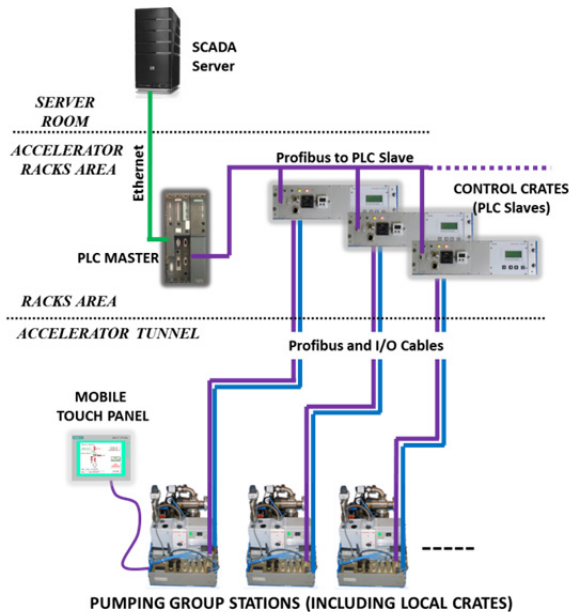


Figure 3: Typical Hardware Architecture.

PLC SOFTWARE

Vacuum devices are defined in a global Vacuum Database (VACDB), and classified by “families”, “types” and “sub-types”. The family classifies actual physical devices (valves, mechanical pumps, active gauges, passive gauges...) and virtual devices (pumping group process, software interlock...).

The type defines the control behaviour, as programmed by a common PLC function (Function Block, FB). The sub-type is used to manage minor differences of behaviour and to set the correct SCADA widget.

The PLC software comprises a set of functions which are part of the generic Vacuum Controls Framework [1], [2]. They are written with SIEMENS-SCL Language, a textual high-level language that follows the standard IEC 61131-3 ST (Structured Text) definition. This language allows to program complex algorithms, arithmetic functions and data processing tasks.

A PLC software project is composed of organisation blocks (OB), functions blocks (FB, FC), and instance or generic data blocks (DB).

Function Block (FB): Programs routine working with instance memories; one routine (FB) per device type and one instance memory block (DB) per device. The OBs, FCs, FBs, and generic DBs are copied from the baseline SVN repository folder, provided by the CERN central repository service. Device instance DBs are automatically generated by the vacuum database editor and compiled within the PLC project.

Process

Valves, pumps and gauges have a “MANUAL” and an “AUTOMATIC” control mode. In “MANUAL” mode, the device is directly driven by an operator order, in “AUTOMATIC” control mode the device is driven by a process running in the PLC.

Two kinds of software interlock override the orders given in any mode, either “MANUAL” or “AUTOMATIC”. The “start-interlock” disables any new actions on the device and; the “full-interlock” switches off the device.

When in AUTOMATIC control mode, the behaviour of the process managing the pumping group depends on the operational mode. There are 6 operational modes: 1 pumping mode, 3 service modes, and 2 venting (stop) modes.

Transitions between modes can be requested either by the operator or by the process itself, according to Fig. 4.

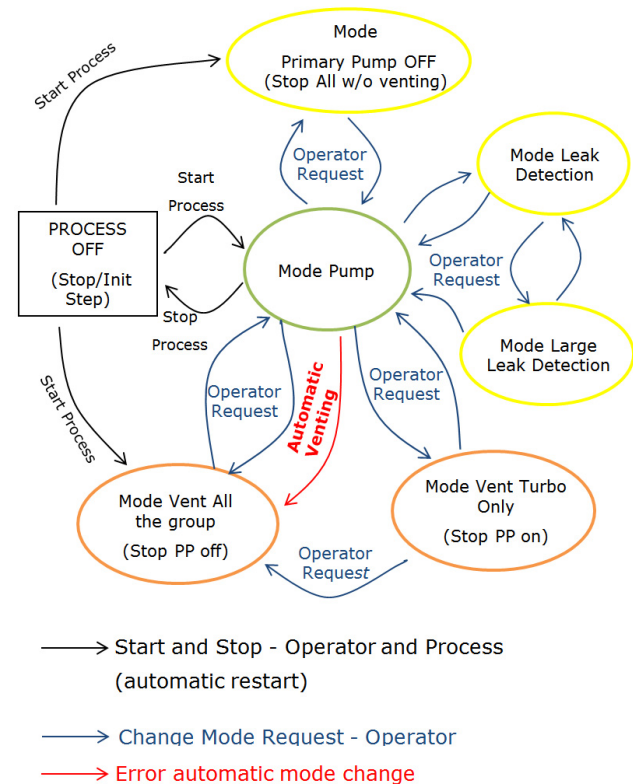


Figure 4: Operational modes and available transitions.

The phase sequencer follows a series of steps, through transition conditions and time dependencies; it sends “AUTOMATIC” orders to the devices within the pumping group, in order to reach a final state according to the operational mode.

The phase sequencer software is developed with the graphical language SIEMENS-GRAPH.

The pumping group process provides automatic venting (to avoid oil contamination from the primary pump), automatic restart and automatic re-opening of the valves to the vacuum vessel, in case of power cut.

Alert information

Troubles and unexpected events are classified to:

- Minor: warning status is set and the warning code is updated; the process continues to run.
- Major: error status is set and an error code is updated; the pumping group is automatically vented or directly shifted to the Initial step (“safety state”), according to user specifications.

SCADA APPLICATION

The pumping group control is included in the Vacuum SCADA vacuum application [3].

The details panel, as shown in Fig. 5, provides a state monitoring and a remote control to operators.

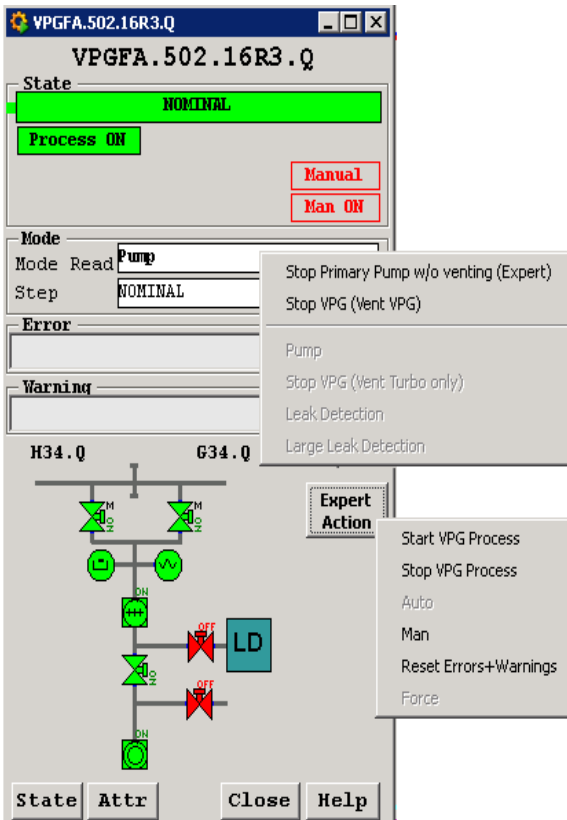


Figure 5: Pumping Group Details Panel.

The state history panel, as shown in Fig. 6, allows comparing archived values of status, operational mode and sequencer step of pumping group; it is very useful for a post-mortem diagnostic.

The pumping group pressure gauges history, as shown in Fig. 7, is an additional tool to check pumps efficiency, analyze pumping speed and detect leaks.

Operators may configure in SCADA e-mail/SMS notifications to be notified in case of pumping group error, stop or unexpected pressure increase.

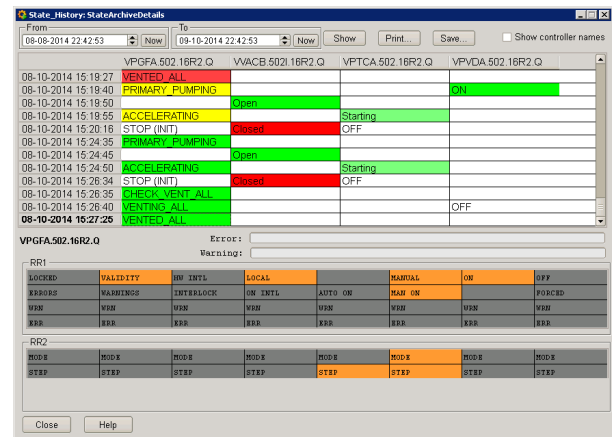


Figure 6: Pumping Group State History.

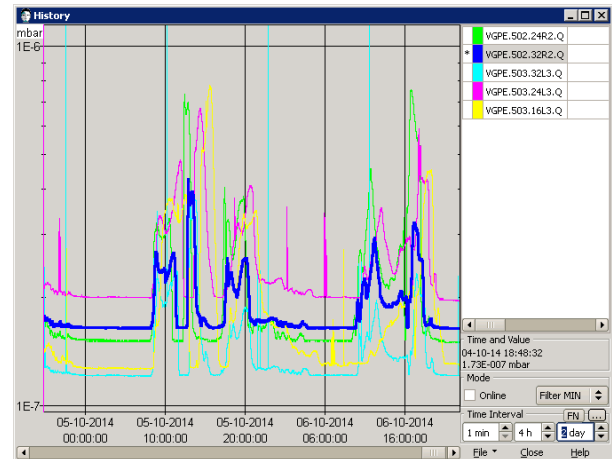


Figure 7: Pressure History.

CONCLUSION

The vacuum pumping group control offers a robust solution to drive a large variety of different turbo molecular pumping groups. It has been successfully deployed to more than 250 pumping groups and is foreseen to be installed in future CERN vacuum systems as a standard solution. The identical software for all turbo molecular pumping groups has reduced the maintenance and facilitated intervention. It has improved flexibility, remote control, diagnostics and data logging.

REFERENCES

- [1] P. Gomes et al., “The Control System of CERN Accelerators Vacuum [Current Status & Recent Improvements]”, ICALPECS11, Grenoble, 2011; MOPMS016.
- [2] P. Gomes et al., “The Control System of CERN Accelerators Vacuum [LS1 Activities And New Developments]”, ICALPECS13, San Francisco, 2013; MOPPC027
- [3] F. Antoniotti et al., “Developments on the SCADA of CERN Accelerators Vacuum”, ICALPECS13, San Francisco, 2013, MOPPC030.

DIAGNOSTICS TEST STAND SETUP AT PSI AND ITS CONTROLS IN LIGHT OF THE FUTURE SwissFEL

P. Chevtsov[#], R. Ischebeck, PSI, 5232 Villigen, Switzerland

Abstract

In order to provide high quality electron beams, the future SwissFEL machine needs very precise beam diagnostics tools. At the Paul Scherrer Institute (PSI), the development of such tools required setting up a dedicated automated test stand, which could significantly simplify the development process. The test stand is equipped with not only major SwissFEL beam diagnostics elements (cameras, beam loss monitors, wire scanners, etc.) but also their controls and data processing hardware and software. The paper describes diagnostics test stand controls software components designed in view of the future SwissFEL operational requirements.

INTRODUCTION

The future SwissFEL facility [1] at the Paul Scherrer Institute (PSI) will produce superb photon beams for a variety of cutting-edge user experiments. The project aims to use the lowest electron beam energy compatible with 1 Å facility operation. In these conditions, to ensure successful facility lasing, the machine must be equipped with advanced diagnostics and controls tools continuously monitoring electron beam generation, transport, and compression. The SwissFEL Injector Test Facility (SITF), which has been a major platform for designing and testing such tools since for the last few years, is going to be put out of operations by the end of 2014. After that, there will be no similar platform available on site for the SwissFEL machine component prototyping and testing. In order to supply a satisfactory workbench for electron beam diagnostics and control developments, a dedicated automated Diagnostics and Controls Test Stand (DCTS) was set up in the SwissFEL diagnostics laboratory located in the SITF building. There are several advantages of this location. For instance, being in the area of a well-established SITF control system infrastructure, any required DCTS computer network is easily configurable and supportable. Another important advantage is the direct access to modern laboratory tools for efficient equipment checking and tuning.

To ensure the reproducibility of test results in the real machine environment, diagnostics and controls projects on the stand are required to organize as close to their SwissFEL operational conditions as possible. For example, wire scanner moving components must be kept in vacuum, motor steering cables have to be about 20-30 meters long, etc.

All DCTS control system software developments must be compatible with EPICS [2], which is the principal controls tool at PSI.

[#]pavel.chevtsov@psi.ch

DCTS DEVELOPMENTS

From its very beginning, the DCTS project was divided into several stages based on the SwissFEL type control system hardware and software availability for beam diagnostics applications.

On stage one, which was the initial step, the basic test stand control infrastructure was set up and the development of the test procedures for all tasks implemented on the test stand began.

Basic DCTS Control Infrastructure

The main DCTS control network is a general purpose 1 Gbit class C subnet. The operator console is a Linux PC furnished with a high resolution monitor. A standard SwissFEL electronics rack houses the DCTS control equipment, which includes one motion control unit and several VME64x crates.

One VME crate, which is a primary DCTS software development crate, is equipped with a new PSI control computer (IOC), IFC_1210, designed in collaboration with IOxOS Technologies SA, Switzerland [2]. The IFC_1210 is a highly configurable FPGA platform associated with XMC, PMC, and FMC mezzanine slots for custom expansions, a powerful dual core PowerPC, and a real time OS. It is built around a high-performance switched PCI Express GEN2 architecture. The VME support firmware provides a complete VME master/slave interface. Compared to the SITF workhorse MVME-5100 single board CPU, this new IOC type offers much more computing power, a higher data throughput, and very flexible I/O features available for software developers. However, to make IFC_1210 fully operational, a lot of work must be done to design and implement EPICS device/driver support software for a variety of controls hardware components that can be handled by this IOC. It was obvious to begin with the easiest part of this work, which is porting existing PSI VxWorks-specific VME memory access software to the new IOC platform. As a result, the most critical VME drivers, including the General Purpose Memory Mapping (GPMM), became available for IFC_1210 by the time of the initial DCTS setup. This has allowed one to relatively easily integrate a variety of VME control cards into the test stand controls environment and to concentrate major software development efforts on supporting fast, not less than 250 Mega-Samples-Per-Second (MSPS), 16 bit ADC, high speed DAC, and digital IO FMC modules available from IOxOS Technologies SA.

The test stand timing system is realized in the second VME crate. It will provide control device triggering mechanisms and synchronous transmissions of DCTS

The work on the DCTS project is organized around a dedicated TWiki page, a popular project management tool JIRA, and a PSI Electronic Logbook (ELOG) [6]. This allows one to efficiently concentrate all available resources on the most important tasks for any particular time frame.

Users employ a set of instruments to provide the required operational conditions and to monitor the state of the project. Instruments are supplied with their control interface equipment, allowing one to operate these instruments efficiently, including remotely. For example,

Figure 1: DCTS user application setup overview.

Some data preparation work for DCTS user applications is done directly on the IOC by a dedicated software module. For example, bunch ID tagged data are collected in allocated EPICS waveform records. Such a software module and the IOC software taking care of the control system application interface form a user application control support package.

Based on the above definition, DCTS user application setup test procedures are divided into the next three categories: user instrument control support, user application control support, and user application test procedures.

The user instrument control support test procedures must be developed by the control system specialists. In particular, such procedures have to include the detailed instructions of how to

- set up and test the user control interface equipment,
- connect this equipment to the control system,
- run the instrument control software and make sure that it communicates with the interface equipment properly,
- troubleshoot any possible control system hardware and user instrument problems.

As a rule, this is done by providing detailed instrument control setup/test/troubleshoot documentation and related project operational tools (e.g. graphical user interface panels).

The user application control support test procedures have to be defined by both control system specialists and users. The goal is to make sure that any required data preparation for user applications is done correctly.

Finally, the user application test procedures must be developed by control system users. The procedures have to demonstrate that applications properly control user instrument operations and their data processing software works correctly, which requires the design and use of instrument signal sources with well-known properties, such as a round laser beam as the camera image source.

One of the main goals of the DCTS is to develop all user project specific test procedures based on the experience gained on the test stand.

Wire Scanner Application Setup

After setting up its basic control infrastructure, the DCTS project advanced to its stage two, which was the SwissFEL type wire scanner (WS) application setup.

The basic idea of a WS is simple. Thin (tungsten, carbon, etc.) wires stretched between two prongs of a metallic fork move through the beam. Secondary particles produced by the beam intercepting a wire are registered by a beam loss monitor (BLM) detector, which is usually a combination of a scintillator and a photo multiplier tube (PMT). The beam profile is completely defined by the level of the BLM signal as a function of the corresponding position of the wire.

The SwissFEL WS control interface equipment consists of a stepper motor with its position encoder and an in-house developed PMT control box. The stepper motor moves the WS fork. The output voltage of the PMT control box is proportional to the PMT signal level. All WS data must be bunch ID tagged.

The associated DCTS WS control hardware includes a DeltaTau stepper motor/encoder controller and an ADC control module. The electron beam is simulated by a laser beam.

In its initial configuration, the WS was set up on the test stand without bunch ID tagging means. Nevertheless, its operations contributed to a basis for WS project test procedures.

In particular, the WS application test procedures include the next steps to be implemented:

- move the WS fork through the laser beam,
- detect scattered photons detected by the loss monitor PMT and corresponding wire coordinates,
- calculate the laser beam size and compare this result with the diameter of the used laser beam.

Nearest DCTS Plans

A series of electron beam diagnostics and control projects is in the list to work on at the test stand in the next few months. The WS data bunch ID tagging has to be implemented. The SwissFEL camera server and camera image saving system must be configured to operate at 100 Hz. A productive IOxOS fast ADC control software development environment has to be created. Besides, as always, all this must be accompanied by the work on the proper project test procedures.

CONCLUSIONS

The DCTS project gives SwissFEL control system users and specialists a unique opportunity to efficiently work on their projects during the SwissFEL construction. A high flexibility of the DCTS framework allows one not only to create and test project components but also to simplify the development of powerful tools, which can be used to make sure that these components are fully operational: instrument and control hardware test signal sources, project test procedures, project setup and operation plans, etc.

ACKNOWLEDGEMENTS

The authors are grateful to B. Kalantari, P. Pollet, R. Kapeller, R. Krempaska, P. Valitutti and G. L. Orlandi for their valuable contributions to the DCTS setup.

REFERENCES

- [1] "SwissFEL Conceptual Design Report", PSI report 10-04 (2012).
- [2] E. Norum et al., "EPICS: Recent Developments and Future Perspectives", ICALEPCS'2003, Proceedings, p. 278-281, Gyeongju, Korea, 2003.
- [3] B. Kalantari et al., "Beam Synchronous Data Acquisition for SwissFEL Test Injector", ICALEPCS'2011, Proceedings, p. 202-204, Grenoble, France, 2011.
- [4] B. Keil et al., "The European XFEL Beam Position Monitor System", IPAC'10, Proceedings, p. 1125-1127, Kyoto, Japan, 2010.
- [5] P. Chevtsov et al., "Current Status and Perspectives of the SwissFEL Injector Test Facility Control System", ICALEPCS'2013, Proceedings, p. 378-380, San Francisco, CA, USA, 2013.
- [6] "The ELOG Home Page", <http://midas.psi.ch/elog>

MAGNET MEASUREMENT SYSTEM UPGRADE AT PSI

P. Chevtsov[#], V. Vrankovic, PSI, 5232 Villigen, Switzerland

Abstract

The magnet measurement system at the Paul Scherrer Institute (PSI) was significantly upgraded in the last few years. At the moment, it consists of automated Hall probe, moving wire, and vibrating wire setups, which form a very efficient magnet measurement facility. The paper concentrates on the automation hardware and software implementation, which has made it possible not only to significantly increase the performance of the magnet measurement facility at PSI, but also to simplify magnet measurement data handling and processing.

INTRODUCTION

The majority of magnets for PSI accelerators and experiments are designed in house and produced by industry. Upon delivery, these magnets are systematically measured to assure that all field quality specifications are met. The measurements are done with the use of the PSI magnet measurement system the technical and operational capabilities of which have significantly advanced in the last months. As a result, it is a modern, automated, user friendly system, which consists of high precision measurement setups based on Hall probe, moving wire, and vibrating wire techniques. The setups are arranged in separate rooms of the PSI magnet measurement laboratory, which are equipped with all necessary measurement tools, control computers, and operational consoles. The magnet measurement system is integrated into the PSI controls, which is based on EPICS [1]. This fact combined with the cutting-edge data acquisition and control software developed for magnet measurement applications at PSI makes the system easy to run not only from local operational consoles but also remotely, eventually from any PC connected to the PSI network.

HALL PROBE SETUP

Basic magnetic field mapping is performed using Hall probes. Fast automated Hall probe measurements are provided by a computerized Magnet Measurement Machine (MMM) created at PSI.

The MMM is a very precise positioning device sliding on compressed air pads over a flat, carefully machined granite block. Hall probes are attached to a titanium arm with carbon fibre extensions that can move in three translation directions (X, Y, Z) and rotate in the horizontal plane (a yaw angle) and around the arm (a roll angle). Each move is performed by a dedicated stepping motor. Positions are determined by Inductosyn® linear encoders with one half-micron accuracy. We note that the MMM measurements are performed in any translation direction while two rotations are used only for the proper probe positioning. Therefore, a measured field map corresponds to a line, a plane, or a volume in a Cartesian coordinate system.

[#]pavel.chevtsov@psi.ch

The Hall probe setup is installed in a temperature controlled ($\pm 0.1^\circ \text{C}$) room, which is important for accuracy of this kind of measurements.

Hall probe potentials are recorded by an Agilent 3458A digital multimeter (DMM device). Periodic probe calibrations are done with the use of a Nuclear Magnetic Resonance based Metrolab PT2025 teslameter (NMR device). Both devices can be remotely controlled via the GPIB bus. Magnet current values are set by PSI digital power supply controllers.

The main Hall probe setup control hardware is implemented in a VME-64x standard. The control computer (IOC) is a MVME-5100 single board CPU running the VxWorks real time kernel. The IOC also runs the EPICS database and control software (MMM server), which handle all Hall probe setup components and provide the information about the state of these components and all measurement data. The stepping motors and their encoders, which are based on Inductosyn® sensors, are interfaced via a MAXv-8000 card and its transition module. In-house developed Industry Pack (IP) control modules PSC-IP2 sitting on a Hytec 8002 carrier board provide the access to magnet digital power supply controllers. A variety of digital (DIO) signals are required for MMM operations. They are handled by Hytec 8505 IP modules. DIO signals are used, for example, to trigger DMM devices and to support a special manual mode (as opposed to its normal mode that is automatic), in which the MMM is positioned by the magnet measurement personnel pressing buttons on a remote control unit that looks similar to a conventional TV remote control. We note that the manual mode is especially valuable for initial MMM positioning with respect to any magnet to measure.

MMM measurements are performed in a “continuous scan on the fly” mode, which means that the machine doesn’t stop to make a particular measurement. This allows one to finish a complex field mapping process for each magnet in a relatively short time frame, which is typically one day or less.

The MMM control is done with the use of a specially designed graphical user interface (GUI) tool, which is called mmmgui. The tool is based on a well-known Qt framework [2], which automatically makes it computer platform independent. At PSI, the tool can run on any 32 or 64 bit Scientific Linux PC console. The mmmgui control software (MMM client) is multi-threaded. Each thread deals with a particular MMM operational mode, which is supported by a dedicated mmmgui panel implemented as a standard Qt tab. Threads communicate to each other over a shared memory synchronized with the MMM EPICS database.

The most frequently used MMM operation, which is driving a Hall probe along a measurement axis, is supported by the Mes(X,Y,Z) mmmgui panel. The operator has to define the start and end positions of the measurement, the motor acceleration and maximum speed, data acquisition time, the number of measurement points, and the magnet power supply current value. Based on this information, the mmmgui control software performs a series of calculations and system parameter settings. In particular, assuming that the measurement points are equidistant, their coordinates are defined and immediately become a part of the EPICS database. This makes the MMM server software ready to trigger the DMM device exactly when the probe reaches such measurement points. Each time the DMM device is triggered, the actual probe potential is written into its internal memory buffer. When the motion is completed, all recorded data are transferred by the mmmgui software to the computer disk. Simultaneously, the measured probe potentials are written into the EPICS database, which already contains time stamps and probe coordinates corresponding to those potentials. The time stamps are obtained directly from the VxWorks operating system and the probe coordinates are DMA transferred from the VME memory associated with motor encoder data at the moments when the trigger signals are generated. The MMM server software assures all necessary real time constraints on the whole data acquisition process. We note that normally the same line is measured again in the reverse direction before moving MMM to another line. Measuring in both directions helps to cancel positional errors and any voltages induced in the probe connections moving in magnetic field gradients.

Another mmmgui panel, which is called Script, supports user programs written in a simple scripting language Lua [3], which is popular in computer games. In this way, field mapping procedures developed by magnet experts and implemented as Lua scripts fully automate the magnet measurement process at PSI. Operator's interventions are required only in case of possible MMM component failures, which is monitored and reported by the control software.

Because of its operational nature, the mmmgui is implemented as a single-user application. When the mmmgui starts, it tries to connect to the MMM server. If the server is already busy with another MMM client, the connection is refused and the mmmgui exits displaying a corresponding error message. Otherwise, the requested connection is established, which gives a user an exclusive access to all MMM resources and assures that the measurement process could never be influenced by anybody else.

VIBRATING WIRE METHOD SUPPORT

A stretched wire excited by an alternating current (AC) starts oscillating in the static magnetic field. The AC frequencies corresponding to natural wire resonances cause particularly large vibrations, which makes such a system very sensitive to the existence of the magnetic

field along the wire. Essentially, when the wire stretched in a multipole magnet stops vibrating, the effective magnetic axis and the wire are aligned. So, to locate the magnet axis, the wire should move until its oscillations vanish. This idea is the fundamental basis of a single stretched vibrating wire method [4], which is the most accurate technique to define the magnetic axes of multipole magnets.

A vibrating wire measurement test stand at PSI is arranged in a room, which is not air-conditioned in order to minimize the air flow. Electronics is kept outside of this room to make sure that temperature changes during the measurements are low and slow. Main monitor/control components are two pairs of linear motorized stages, a vibration detector, and temperature sensors. The linear motorized stages Newport M-ILS150CCL move in horizontal and vertical directions. One stage pair is static on a measurement table. The other pair can be relocated, which allows for measurements with different wire lengths. The magnets are placed on the table at the distance of a quarter wire length next to the static stage pair. At the other wire end, the vibration detector is positioned with the equal distance next to the movable stage pair. The reference point positions on the wire supports and the magnets are found with a FaroArm Quantum device. The position accuracy is better than 10 μm . The detection of wire vibrations is done by a novel PSI detector consisting of four pick-up coils, which form two orthogonally positioned pairs allowing one to detect the complete wire vibrations in space. The measurement signal from the pick-up coil pair contains the information about the wire position relative to the center of the coils and about the wire vibration. At that, the position of the wire in the detector doesn't influence the vibration reading and the vibration-induced voltage is virtually independent of the wire current frequency.

The core instrument of the measurement system is a digital lock-in amplifier HF2LI produced by Zurich Instruments (ZI) [5]. Two lock-in demodulators are used for the vibration detection and two for the wire position detection. One internal oscillator generates a constant voltage output powering the wire and serving as a reference signal.

At the time being, magnet and air temperatures in the room are monitored by Sensirion SHT75 sensors connected to the in-house developed embedded temperature and humidity measurement controller (ETHMC), which is fully integrated in the EPICS environment. This solution provides a good (0.01° C) temperature resolution. It is easy to configure and run. Unfortunately, the ETHMC loses the network connection (which is the only interface to the PSI controls) from time to time and must be rebooted to fix it. In order to improve the reliability of the temperature measurement system, we are planning to switch to WAGO [6] commercial controllers in the next few months.

The linear stages are individually handled with Newport SMC100 motion controllers. Although each stage has its

own controller, the motions of two horizontal and two vertical stages are made to occur simultaneously.

The vibrating wire measurement control software runs on two IOCs. One of them is a Linux box communicating with Newport SMC100 stage controllers over a standard serial (RS232) port and with the ETHMC via the computer network. The software is built on top of a standard EPICS Stream Device support package, which makes it easy to handle any device configuration and monitor its parameters on-line. The second IOC is a Windows PC talking to the ZI lock-in amplifier over a local USB port. The control software monitors the amplifier state with the use of the API library provided by Zurich Instruments. It also handles the EPICS records associated with the amplifier settings and status. We note that EPICS doesn't provide a standard solution for handling USB devices, and this software is a good example of how to integrate such a device into the control system. In particular, only event driven API functions are used by the amplifier EPICS driver/device support modules, which notably increases the overall system efficiency.

Being a part of the PSI EPICS controls, all vibrating wire setup parameters are available for on-line monitoring and control functions on any computer connected to a local network. The main application controlling the whole measurement process is written in Python, which is supported by EPICS. The application significantly simplifies the work on the measurement setup tuning, tests, and operations.

MOVING WIRE SETUP

A single stretched moving wire method is suitable for harmonics measurements in multipole magnets. The idea of this technique is relatively simple: move a stretched wire along a cylindrical surface in the magnet aperture and measure the magnetic flux change as a function of the rotation angle.

A moving wire setup created at PSI is based on a high performance multi-axes Newport XPS motion controller/driver, which implements advanced trajectory and synchronization features to precisely control complex motion sequences. A standard EPICS XPS support software package is a part of a special moving wire measurement control tool developed at PSI. The tool consists of a Moving Measurement Control (MMC) application, which runs on a Linux PC, and a set of MEDM GUI panels that are very easy to use for handling measurements. The MMC application communicates with the XPS controller over the computer network. Two pairs of linear motorized stages connected to the XPS controller are configured (as XY groups) to synchronously move both ends of a stretched wire along a specified line or arc, which makes it easy to perform any required plane or cylindrical motion. The MMC application must be provided with a wire trajectory definition file containing a set of reference points through which the system has to move the wire and a number of trajectory points (including start and end ones and

assuming that they are equidistant) in which the XPS controller will generate trigger signals for external electronics. Simultaneously with generating a particular external trigger signal, the XPS controller writes the corresponding wire coordinate into its local memory buffer. The XPS trigger signals are caught by a DMM device dedicated to the moving wire setup. This device writes the information about the flux change induced voltage of the moving wire at those moments into its internal memory buffer. The MMC application assures that the DMM device configuration follows the XPS controller settings. When the wire motion along the specified trajectory is finished, the MMC application transfers XPS and DMM device local memory buffers into the EPICS waveform records associated with a 2D trajectory representation and corresponding wire current values, which immediately makes measurement data available for archiving, post processing, modeling, etc.

CONCLUSIONS

A new magnet measurement system has been in successful operations at PSI for the last few months. The system is automated following PSI controls standards. The automation software is implemented as a set of tools supporting Hall probe, moving wire, and vibrating wire magnet measurement setups. Each tool consists of a main control application handling the measurement process and a set of GUI panels, which are used to run that application and monitor its state. Being a part of the EPICS control environment, the magnet measurement data are very easy to work with. For instance, the data archiving is done with the use of a standard EPICS Archiver. The applications written in popular scripting languages (Python, Lua, Bourne shell, etc.) allow users to efficiently handle magnet measurement processes remotely, perform on-line and off-line data analysis, and generate measurement data reports. Further system developments are going to be concentrated on the design and implementation of EPICS 4 data services [7], which should organically combine the measurement data, the data processing models, and the data processing results for each particular magnet into a transparently structured data object associated with this magnet.

REFERENCES

- [1] E. Norum et al., "EPICS: Recent Developments and Future Perspectives", ICALEPCS'2003, Proceedings, p. 278-281, Gyeongju, Korea, October 2003. TU601.
- [2] <http://qt-project.org>
- [3] R. Ierusalimsky, "Programming in Lua", third edition, Lua.org, 2013, 366 p.
- [4] A. Temnykh, "Vibrating Wire Field Measuring Technique", NIM., A 399 (1999), p. 185.
- [5] <http://www.zhinst.ch>
- [6] <http://www.wago.com>
- [7] G. White et al., "EPICS Version 4 Progress Report", ICALEPCS'2013, Proceedings, p. 956-959, San Francisco, USA, October 2013. TUCOB04.

STATUS OF CONTROL SYSTEM FOR THE TPS COMMISSIONING

Y. S. Cheng, Jenny Chen, C. Y. Wu, P. C. Chiu, C. Y. Liao, K. H. Hu, Demi Lee,
Y. T. Chang, C. H. Huang, S. Y. Hsu, C. H. Kuo, C. J. Wang, K. T. Hsu
National Synchrotron Radiation Research Center, Hsinchu 30076, Taiwan

Abstract

Control system for the Taiwan Photon Source (TPS) project has been implemented. The accelerator system began to be commissioning from third quarter of 2014. Final integration test of each subsystem will be done. The EPICS was chosen as the TPS control system framework. 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, 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. Development and test of the high level and low level software systems are in final phase. The efforts will be summarized at this report.

INTRODUCTION

The TPS [1] is a latest generation of high brightness synchrotron light source which is being in construction at the National Synchrotron Radiation Research Center (NSRRC) in Taiwan. It consists of a 150 MeV electron linac, a booster synchrotron, a 3 GeV storage ring, and experimental beam lines. Civil construction had started from February 2010. The construction works are approximately finished in half of 2013. Accelerator system installation and integration was proceeding in later 2013. The control system environment was ready in half of 2014 to support subsystem final integration test and commissioning without beam. Commissioning with beam was started from August 2014.

Control system for the TPS is based on the EPICS framework [2]. The EPICS toolkit provides standard tools for display creation, archiving, alarm handling and etc. The big success of EPICS is based on the definition of a standard IOC structure together with an extensive library of driver software for a wide range of I/O cards. The EPICS toolkits which have various functionalities will be employed to monitor and to control accelerator system.

The control system consists of more than a hundred of EPICS IOCs. The CompactPCI (cPCI) IOC will be equipped with input/output modules to control subsystems as standard IOC or the TPS control system. The power supply and fan module of the cPCI crate will be hot-swapped. Adopting cPCI platform for EPICS IOCs provides us a chance to take advantages of local IT industry products with better supports and low cost. The other kinds of IOCs are also supported by the TPS control system, such as BPM IOC, PLC IOC, various soft-IOC

and etc. The IOCs, control consoles and servers are also running Linux operation system.

To achieve high availability of the control system, emphasis has been put on software engineering and relational database for system configurations. Data channels in the order of 10^5 will be serviced by the control system. Accessibility of all machine parameters through control system in a consistent and easy manner contributes to the fast and successful commissioning of the machine. High reliability and availability of TPS control system with reasonable cost and performance are expected.

CURRENT STATUS

Installation for the control system is almost done. The system is ready to do final integration with subsystems. The software environment is ongoing final revision to connect various subsystems. Control related applications are on-going before subsystem ready in June/August 2014. The progress of the control system is summarized in following paragraphs.

Networking

Mixed of 1/10 Gbps switched Ethernet are deployed for the TPS control system [3]. The Gigabit Ethernet connection will be delivered at edge switches installed at control and instruments area (CIA). One CIA corresponding to one cell of the storage ring, there are 24 CIAs in total. 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 will support fast Ethernet and GbE. Adequate isolation and routing topology will balance between network security and needed flexibility. The file and database servers are connected to the control and intranet network, allowing the exchange of data among them. Availability, reliability and cyber security, and network management are focus in the design phase.

Equipment Interface Layer

There are several different kinds of IOC at equipment layer to satisfy various functionality requirements, convenience, and cost consideration. Most of the devices and equipments will be connected to cPCI IOCs with EPICS running directly. The 6U cPCI platform was chosen for the EPICS IOC. To simplify various developments at construction phase, only 6U modules are supported for the machine control system. The cPCI EPICS IOC equipped with the latest generation CPU board will be standardized as ADLINK cPCI-6510 CPU module [4]. The CPU module equipped with Intel Core i7

CPU running Linux provides high performance to meet various applications.

The cPCI-7452 128 bits DI/DO module is used for BI, BO solution, this high density version in 6U form-factor satisfy most of applications. Industry pack (IP) carrier board in 6U cPCI form-factor can equip up to 4 IP modules. Various IP modules are adopted for required applications. ADC and DAC modules in IP module form factor will be used for smaller channel count application, such as insertion devices control. Event system modules are in 6U cPCI form-factor. Private Ethernet will be heavily used as field-bus to connect many devices. Power supplies of all magnets except for correctors are equipped with Ethernet to the EPICS IOC. Multi-axis motion controller with Ethernet interface will be the standard for the control system.

Ethernet attached devices will connect to the EPICS IOC via private Ethernet. Some network attached devices might not work properly if the network traffic is high due to the simple TCP/IP stack implementation. Private network with lower traffic is to ensure the reliability and performance of the links. Devices support VXI-11, LXI, Raw ASCII and Modbus/TCP protocol are supported to connect to EPICS IOC directly via 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 supply will be driven by the corrector power supply controller (CPSC) module [5]. The CPSC equip with 20 bits DAC and 24 bits ADC. Two SFP ports supported by the on board FPGA (Spatan 6), these SFP ports will receive correction setting (Aurora and Gigabit Ethernet by using UDP/IP protocol) from fast orbit feedback FPGAs to slow orbit feedback PC, feed-forward correction computer and IOC. Setting command sent to these SFP ports will be added with the slow setting from EPICS CA client.

Power Supply System Control

TPS power supplies control interface are divided into three categories rather than a unified solution. All of the power supplies were provided by three different vendors. The reason of this choice is to meet the practical situation from budget and available vendors and manpower.

The small power 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 design CPSC module will be installed at control slot of the sub-rack. The CPSC will be embedded with EPICS IOC and provide fast setting SFP ports to support orbit feedback functionality. Power supply modules installed at the same sub-rack will interface to this CPSC module. The CPSC installed 20 bit DAC and 24 bit ADC to ensure necessary performance for the orbit control especially in vertical plane of the storage ring. To simplify the type of power supply, this category power supply will be used for the corrector of

LTB/BTS/Booster Synchrotron, and storage ring as well as skew quadrupole magnet power supply. The CPSC modules support waveform capability which can support corrector ramping for the booster synchrotron if necessary.

The intermediate power supply with current rating 250 Amp will be equipped with Ethernet interface. Power-supplies are expected to have internal data buffer with post-mortem capability. Output current of the power supply will output at rear plane BNC connector, which can connect to the cPCI ADC module also. There are two versions of power supply in this category, sextupole power supply with 16 bits resolution and quadrupole power supply with 18 bits resolution DAC. Both kinds of power supply can meet the 50 ppm and 10 ppm performance specifications for long-term drift. Noise level in the 10 ppm range was achieved.

Storage ring dipole DC power supply and power supplies for the dipoles and quadrupoles of the booster synchrotron were contracted to Eaton. Each power supply equip with RS-485 serial interface. MOXA serial to Ethernet adapters enable directly. All of these power supplies will interface with the EPICS IOCs directly. The storage ring dipole will be control via this link. Booster dipole and quadrupole power supplies will interface by precision analogue interface; DACs and ADCs operated synchronize by the same clock and trigger to achieve better reproducibility. Waveform generate from the DAC on IOC will drive these booster power supplies. This functionality is essential for energy ramping of the booster synchrotron [6]. Control resolution of these power supplies has 18 effective bits.

Timing System

The event system consists of event generator (EVG), event receivers (EVRs) and a timing distribution fiber network [7, 8]. EVG and EVRs can be installed with various universal I/O mezzanine modules to meet different input/output requirements. The mechanical form factor of EVG and EVRs is in 6U cPCI module. The 125 MHz event rate will deliver 8 nsec coarse timing resolution. Fine delay is 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. This mechanism simplifies the operation of the machine and allows complex sequences of events to be carried out by changing few parameters. The system is ready for system integration test and commissioning.

Insertion Devices Control Interface

Insertion devices (ID) control for the phase I project include one set of EPU46, two sets of EPU48 [9] 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). The motion mechanism of EPU46 and EPU48 are driven by servo motors. Undulator IU22-2m are driven by stepping motors, IU22-3m and IUT22-3m are drive by servo motors. Motion control is done by the

Galil DMC-404x motion controller. In-house developed EPICS devices supports for this motion controller is in use. A cPCI EPICS IOC equips with AI/AO/BI/BO I/O modules will serve an ID. The SSI optical encoder is selected for all IDs. The encoders are connected to the motion controller directly. All parameters of motion controller will be created as EPICS PV. Update rate may be up to 200 Hz. This would be useful for feed-forward compensation process.

Diagnostic System Interface

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. Post-mortem analysis can help to find the weakest point and provide information to improve system reliability.

High precision beam current reading and lifetime calculation will be done at a dedicated IOC. This IOC will install EVR to received booster cycle timing signals and high resolution IP ADC modules to digitize the DCCT signal and perform beam lifetime calculation.

The GigE Vision digital cameras will capture images for diagnostic purposes and other applications.

Counting type and integrating type beam loss monitors will be connected to the control system by counter or ADC modules installed at IOCs.

PLC and Interlock Solution

The Yokogawa FM3R PLC will be used for most of control system related interlock system [10]. FM3R with embedded EPICS IOC is also used for some applications, such as pulse magnet power supply control. Each subsystem is responsible to build their own interlock and protection system based on their requirement and preference. The global machine interlock system will collect various interlock signals from local interlock subsystem of orbit, vacuums, front-ends, radiation dosage monitors and etc. The beam disable commands to trip beam or inhibit injection can be distributed to the specific devices or subsystem by the global machine interlock system or uplink functionality of the event system.

Operator Interface

The operator interface level consists of Linux PCs for consoles and servers for various purposes. OPI is implemented by EDM, MATLAB and CS-Studio. Consoles in the control room have multiple LCD screens as shown in Fig. 1. These OPI consoles will be installed at the equipment area of control room with optical PCIe extension to remote display unit at control room. This option provides better cooling for the computer, reduce loudness at control room and provide clean control consoles. Large screen format displays hang on the roof at control room will be available for display of important

parameters likes beam current, lifetime, vacuum distribution, synchrotron radiation image and etc.



Figure 1: Console in the TPS control room includes multiple LCD screens and ready for use.

Control Applications

Generic applications provided by the EPICS toolkit will be used for all kinds of applications. 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.

SUMMARY

The TPS control system take advantages of the latest hardware and software developments to deliver high performance and rich functionality, and to be economical. The TPS control system environment is ready in June/July 2014 to support various subsystems final integration test. Commissioning of the TPS booster ring is in progress, and commissioning of the TPS storage ring is scheduled in later 2014.

REFERENCES

- [1] TPS Design Book, v16, September 30, 2009.
- [2] EPICS, <http://www.aps.anl.gov/epics/index.php>
- [3] C. H. Huang et al., "Network Architecture at Taiwan Photon Source of NSRRC", these proceedings, and reference therein.
- [4] ADLINK, <http://www.adlink.com>
- [5] D-TACQ, <http://www.d-tacq.com>
- [6] P. C. Chiu et al., "Control Supports for TPS Booster Synchrotron", Proceedings of IPAC 2014. THPRO121.
- [7] Micro Research Finland, <http://www.mrf.fi>
- [8] C. Y. Wu et al., "Integration of the Timing System for TPS", Proceedings of IPAC 2014. TUPRI113.
- [9] C. Y. Wu et al., "Control System of EPU48 in TPS", Proceedings of IPAC 2014. THPRO123
- [10] C. Y. Liao et al., "Implementation of Machine Protection System for the Taiwan Photon Source", Proceedings of IPAC 2014.

NETWORK ARCHITECTURE AT TAIWAN PHOTON SOURCE OF NSRRC

Chih-Hsien Huang, Yin-Tao Chang, Yung-Sen Cheng, Chang-Hor Kuo and Kuo-Tung Hsu
NSRRC, Hsinchu 30076, Taiwan

Abstract

A robust, secure and high throughput network is necessary for the 3 GeV Taiwan Photon Source (TPS) in NSRRC. The NSRRC network divides into several subsets according to its functionality and includes CS-LAN, ACC-LAN, SCI-LAN, NSRRC-LAN and INFO-LAN for the instrumental control, subsystem of accelerator, beam-line users, office users and servers for the information office respectively. Each LAN is connected via the core switch by routing protocol to avoid traffic interference. Subsystem subnets connect to control system via EPICS based channel-access gateways for forwarding data. Outside traffic will be block by a firewall to ensure the independence of control system (CS-LAN). Various network management tools and machines are used for maintenance and troubleshooting. The network system architecture, cabling topology and maintainability will be described in this report.

INTRODUCTION

Taiwan photon source (TPS) [1] is a 3 GeV synchrotron radiation facility with ultra-high photon brightness and extremely low emittance in National Synchrotron Radiation Research Center (NSRRC). The construction began in February 2010, and the commissioning started in the third quarter of 2014.

A high secure and robust network is necessary for such a new accelerator. In order separate the traffic of various kinds of users, 5 layer-3 middle switches are used. TPS control system used for the operations of accelerators and beamlines is implemented by the experimental physics and industrial control system (EPICS) [2, 3] software toolkit. Control devices are connected by the control network and integrated with EPICS based input output controller (IOC).

THE NETWORK ARCHITECTURE OF NSRRC

The NSRRC campus hosts a 1.5 GeV Taiwan light source (TLS), several buildings for office and experimental users, and TPS building. In order to maintain the network functionality of the existed infrastructure and transport to TPS new infrastructure, it is gathered by a middle-layer switch labelled with NSRRC-LAN that the existed wired network including the control system of TLS in the first step. The bandwidth of the backbone upgrades to 10 Gbps and upgrades to 1 Gbps for users gradually. During the upgrade, vlan introduces to the setting of switches in order to replace the use of the public internet protocols (IPs).

In the building of TPS, three subnets including CS-LAN, ACC-LAN and SCI-LAN are created for the control system, subsystems of accelerator and beam-line, shown in Fig. 1. The office automation users belong to the NSRRC-LAN. The subdomain of each LAN is setting in a layer-3 switch. The traffic between middle switch and core switch is through the routing protocol. Single mode fibers supporting 10/40 Gbits/sec link between network rooms and equipment areas at reasonable cost. 10 G links are setup as backbone at this moment.

Control and subsystem network services are available in 24 control instrumentation areas (CIAs) with an individual edge switch in the inner ring area. Major devices and subsystems are installed inside CIAs. There are 4 network rooms and one network & server room outside the ring. These provide the network for the experimental users and the fiber adapter of timing system and intranet network between the middle-layer switch and edge-layer switch, which is installed within beam line station.

There are many servers (e. g. e-mail server, AD server, employee portal server, Web server) belonging to the information office that serve for the employee or various kinds of users. An INFO-LAN will be created for this kind of usage after the servers are moved to TPS. Two subnets with one public IP and one private IP will be signed in the INFO-LAN for the internet and intranet servers.

The traffic of the wireless LAN (WLAN) is independent of the wired network in order to insure the normal operation of the wireless network while the local wired network breaks down. Each access point (AP) broadcasts four service set identifiers (SSIDs), i.e. NSRC-Staff, NSRRC-TEL, NSRRC-Roaming and NSRRC-Guest for staff, IP phone, roaming and guest users. Each user must be certificated before using the wireless. For the staff users, the traffic is direct into core switch without passing through firewall for the intranet access. However, for the Roaming or guest users, the traffic must pass through the firewall and the usage of the intranet is limited by the policy of the firewall.

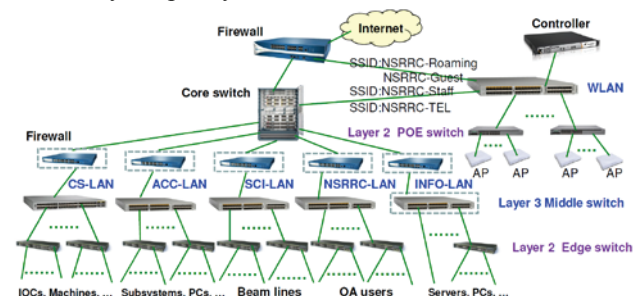


Figure 1: The network infrastructure in NSRRC campus.

ACCELERATOR CONTROL SYSTEM NETWORK

Accelerator operators are the principal users of the control system. Control consoles with remote multi-display are used to manipulate and monitor the accelerator through network. In order to remote monitoring and control TLS facility, dedicated control consoles are also installed in TPS control room. The control console computers, EPICS control servers, database servers, and network equipment are all in the network and server room.

Control network services are not only available at the control room, but also at the network and server room, 24 CIAs, linear accelerator equipment area, tunnel, transport lines, main power supply equipment room and control system laboratories. The TPS control network infrastructure is shown in Fig. 2. Connection between CS-LAN and the core switch will be through a firewall. Outside traffic will be blocked by the policy of firewall except for some particular purposes such as the remote access for maintaining machines which is proposed in advance.

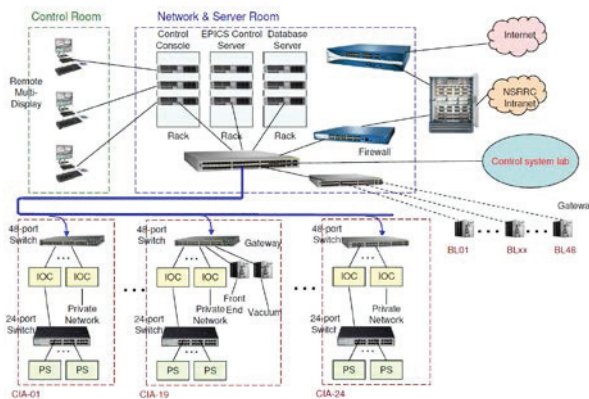


Figure 2: TPS control network infrastructure.

A high performance switch with 48 10-Gbps and 4 40-Gbps fiber ports is defined as the middle-layer switch of CS-LAN. There are two types of switches used in every CIA for control system. The first type is defined as the edge switch which is used to connect IOC nodes and uplink to the high-speed backbone through 10-Gbps fiber uplinks. A 24-port switch is selected as the switch for gathering the uplink for power supplies.

One class B subnet (172.20.0.0) is used for IOC network. For the IP 172.20.xx.yy, xx represents locations (e.g. number of CIA) and yy identifies for functional groups. This IP addressing schema makes identify the locations of IOCs and devices easily. This is also helpful to speed up finding the devices for maintenance and troubleshooting. There are multiple Class C private networks for respective subsystems, such as power supplies, motion controllers, GigE Vision, etc. These Class C private networks use IP range 172.21.xx.yy in which schema is also the same as above.

The fieldbus of the TPS control system needs highly reliable Ethernet. Power supplies for dipole, quadrupole

and sextupole are connected to the EPICS IOCs by Class C private Ethernet within the CIAs. In order to reduce the network traffic and provide additional access security, EPICS based CA gateway or IOC provide necessary connectivity and isolation. Its functionality is to forward channel access to different network segments.

GigE vision for diagnostics is based on the IP standard and can be adapted to EPICS environment. The images can be easily accessed through network for machine studies. The GigE vision cameras connect to control system through Ethernet with the data transfer rate up to 1 Gbits/s. To decrease traffic loading, one Class C private network and one CA gateway will also be used for the GigE vision cameras to connect with the control system.

ACCELERATOR SUBSYSTEM NETWORK

Many technical groups prefer their own subnets to monitor and control their system outside the control system (CS-LAN). The ACC-LAN serves for this purpose for vacuum, front-end, glider control system, radiation monitoring and access control system, RF system, etc., shown in Fig. 3. It only responses for the non-critical safety data transfer. Two subnets of private IP, labelled as intranet private IP (IPIP) and regional private IP (RPIP) are provided for each subsystem. The traffic of RPIP is limited within its subnet and the traffic of the IPIP is allowed inside NSRRC campus via the routing of core switch and outside the Campus via the network address translation (NAT) of the firewall. The IPIP provides the convenience to remote monitoring the status of each subsystem from the office and the RPIP provides the network security protection of each machine. The data exchange between each sub-system and the control system is through dedicated EPICS gateways.

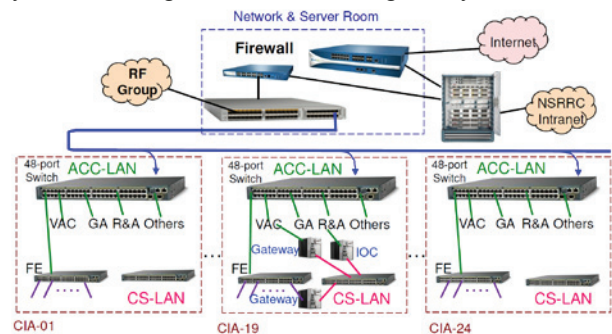


Figure 3: TPS ACC-LAN infrastructure for the vacuum (VAC), front-end (FE), glider alignment (GA), radio frequency (RF), radiation monitoring and access (R&A) control system, etc.

BEAM-LINE NETWORK

The SCI-LAN is planned for network of the TPS beam-line and experimental stations. Each beam-line has a Class C RPIP for control and data acquisition and IPIP for the internet/intranet access. Data exchange with the accelerator control system is via a dedicated EPICS gateway for each beam-line, shown in Fig. 4. The TPS

networks support high through data transfer with 10/40 Gbps rate between beam-line and experimental station to the storage farm or computation farm which locates at different area within the TPS facility.

Beam-line users can access accelerator control system by PVs via CA gateways. This design provides necessary connectivity between the machine control system and beam-line control system and also restricts unnecessary network traffic across different network segments.

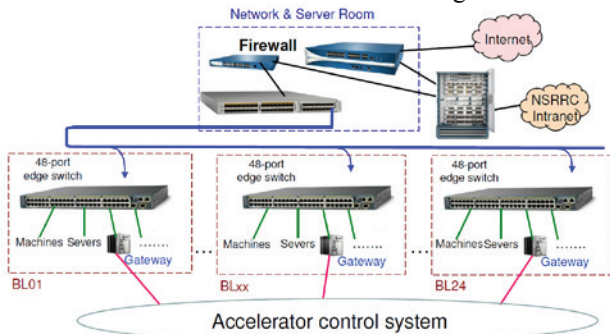


Figure 4: TPS SCI-LAN infrastructure.

NETWORK MANAGEMENT

Spanning tree protocol (STP) or rapid spanning tree protocol (RSTP) is configured to prevent looping. Network monitoring software (e.g. Cacti, MRTG,) is used to monitor traffic and usage information of the network devices. By collecting and analyzing the packets, it can measure the traffic to avoid bandwidth bottlenecks.

It is necessary to access the control system from outside to realize the machine problems. Remote maintenance or troubleshooting has the advantages of convenience and time-saving. Virtual private network (VPN) is used to penetrate the firewall system of the protected network. It can establish an encrypted and compressed tunnel for TCP or UDP data transfer between control network and public networks inside or outside the TPS.

The network time protocol (NTP) and precision time protocol (PTP) serve for timekeeping. NTP is used for synchronizing the clocks of computer systems over the TPS network with millisecond precision. PTP is served also for microsecond precision application.

CYBER SECURITY

Current accelerator control systems are commonly based on modern Information Technology (IT) hardware and software, such as Windows/Linux PCs, PLCs, data acquisition systems, networked control devices, etc. Control systems are correspondingly exposed to the inherent vulnerabilities of the commercial IT products. Worms, viruses and malicious software have caused severe cyber security issues to emerge.

It is necessary to use network segregation to protect vulnerable devices. Combining firewall, NAT, VLAN... technologies, control network is isolated to protect IOCs and accelerator components that require insecure access services (e.g. telnet).

Firewall only passes the packets from authorized hosts with pre-defined IP addresses outside control network and opens specific service ports for communications. But firewall is not able to resist the spread of worms. Worms are not only designed to self-replicate and spread but also consume the network bandwidth. Intrusion prevention system (IPS) can detect and stop network threats such as worms, viruses, intrusion attempts and malicious behaviors. The next generation firewall which combines functionality of firewall and IPS is adopted for TPS network. Security always puts at the highest priority for the TPS networking system. Restricted security policy for network is essential for TPS facility and equipment. However, balance between security and convenience will be addressed also.

CURRENT STATUS

All backbone fiber network are finished in October, 2013. Full access of the network for ACC-LAN, CS-LAN, WLAN is available in the first quarter of 2014 for the subsystem installing and beam commissioning. The network for the SCI-LAN is scheduled and under installing. That will be finished before the users of beamline is needed.

It is finished that the basic function of network manager tools such as Cacti and IPScan which provide us to realized the traffic of each node and the IP or Mac using in the network. The detail setting is under way for obtaining the complete information.

SUMMARY

This report describes the infrastructure of network in NSRRC. NSRRC-LAN, ACC-LAN, CS-LAN, SCI-LAN and INFO-LAN are or will be created for various users. An individual wireless LAN is also setup for ensure the network service while local wire network is out of function. An adaptive, secure and fault-tolerant control network is essential for the stable operation of the TPS. The control network is separated from the general purpose network for imposing security. Subsystem and beam-line subnets connect to the control system via EPICS CA gateways or IOC for forwarding data and reducing network traffic. Network management tools are used to enhance productivity. Remote access mechanism with proper authentication is implemented for the system maintenance and troubleshooting.

REFERENCES

- [1] TPS Design Book, v16, September 30, 2009.
- [2] EPICS, <http://www.aps.anl.gov/epics/>.
- [3] Y. S. Cheng et al., "Construction of the TPS Network System", ICALEPCS 2013, San Francisco, USA.

BPM CONTROL, MONITOR, AND CONFIGURATION ENVIRONMENTS FOR TPS BOOSTER

P. C. Chiu, K. H. Hu, C. H. Kuo, Y. S. Cheng, K. T. Hsu, NSRRC, Hsinchu 30076, Taiwan

Abstract

Booster synchrotron for the Taiwan photon source project which is a 3 GeV synchrotron light source constructed at NSRRC is in commissioning. The BPM electronics Libera Brilliance+ [1] are adopted for booster and storage ring of Taiwan Photon Source (TPS). The acceptance test had been completed in 2012 [2]. The provided BPM data is useful for beam commissioning where it can be used to measure beam position, rough beam intensity along the longitudinal position and also for tune measurement. This report summarizes the efforts on BPM control, monitor and configuration environment.

INTRODUCTION

The TPS is a state-of-the-art synchrotron radiation facility featuring ultra-high photon brightness with extremely low emittance [3]. Civil constructions had been completed in early 2013. 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 which include 7 BD dipoles with 1.6 m long and 2 BH dipoles with 0.8 m long in each cell. Its circumference is 496.8 meters and it is concentric with the storage ring in the same tunnel. Libera Brilliance+ electronics has been adopted for the position measurement for booster and storage ring. It has provided precise beam position measurement and useful tools during booster commissioning in progress. This report will summarize the booster BPM related environment and functionalities.

BOOSTER BPM LAYOUT AND FUNCTIONALITIES

The TPS booster ring has six cells where each cell is equipped with 10 BPMs which can be used to measure beam position and rough beam intensity along the longitudinal position. Fig. 1 shows the mechanical drawing of booster BPM which shapes 35x20 mm elliptical and button diameter 10.7 mm. The calibration factor K_x and K_y is 8.25 and 9.66 mm respectively.

The conceptual functional block diagram of the BPM electronics is shown in Fig. 2. It will provide several data type for different application. ADC and TBT data is acquired on demand by trigger; 10 Hz slow data is for DC average orbit and 10 kHz fast data could be applied for booster ramping orbit or fast orbit feedback application. It is also embedded with EPICS IOC for control, monitor and configuration. The timing AMC module would provide functionalities of synchronization, trigger, interlock and post-mortem.

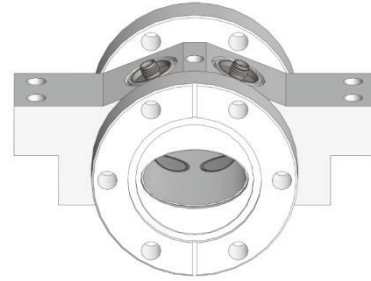


Figure 1: Button-type BPM for TPS Booster.

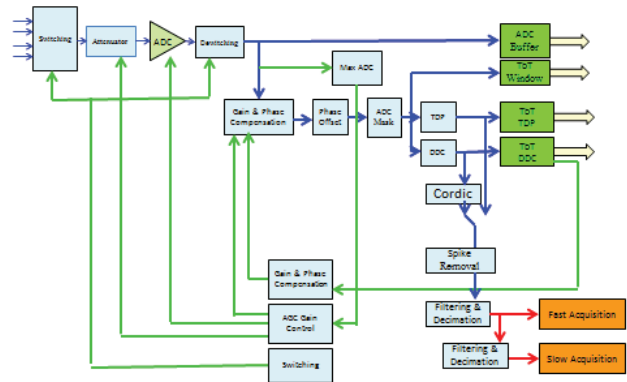


Figure 2: BPM platform functional block diagram.

FUNCTIONALITY TEST AND MANAGEMENT ENVIRONMENT

To support operation of the BPM electronics, functionalities like cold start, shutdown, housing, control system interface should meet the requirements. The delivered units also had been performed functionality and performance test to ensure compliance with this specification.

Cold Start and Shutdown

All BPM platforms and BPM electronics have been performed cold and shutdown test. Only minor problems encounter. Non-booting issue due to incorrectly shutdown procedure is reported to the vendor. Uploading the new FPGA image would fix it.

EPICS Interface

The EPICS interfaces for all BPM platforms are tested. Several defects is identified and then resolved after firmware upgrades. GUI based on EDM and Matlab is also developed for configuration and monitor.

Housekeeping

The BPM platform provides power supply status monitoring, temperature monitoring, fan status monitoring and fan control. The status looks good from the system maintenance point of view. EDM pages are

also provided for status monitor and display as Fig. 3.

Synchronization Stability

Synchronization is accomplished by internal PLL to lock to the revolution frequency and the stability is observed for 9 modules during 60 hours. The phase errors had never gone out of $\pm 1^\circ$ region for the PLL had always remained locked status.

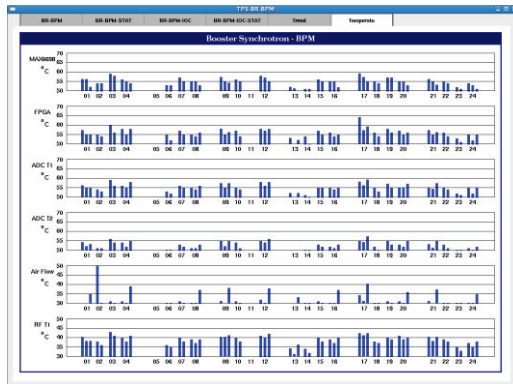


Figure 3: EDM Page for BPM housekeeping.

BOOSTER BPM MEASUREMENT

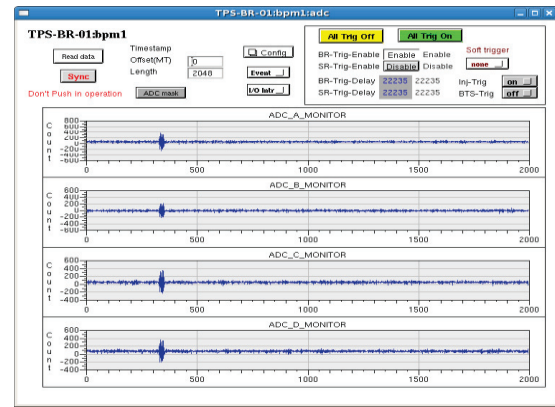
For TPS booster BPM, there are 60 sets of phase-trimmed 0.240" form polyethylene coaxial cables connected between the buttons and BPM electronics. The gain variation of BPM electronics is less than 5%. The equal length of all cable sets will contribute the same power consumptions and make possible to use the sum signal of BPM as an intensity indicator. The error would only come from position dependence of the beam. The BPM position and intensity data is useful for beam commissioning. In this section, different BPM data flow will be demonstrated for different applications.

ADC Raw Data

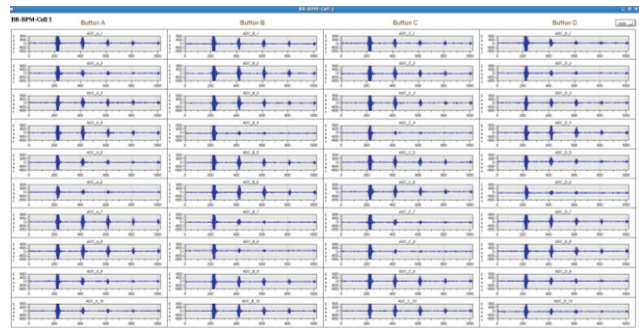
The ADC raw data is useful for checking the timing of the beam and beam property especially in the first turn. Fig. 4(a) shows that the first beam passing through the injection septum and kicker and arriving the 1st BPM of the booster ring; (b) the beam circulates complete 5 turns in the booster.

First-turn & Multi-Turn Application

BPM electronics provides single pass mode for calculating first turn trajectory from ADC data. However, vast beam losses and ADC DC offset will result in worsen signal to noise ratio and position calculation error. Therefore, a soft IOC would be applied to acquire more precise first turn trajectory from ADC raw minus DC offset. Fig. 5 shows the first turn orbit trajectory and sum along 60 BPMs. Horizontal trajectory shapes like dispersion function due to energy drift from Linac modulator. Moreover, ADC buffer with 2048 length could be applied to provide the first 9 turns data developed by Matlab as Fig. 6.



(a)



(b)

Figure 4: (a) The ADC data when beam passes through the 1st BPM of the booster synchrotron. (b) The ADC data as the beam circulates complete 5 turns in the booster.

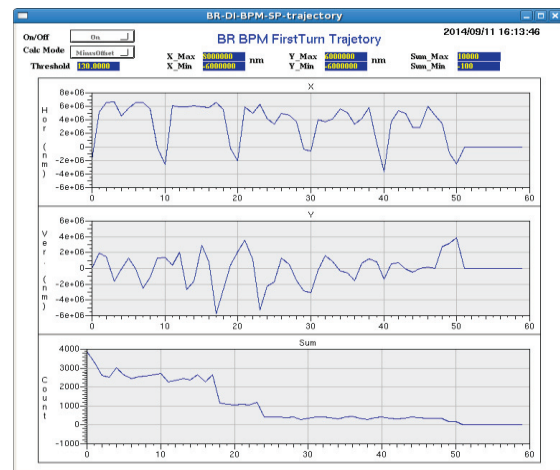


Figure 5: First turn horizontal, vertical trajectory and sum along 60 BPMs of booster.

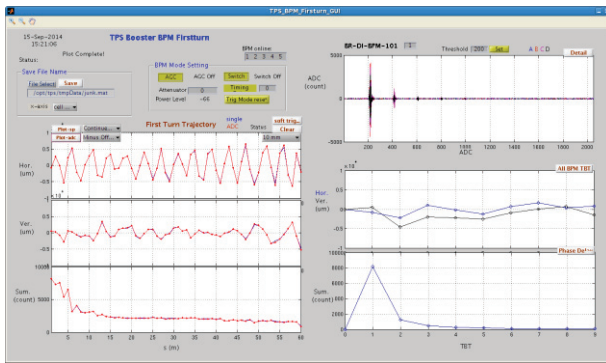


Figure 6: The first 9 turns data calculated from ADC.

Turn by Turn Application

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 μm at 0.5 m. To use TDP properly, phase offset should be adjusted by beam and mask window also should be set correctly according bunch length. It could be observed that the beam charge loss could be well resolved. Besides, the BPM TBT data could also be applied to calculate tune value as Fig. 7.

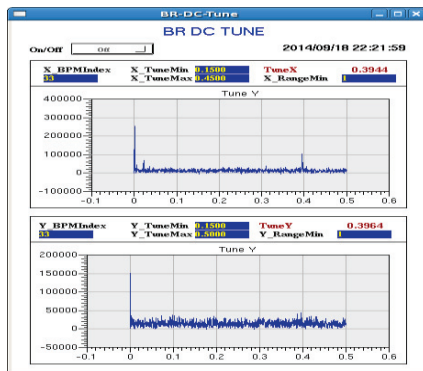


Figure 7: Tune Measurement from BPM TBT data.

Slow & Fast Position Measurement

The BPM electronics also provide 10 Hz slow and 10 kHz fast position data to measure average stored beam orbit. The related applications including GUI and acquiring scripts are developed and provided for studying and helping commissioning as Fig 8 & Fig 9.

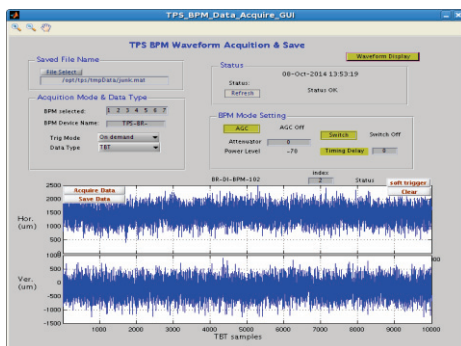


Figure 8: Acquisition tool for different data type of BPM.

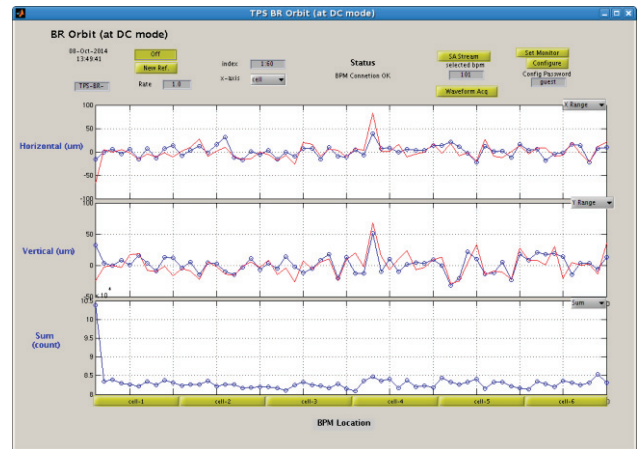


Figure 9: Booster DC orbit display page.

CURRENT STATUS

Final system integration test of booster with the installation of the storage ring and preliminary beam commissioning of the booster synchrotron is in progress at NSRRC now. Beam was circulated one turn soon after commissioning work started. To store beam and ramp the energy into 3 GeV are the recent efforts. BPM functionalities and performance for the booster synchrotron has been exercised with beam during last several weeks. Supporting tools of software have been continuously revised. Preliminary results show the BPM being a useful diagnostic tool during beam commissioning.

ACKNOWLEDGEMENT

Thanks for the help from Y. S. Cheng, Y. T. Chang, and Demi Lee. The authors appreciate help from staffs of I-Tech for brainstorming and discussion.

REFERENCES

- [1] Instrumentation Technologies: <http://www.i-tech.si/>.
- [2] P. C. Chiu, et al., "TPS BPM electronics Performance Measurement and statistics", Proc. IBIC 2012, Tsukuba, Japan, <http://jacow.org/>.
- [3] TPS Design Handbook, version 16, June 2009

A MODULAR PERSONNEL SAFETY SYSTEM FOR VELA BASED ON COMMERCIAL SAFETY NETWORK CONTROLLERS

D.M. Hancock, B.G. Martlew
STFC Daresbury Laboratory, Warrington, UK

Abstract

STFC Daresbury Laboratory has recently commissioned VELA (Versatile Electron Linear Accelerator), a high performance electron beam test facility. It will be used to deliver high quality, short pulse electron beams to industrial users to aid in the development of new products in the fields of health care, security, energy and waste processing and also to develop and test novel compact accelerator technologies. In the early stages of the design it was decided to use commercial Safety Network Controllers and I/O to implement the Personnel Safety System in place of the electro-mechanical relay-based system used on previous projects. This provides a high integrity, low cost solution while also allowing the design to be modular, programmable and easily expandable. This paper describes the design and realisation of the VELA Personnel Safety System and considers its future development. In addition, the application of the system to the protection of high-power laser systems and medical accelerators will also be discussed.

INTRODUCTION

The VELA photoinjector consists of a 2.5 cell S-band RF gun using a copper photocathode, driven by a sub-100fs UV laser, designed to provide low emittance, short pulses of electrons. VELA features a suite of diagnostics including YAG screens for transverse beam profiles, a variety of emittance measurement devices, an energy spectrometer dipole and a transverse deflecting cavity for bunch length and longitudinal phase-space measurements [1]. The machine is housed inside a 2m thick concrete enclosure designed to provide shielding for future higher energy upgrades. The accelerator vault is split into 3 distinct areas; the accelerator room, beam area 1 and beam area 2 and is designed in such a way that set-up work can be carried out in either of the beam areas whilst the accelerator is operational. The basement area beneath the accelerator is also shielded. Construction of VELA began in September 2011 with first beam achieved in April 2013 and the first experimental users in September 2013. The general layout of VELA is shown in Fig. 1

PERSONNEL SAFETY SYSTEM OVERVIEW

The VELA Personnel Safety System (PSS) controls the generation of ionising radiation by enabling the operation of the electron gun and RF cavities. The gun and RF sys-

tem may only be operated when the injector room and basement areas have been searched and interlocked. The PSS also controls the transmission of the electron beam to beam area 1 and beam area 2 by enabling the operation of radiation shutters and dipole magnets.

Search Procedure

A two-person search system is employed and each area (accelerator room, basement and beam areas) can be searched independently. A card reader is used to limit initiation of a search to those trained in the correct search procedure. Pairs of search buttons (which must be pressed in sequence) guide the search team along a pre-defined route which covers the entirety of the shielded area.

Warning Devices

‘Secret until lit’ warning signs and audible warning devices alert staff to the status of the accelerator. Illuminated signs use arrays of LEDs (light emitting diodes) which have a low failure rate (the manufacturers quote a MTBF of 100,000 hours which equates to >10 years continuous use). All warning devices are ‘fail-safe’ and this function is achieved by the use of current sensing circuits to detect current flowing through the devices and these circuits provide interlock inputs to the PSS. Sensors used for flashing signs and audible warning devices have a build-in time delay to allow for the variation in current. The sensors have an adjustable current threshold which gives an indication that the device is working when the current threshold is exceeded.

Safety Console

A ‘safety console’ located in the control room provides a number of PSS keys which enable individual accelerator functions. These are:

- Shielding key – removal prevents a search of the accelerator room and basement
- RF Mode Key – removal prevents operation of the RF system
- Run Key – removal prevents operation of the gun by disabling the photoinjector laser shutters
- BA1 Key – removal prevents a search of beam area 1
- BA2 Key – removal prevents a search of beam area 2

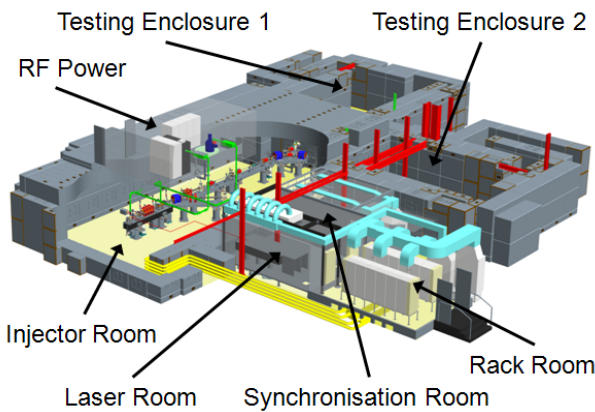


Figure 1: VELA layout.

Radiation Monitors

Gamma radiation measuring instruments are situated outside the shield wall. Readout of the radiation level is via a front panel analogue meter and 'trip' outputs are provided in the form of normally closed relay contacts. If the trip level is exceeded ($> 25\mu\text{S/hr}$), internal visual and audible alarms are activated and the trip relay contacts open. The PSS monitors the state of all radiation monitor trip outputs and takes appropriate action if the trip level is reached. Trip conditions are latched in the PSS and this latched state must be manually cleared before operation of the accelerator can recommence.

System Hardware

Omron NE1A Safety Network Controllers and DST1 series Safety I/O Terminals [2] enable the construction of a safety control network that meets the requirements for Safety Integrity Level (SIL) 3 according to IEC61508 (Functional Safety of Electrical/Electronic/Programmable Electronic Safety Related Systems) [3] and the requirements for Safety Category 4 according to EN 954-1 [4]. The NE1A series controller provides safety logic operations, safety I/O control and a DeviceNet safety protocol. DST1 series safety I/O Terminals support the DeviceNet safety protocol and interface directly to I/O devices. A master-slave relationship is established for each connection on the DeviceNet Safety Network and the status of the safety I/O data can be monitored in a standard Omron CJ2M PLC on the same DeviceNet network using standard I/O communications. This I/O data is transferred over Ethernet to the VELA control system which runs within an EPICS (Experimental Physics and Industrial Control System) environment [5].

Dual channel mode can be set for pairs of related local inputs. When dual channel mode is selected input data patterns can be evaluated and the time discrepancy between input signals can be analysed. Output data patterns can also be analysed.

Test pulses are used to check the NE1A's internal circuits, external circuits and external wiring – enabling the immediate detection of short circuits. To protect external circuits, outputs are cut off when an overcurrent condition is detected.

Data stored in the NE1A-series Controller can be locked to protect the data after it has been downloaded and a password can be set to prevent unintended or unauthorized access to the NE1A-series Controller.

DeviceNet

DeviceNet is an open-field, multi-vendor, multi-bit network which enables the easy connection of Safety Network Controllers, PLCs, sensors and actuators. DeviceNet supports message communications as well as remote I/O communications. A baud rate of 500kbs can be achieved with a maximum of 64 connected nodes.

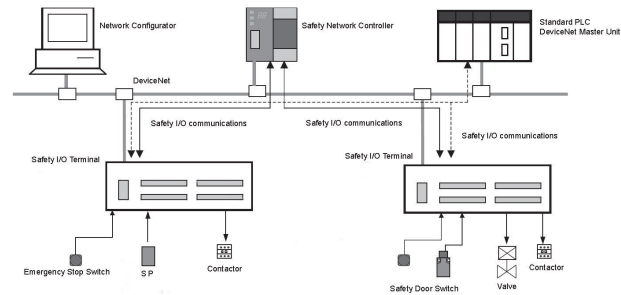


Figure 2: DeviceNet schematic.

Programming

Network Configurator (running on a windows based PC) is the software package that is used to create the complex logic designs required to implement the PSS. Standard logic operations such as AND, OR, NOT are provided along with dedicated function blocks such as emergency stop and safety gate monitoring. The completed program can then be downloaded to the NE1A via a standard USB connection. The state of each safety input/output channel can then be monitored via the same USB connection. Up to 254 logic function blocks can be used in each program and a program password can be set to prevent unintended or unauthorized access to the NE1A-series Controller.

Program Debugging

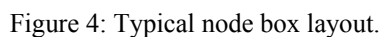
The NE1A *Logic Simulator* software is an offline debugging tool that can be used to test logic designs before downloading to the NE1A. Any combination of input conditions can be set and all output states can be monitored. Program execution can be started, paused and stopped at any point and timing charts can be created.

Physical Layout

The Safety Network Controllers, standard PLC and 24VDC power supplies mounted in 19" racks located in the VELA instrumentation room. I/O modules, wired directly to sensors and actuators, are mounted in *node boxes*, situated inside the shielded areas. A single *thick* DeviceNet cable links all the I/O modules and safety network controllers.



Complex systems can be designed using only 2 types of I/O modules, thus reducing the number of spares required, resulting in a significant cost saving for the project.



Following the successful implementation of the VELA PSS, the safety network controller based system has now become the standard solution for all future PSS applications at Daresbury. Over the last 12 months, the same basic design philosophy has been used to design and

- **Medical Teaching and Research Laboratory (MTRL)** – this is a partnership between STFC, the University of Liverpool and the Royal Liverpool Hospital. The MTRL houses a SPECT/CT scanner which provides hand-on training for MSc students.

REFERENCES

- [1] P.A. McIntosh et al., VELA: A new Accelerator Technology Development Platform for Industry. Proceedings of IPAC 2014, Dresden, Germany.
- [2] OMRON Industrial Automation
website: <http://industrial.omron.co.uk/en/home>.
- [3] IEC61508 International Electrotechnical Commission
website: <http://www.iec.ch/functionalsafety>.
- [4] EN954-1BSI
website: <http://www.bsigroup.com>
- [5] EPICS - Argonne National Laboratory, Advanced Photon Source
website: <http://www.aps.anl.gov/epics>.

BENEFITS, DRAWBACKS AND CHALLENGES DURING A COLLABORATIVE DEVELOPMENT OF A SETTINGS MANAGEMENT SYSTEM FOR CERN AND GSI

R. Mueller, J. Fitzek, H. Huether, GSI, Darmstadt, Germany
G. Kruk, CERN, Geneva, Switzerland

Abstract

The settings management system LSA (LHC Software Architecture) [1] was originally developed for the LHC (Large Hadron Collider). For FAIR (Facility for Antiproton and Ion Research) a renovation of the GSI control system was necessary. When it was decided in 2008 to use the LSA system for settings management for FAIR, the middle management of the two institutes agreed on a collaborative development. This paper highlights the insights gained during the collaboration, from three different perspectives: organizational aspects of the collaboration, like roles that have been established, planned procedures, the preparation of a formal contract and social aspects to keep people working as a team across institutes. It also shows technical benefits and drawbacks that arise from the collaboration for both institutes as well as challenges that are encountered during development. Furthermore, it provides an insight into aspects of the collaboration which were easy to establish and which still take time.

INTRODUCTION

The idea of collaboration between CERN and GSI on the settings management system was first discussed in 2006 between the Controls groups management of both institutes. At that time the development of LSA was well advanced but completion of functionality necessary for the LHC startup in 2008 still required a significant amount work. While GSI was evaluating different possibilities for a new control system that could be used for FAIR, it was agreed that two software engineers from GSI will spend 18 months at CERN to help with the development and commissioning of LSA.

The common development had clear benefits for both institutes. For CERN it was a reinforcement of the LSA team by two skilled developers in the view of the upcoming deadline and a fresh view on the system by new external people, while for GSI it was an excellent occasion to gain a valuable experience, learn about the system and main ideas behind, and to evaluate its possible use at GSI.

Since the first impressions about the portability of LSA were very positive, after the 18 months of joint development at CERN, the two GSI developers deployed a dedicated version of LSA in their home institute for further evaluation. The first goal was to prepare a working version of an LSA-based control system for the existing SIS18 synchrotron, which would help other GSI developers and users get deeper insight in the system and be a final verification of applicability of LSA to the whole accelerator complex at GSI.

After positive feedback and successful machine development sessions on SIS18, it was decided to use LSA for FAIR and continue the collaborative development of LSA by both institutes [2–5].

BUILDING THE TEAM

The initial 18 months of common development in one place was certainly beneficial in building a solid foundation for further collaboration as it created strong bonds within the team who worked together towards shared goals. It gave everyone a good overview about the technical aspects of the control systems in both institutes as well as a good insight into the work processes of the colleagues from the other institute, their constraints and their deadlines. This certainly helped in increasing the mutual understanding and in taking certain decisions related to the collaborative development.

To keep the collaboration active and to maintain the team spirit developed during those 18 months in the longer term, it was perceived as important to maintain a regular contact. This goes beyond exchanging ideas via mail or phone calls to also having in person discussions during regular visits. For the collaboration between CERN and GSI, these take place twice a year and are used to discuss, agree and schedule major changes to be applied in the system.

DEVELOPMENT PROCESS

Modularization

Even though LSA had been organized in a modular way from the beginning, during the common development in 2007 and 2008 it has been further restructured to split the code base into generic and CERN-specific parts, allowing possible extensions by GSI.

Figure 1 shows the current package hierarchy. The common modules in the middle contain the generic settings-management framework and have no outward dependencies, so they can be compiled and released independently. The domain objects and logic related to particular accelerator models, types of equipment, infrastructure (such as timing) and operation modes are implemented in the institute-specific parts (on the left and right side in the Figure 1).

The institute-specific extensions are developed and released independently by CERN and GSI. However all changes done in common modules are subject to review and acceptance by developers from both institutes.

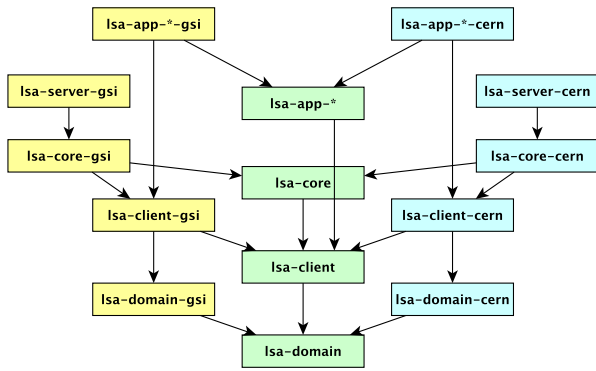


Figure 1: Layout of modules.

Changes in Common Modules

Depending on the nature of the change, the procedure to develop and review it is different. We have defined three categories of code changes that are described in Table 1.

Table 1: Categorization of Code Changes

Term	Descriptions
Major Change	Change that requires modification of the core behavior, data model or APIs with big impact on the system. Deployed only during non-operational periods (shutdown).
Significant Change	New feature or modification of existing functionality. Can typically be deployed during operational periods (technical stops).
Non-significant Change	A change with no impact on the functionality. It must always be backward compatible. Can be deployed at any time.

The *non-significant* changes are done and committed to the source repository directly by both CERN and GSI developers. Those are typically bug fixes, cosmetic changes or improvements of the documentation. Developers from the other institute usually review such changes when synchronizing with the source code repository.

The *significant* changes are discussed upfront by involved developers from both institutes. This is typically done via e-mails, phone calls or video conferences. The review of the code is usually done before committing it to the repository e.g. by sending a patch containing the change set.

The *major* changes are subjects of more in-depth analysis among all developers to find the best solution that works for everyone but also to examine the impact on the overall control system in both institutes. Such changes are typically discussed during visits which take place twice a year. They are then developed in dedicated source repository branches, and merged back to the main branch when ready for release. Deployment into the operational environment is done during longer technical stops (few days) or during shutdown periods to not influence regular operation of accelerators.

Artifacts Repository and Release Procedure

Until now, the management of all controls' software artifacts at CERN is done using Common-Build [6] (an Ant-based build system developed in-house) and a proprietary artifacts repository. Due to several dependencies on CERN's infrastructure this solution could not be used at GSI. Therefore GSI decided to use Maven, one of the major industry standard solutions.

Releases of both the common and institute-specific modules are done independently at CERN and at GSI, and are deployed into repositories at each institute respectively. All dependent artifacts developed at CERN that are necessary to build and run LSA are mirrored from CERN repository to the GSI one via a custom replication mechanism.

COLLABORATION AGREEMENT

As a formal basis for the collaboration, an agreement document is being established between the controls groups in both institutes, as an addendum to the existing overall agreement between CERN and GSI. The document defines various aspects of the collaboration, including project roles, ownership of modules, responsibilities of both parties and decision making process concerning the common development of LSA but also of any other system that may be a subject of the collaboration in the future.

The following four categories of modules have been defined in the document:

- C1: Modules being part of the collaboration
- C2: Generic modules used by C1 (developed by either institute)
- C3: 3rd-party libraries used by C1 or C2
- C4: Institute-specific modules of C1

For (C1) modules the document ensures that each party gets a free and irrevocable license on the source code. For (C2) modules, it grants usage rights. Upgrades to newer version of third party libraries (C3) need to be coordinated between the institutes. The (C4) modules are in the responsibility of each institute.

The two development teams agreed on the general practicalities of the collaboration already in 2009; however the collaboration agreement has not yet been officially finalized. Making it an addendum to the existing agreement between GSI and CERN has required an additional effort and involves other entities, like the legal departments of the two institutes.

BENEFITS, DRAWBACKS AND CHALLENGES

Based on a few years' experience there are pros and cons of the collaboration that has been seen by both parties.

Benefits

One of the biggest profits for both institutes is the joint manpower working on the system and therefore saving man-years of work on both sides.

GSI could join an already mature and relatively stable project, used in operation for several accelerators at CERN.

The foundations of the domain model were established and many conceptual problems had already been resolved. In addition several operational GUIs were implemented and could be used straight away at GSI. This meant that GSI could relatively quickly focus on the institute-specific extensions rather than spend time on developing a new framework.

For CERN, besides the additional development resources, the collaboration influenced establishing a better structure of the code and packages, which are always reviewed by GSI for portability. It also encouraged a more generic approach, making the system more flexible, also for future extensions at CERN. Since GSI was not yet in operation, in several cases new concepts or features developed in the framework could be first validated at GSI, before operational deployment at CERN.

Drawbacks

The major issue for both institutes is the overhead of introducing any significant changes in the common code. Any such modification proposed by one party must be validated by the other party. Checking usability, modeling it in a generic way, reviewing and testing against use cases in both institutes introduces additional delays as compared to if it would be done in one institute only, thereby increasing the time required from the moment a change is proposed until it is released in production.

Challenges

The principal challenges that we currently face are technical ones. Finding a generic solution in the common code that works for both institutes is often non-trivial and requires a few iterations of analysis until a satisfactory result is achieved.

Another challenging area for the collaboration is the Graphical User Interfaces. Even though functionality-wise both institutes have very similar requirements, different appearance or specific features are frequently requested by respective users. This requires designing these tools in a configurable and pluggable manner to allow customization when needed.

There are also several other technical difficulties that must be overcome to streamline the collaborative development such as easy and reliable access to generic libraries developed in both institutes, tracking changes in the database model and synchronizing them with corresponding code changes, or defining a better procedure to deal with urgent change requests that cannot wait for acceptance of the other institute. The latter one will become especially important once GSI enters into the commissioning stage and, eventually, the stable operation phase.

CONCLUSIONS

Several conclusions can be drawn concerning what has been working very well and what we could have done better.

Not putting enough attention to the build and release of shared artifacts from the very beginning caused several problems and required manual work to resolve compatibility conflicts. Another area for improvement is to better organize the flow of information and make change reviews more efficient to not block one of the parties from doing necessary modifications.

The modularization of the system into generic packages and institute-specific extensions before the common development started was certainly a good decision. Also having an initial “gentlemen’s agreement” on how we plan to work together, which was later translated into a formal agreement, gave a solid basis for the future collaboration.

Although the collaboration brings many challenges and has certain drawbacks, our overall experience is very positive and we will certainly continue the common development in the future. We believe that the key aspect of a successful collaboration is the human factor. People involved must feel as a team working together, respect each other, give honest feedback and listen to feedback of others, and finally be prepared for compromises since that is what collaboration often requires in order to move forward.

REFERENCES

- [1] G. Kruk et al., “LHC Software Architecture (LSA) - Evolution Toward LHC Beam Commissioning”, ICALEPCS’07, Knoxville, Tennessee, USA, WOPA03
- [2] R. Mueller, J. Fitzek, D. Ondreka, “Evaluating the LHC Software Architecture for data supply and settings management within the FAIR control system”, ICALEPCS’09, Kobe, Japan, THP012.
- [3] R.C. Bär et al., “Development of a New Control System for the FAIR Accelerator Complex at GSI”, ICALEPCS’09, Kobe, Japan, TUP107.
- [4] J. Fitzek, R. Mueller, D. Ondreka, “Settings Management within the FAIR Control System Based on the CERN LSA Framework”, PCaPAC’10, Saskatoon, Saskatchewan, WEPL008.
- [5] H. Hüther, J. Fitzek, R. Müller, D. Ondreka, “Progress and Challenges During the Development of the Settings Management System for FAIR”, PCaPAC’14, Karlsruhe, Germany, WPO005.
- [6] G. Kruk, “Development Process of Accelerator Controls Software”, ICALEPCS’05, Geneva, Switzerland, FR2_5-60.

ePlanner SOFTWARE FOR MACHINE ACTIVITIES MANAGEMENT

Bakshi Sanjai Kumar Srivastava, Rajesh Kumar Agrawal, Pravin Fatnani,
Raja Ramanna Centre for Advanced Technology (RRCAT), Indore, India

Abstract

For Indus-2, A 2.5 GeV Synchrotron Radiation Source, operational at Indore, India, the need was felt for software for easily managing various related activities for avoiding communication gaps among the crew members and clearly bringing out the important communications for machine operation. Typical requirements were to have the facility to enter and display daily, weekly and longer operational calendars, to convey system specific and machine operation related standing instructions, to log and track the faults occurring during the operations and follow up actions on the faults logged etc. Overall, the need was for a system to easily manage the number of jobs related to planning the day to day operations of a national facility. The paper describes such a web based system developed and in use regular use and found extremely useful.

OVERVIEW

Indus-1 and Indus-2, the Synchrotron Radiation Sources (SRS) at RRCAT Indore are national facilities operated round the clock to provide synchrotron radiations to users as well as carrying out machine studies. The concerned groups among which good information communication is considered essential include the beam line users, the operation crew, the machine sub-system experts and the management. Keeping sync among various groups is important and necessary for effective management of various activities and smooth running and utilisation of the facility. Large number of sub systems and frequent changes often required in various systems also call for effective monitoring of the system changes and communicating the same to all concerned. The Eplanner software was conceived to minimise various difficulties faced in day to day operation of this facility.

SYSTEM REQUIREMENTS

Based on the previous experience and the feedback received from various system experts and machine operation crew members following basic requirements were considered before developing the ePlanner software:

- (1) It should be possible to use the software by multiple users simultaneously connected over campus network.
- (2) The ePlanner should be easy to use for especially non computer experts.
- (3) Only authenticated & authorized users should be able to use the specified modules of ePlanner. However it should be possible to get the logged data without authentication in read only mode.

- (4) It should be possible to query & retrieve the specific information from stored historical data in chronological order.

SOFTWARE DESCRIPTION

Access to ePlanner has been protected with password so that only authenticated users could use the system. Each Section of ePlanner has different user group. System checks user's credentials using institute's central e-mail server. Hence users may use the software using their official e-mail login and password. This approach was adopted to avoid storing duplicate credentials for same person. Essential functions of ePlanner are depicted in Figure 1. ePlanner provides functionalities for Work Plan Management, Machine Shutdown Management, Beamline Booking Management, Standing Instructions Management, Electronic Notice Board (eNoticeBoard) and Information Display.



Figure 1: ePlanner Functions.

It is felt by Indus operation crew members that Indus Fault Logbook (FLogbook) which is a separate application used for tracking the details of faults occurring during round the clock operation of Indus-1 and Indus-2 should be merged with ePlanner so that all the information could be managed from a single software package.

Data Input

Using HTML & JavaServer Pages (JSP) ePlanner provides the form (having text boxes, check boxes, drop down list, etc.) for entering/logging the related information. Logged textual data with attached document (if any) may also be sent immediately to system persons concerned through e-mail. A unique log-id is generated

for each logged entry so that historical data could be diagnosed in future by its log-id.

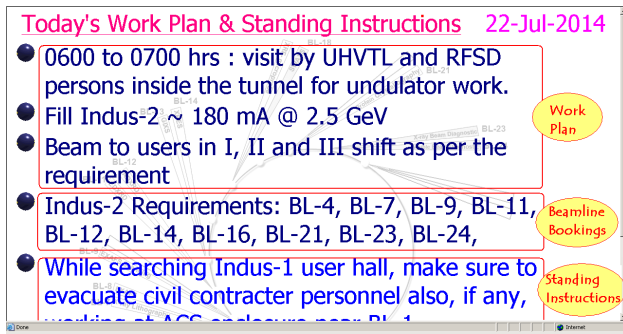


Figure 2: Information generated as per prescheduled work plan, beamline bookings and Standing Instructions.

Data Output

ePlanner provides query forms for retrieving the logged historical data in chronological order. Some customized web pages have been designed which display the auto refreshed data from different tables of ePlanner database. For example as shown in Figure 2, data are being displayed from work plan system, beamline booking system and standing instructions system. These pre planned information are date wise stored in ePlanner database by machine coordinators/shift in-charges, beamline users and system experts.

Feature has been provided by the software to log comments on selected specific historical faults and email it to selected concerned persons. If needed the designated authorized person may delete the unwanted duplicate entries within the specified time limit.

Software Design

Java Server Pages (JSP), JavaBeans and SQL databases have been used for designing & developing the various components of ePlanner. JSP with JavaBeans are used for developing the applications for web based environments. Java is main programming language used here for developing the complete ePlanner package.

SOFTWARE ARCHITECTURE

The ePlanner Software follows the three-tier software architecture for designing & executing its building blocks. Here Web Browser resides on client machine and work as first tier or client tier. In our working environment we mainly use Microsoft Internet Explorer (IE) for accessing the web sites and web based applications hence we have developed and tested the ePlanner for IE users. ePlanner uses JavaServer Pages (JSP) & JavaBeans for developing the presentation/view and business/application logic. JavaServer Pages (JSP) technology enables Web developers and designers to rapidly develop and easily

maintain, platform independent, information-rich, web applications that leverage existing business systems.

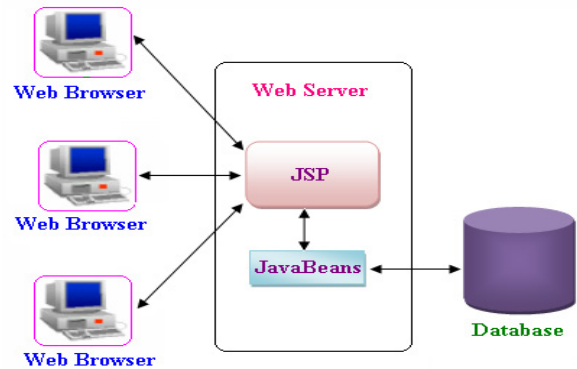


Figure 3: ePlanner Architecture.

JSP technology separates the user interface (content presentation) from content generation, enabling designers to change the overall page layout without altering the underlying dynamic content. Tags for content *access and presentation* reside in the webpage. Logic and programming code for content *generation* reside in reusable components. After receiving the client request, the JavaServer Page requests information from a JavaBean. The JavaBean can in turn request information from a database (Figure 3). Once the JavaBean generates content, the JavaServer Pages can query and display the Bean's content. JavaBeans components (beans) are reusable software programs that we can develop and assemble easily to create sophisticated applications.

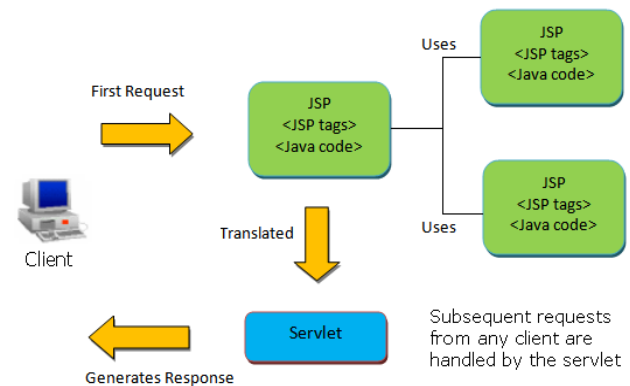


Figure 4: Execution of a JavaServer Page (JSP).

ePlanner uses JDBC (Java Database Connectivity) inside JavaBeans components for accessing the ePlanner database for inserting & retrieving the data.

The Type 4 Driver for JDBC has been used which provides JDBC access through any java-enabled applet, application, or application server. It delivers high-

performance point-to-point and n-tier access to SQL database across the internet & intranets.

The JavaServer Page is identified to the server by a *.jsp* extension; this tells the server that special handling is required. As shown in Figure 4, the first time a request is made for such a file, the *.jsp* file is translated to a servlet & compiled into an object. (For that reason, there can be a slight delay on the first request for a *.jsp* page.) The output from the object is standard HTML which the browser interprets and displays as usual. After compilation, the compiled-page object is stored in memory on the server. On subsequent requests for that page, the server checks to see if the *.jsp* file has changed. If it has not changed, the server uses the compiled-page object stored in memory to generate the response to the client. (Because the object is stored in memory, the response is very fast.) If the *.jsp* file has changed, the server automatically recompiles the page and replaces the object in memory.

We have used Apache Tomcat as a web server for executing & serving web components of ePlanner. Apache Tomcat (or simply Tomcat) is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Oracle Corporation, and provides a "pure Java" HTTP web server environment for Java code to run. Indus fault logbook (FLogbook) uses JavaMail API for sending the e-mails composed of the information logged by the operation crewmembers into FLogbook database. The JavaMail API provides a platform-independent and protocol-independent framework to build mail and messaging applications.

Microsoft SQL Server based relational database was designed to implement the data tier of ePlanner, which stores the complete information in related tables. Information related to work plan, beamline booking, machine shutdown, etc. are stored in its respective tables. Subsystem names of both the accelerators (Indus-1 & Indus-2) and concerned persons' details are configured in the application so that logged fault information or acknowledgement/comments could be mailed electronically using FLogbook.

SOFTWARE DEPLOYMENT

The ePlanner software has been deployed on Apache Tomcat web server configured on a separate server machine. This server machine as shown in Figure 5 is on a separate network and connected with firewall. Firewall is connected with accelerators technical network (AccNet), campus network (RRCATNet) and Large Format Display Network (LFDNet). Large Format Display Computers are connected with LFDNet and installed at various places of Indus premises for

displaying the current work plan, beamline requirement, standing instructions, notice, etc. The customized web pages have been developed for this purpose which retrieve the relevant information from ePlanner database and display (auto refreshed) on Large Format Display Computers (LFDs). The URL of the application has been mapped in DNS of the networks so that it could be accessed uniformly from the machines of all the networks. DNS Server of accelerators' technical network (AccNet) is running on Domain Controller (DC) Server machine.

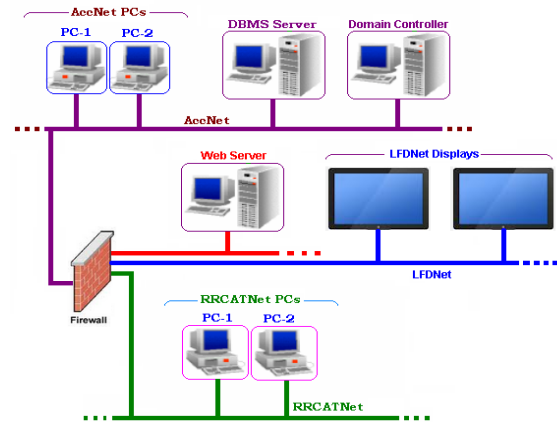


Figure 5: Software Accessibility over LANs.

CONCLUSION

ePlanner has been deployed in the field and being used regularly for managing and monitoring the various type of day to day activities needed in round the clock operation of both accelerators (Indus-1 & Indus-2). The system is very useful for not only the accelerators operation crew members but also for machine/subsystem experts, beamline users and general visitors. The complete software was developed following the modular approach so the future enhancements could be accomplished in minimum time.

ACKNOWLEDGMENT

This development represents the realisation of ideas of many people involved with its use. Authors acknowledge the active feedback and suggestions of the system experts involved in the regular Indus review meetings and Indus shift crew members. Support, encouragement and acknowledgement from senior management including Director, RRCAT has always been motivating.

RECENT HIGHLIGHTS FROM COSYLAB

M. Plesko, F. Amand, Cosylab, Ljubljana, Slovenia

Abstract

Cosylab was established 13 years ago by a group of regular visitors of the PCaPAC. In the meantime, it has grown to a company of 90 employees that covers the majority of accelerator control projects. In this talk, I will present the most interesting developments that we have done in the past two years on a very different range of projects and I will show how we had to get organized in order to be able to manage them all. The developments were made for labs like KIT, ITER, PSI, EBG-MedAustron, European Spallation Source, Maxlab, SLAC, ORNL, GSI/FAIR but also generally for community software like EPICS, TANGO, Control System Studio, White Rabbit, etc. And they range from electronics development to high level software: electric signal conditioning and interfacing, timing system, machine protection system, fibre-optic communication, Linux driver development, core EPICS development, packaging, high performance networks, medical device integration, database development, all the way up to turnkey systems. Efficient organisation comprises a matrix structure of teams and groups versus projects and accounts, supported by rigorous reporting, measurements and drill-down analyses.

INTRODUCING COSYLAB

In the course of the writing of this paper we received final confirmation of a 2.5 million CHF project for Cosylab on the SwissFEL at PSI, Switzerland. Great news for us and a confirmation of Cosylab's growing involvement in the control system work on cutting edge Big Physics machines around the globe. In this paper we will start by highlighting a few particular projects we are taken on as we speak with a turnkey approach. We will then zoom in on project management aspects that are key to keeping such endeavours on track and delivering on the promise. We will finally look at operational aspects, how Cosylab is internally organized, as the organization is tailored to nature of the projects, their size, as well as our current size.

ELI-NP

The Extreme Light Infrastructure (ELI) [1] currently consists of four projects that will provide a great platform for the study of the fundamental processes that unfold during light-matter interactions. One of them is ELI-NP (Nuclear Physics) in Magurele, Romania.

Once built, the ELI-NP will be the most advanced laser and gamma beam facility in the world. It features Cosylab's first true turnkey control system, designed as such

from the ground up. Coordinated efforts towards turnkey solutions have existed for several years and are being applied by Cosylab to other projects – most prominently for the current projects for ESS and MedAustron.

SOLARIS

Solaris is a project of the Polish Synchrotron Consortium (comprising 35 research institutes and universities). It received financing from the European Structural Funds, and is being constructed as we speak, with first research planned for 2015.

Specific to this project is a strong partnership with the MAX-IV project [2] in Lund, Sweden. The design of the synchrotron ring aims at maximizing reuse of the design of the 1.5 GeV storage ring of MAX-IV. This has repercussions on the design of the control system: it is based on the TANGO control system [3], chosen by MAX-IV.

Cosylab has been selected for the delivery of the control system integration services, including delivery of the timing system hardware. With this choice the Solaris team has opted not to reinvent the wheel on control system integration, just as they chose not to reinvent the storage ring design.

PROJECT MANAGEMENT ASPECTS

Incomplete Requirements

The incomplete requirements, that are so typical of ground breaking and pioneering big physics projects, including the control system part, have a tendency to stall projects before they have even started.

We have a strong philosophy about incomplete requirements, namely to accept them as a fact of life, rather than to fight them (by insisting on and waiting for clearer requirements, hence: stalling). We rather start with designing on a “straw man” design for the control system (with the requirements we have) and develop an early vertical prototype. As such we are pro-active in eliciting the actual requirements, because seeing physical things makes people think much more.

Stalling or over-analysing instead makes little sense, because no complex development ever follows the simple “waterfall” scenario with the phases like requirements, design and implementation following linearly one after the other only once the previous is finished. In reality, some requirements are dropped during the way, new come in, and what's more, development cycles often follow a spiral and pass several times through all the phases.

#80855: ITER: WO29 Drivers

LastEntry · Progress · Parent Time · Gantt · ProjectReport

Open · Steal ... Reply · Resolve · Meeting time

Ticket metadata

The Basics

Id: 80855

Status: new

Priority: 15

Queue: ACC-ITER-WO29_Drivers

Custom Fields

Ticket Type: (no value)

Feedback: (no value)

Severity: (no value)

ProcessType: (no value)

People

Owner: @cosylab.com>

Requestors: @cosylab.com>

Cc:

Project Manager: @cosylab.com>

Dates

Created: 2012-07-19

Starts: 2012-07-16

Started: Not set

Last Contact: 2013-03-27

Due: 2013-01-31




Closed: Not set

Updated: 2013-03-29 by

Links

Depends on:

Depended on by:

Hierarchy:   

80855: ITER Master ticket [new]

80856: Management () [resolved]

80857: Bugfixing () [resolved]

80858: Test cases () [resolved]

80859: Documentation () [resolved]

80860: QA () [resolved]

Contract task: cut the project in subprojects

Figure 1: Example of a Master Ticket, representing a project in the RT Tool, Cosylab's planning, tracking and reporting tool.

Planning, Tracking & Reporting: RT Tool

Cosylab has taken an issue tracking system, called RT, and extended it into a full project management planning and tracking tool. This year, our system celebrated its one hundred thousand and first ticket, to show on what scale and intensity it is used. As software gurus would say, "everything is an object/actor/" At Cosylab EVERY activity or task, is a ticket. And every such ticket = task is typically a part (a child ticket) of a project task and the project task a part of the overall project.

1.1. Time Project budget

Project size: 17.81 md : 3.56 mw : 0.69 mm
 Time spent: 9.21 md : 1.84 mw : 0.46 mm (51%)
 Time spent (4420/8550)
 Estimated time to finish: 2.06 md : 0.41 mw : 0.10 mm

Project History

Spent time (red line), Planned time (blue line)

Figure 2: Project Tracking with RT Tool.

Planning is worthless if it is not followed up with meticulous tracking of work items to assess the overall status of the project and in-depth analysis of individual causes of delay. RT has features to assist these essential processes within Cosylab.

Certified Processes: ISO 9001:2008 and Medical ISO 13485:2003

It's important to do things in the right order: if you design processes to deliver traceable quality, then preparing for ISO certification and renewal becomes a low hurdle, you merely illustrate from the evidence in the IT system, that you are doing what you are saying. The certification is "just" an external confirmation of an internal awareness for quality. Of course, when collaborating on the medical front-end of the machine, such as in MedAustron's dose delivery system, medical ISO certification becomes a condition sine qua non.

HOW COSYLAB IS ORGANIZED

To explain how Cosylab is structured to optimally take on big projects, it is best to start by describing the different job positions, functions and roles our people will be taking on.

Positions are full time job descriptions, *functions* are additional responsibilities assumed by an employee and *roles* are technical roles in the context of a specific project (e.g. software QA)

Management Software Projects

ISBN 978-3-95450-146-5

133

Copyright © 2014 CC-BY-3.0 and by the respective authors

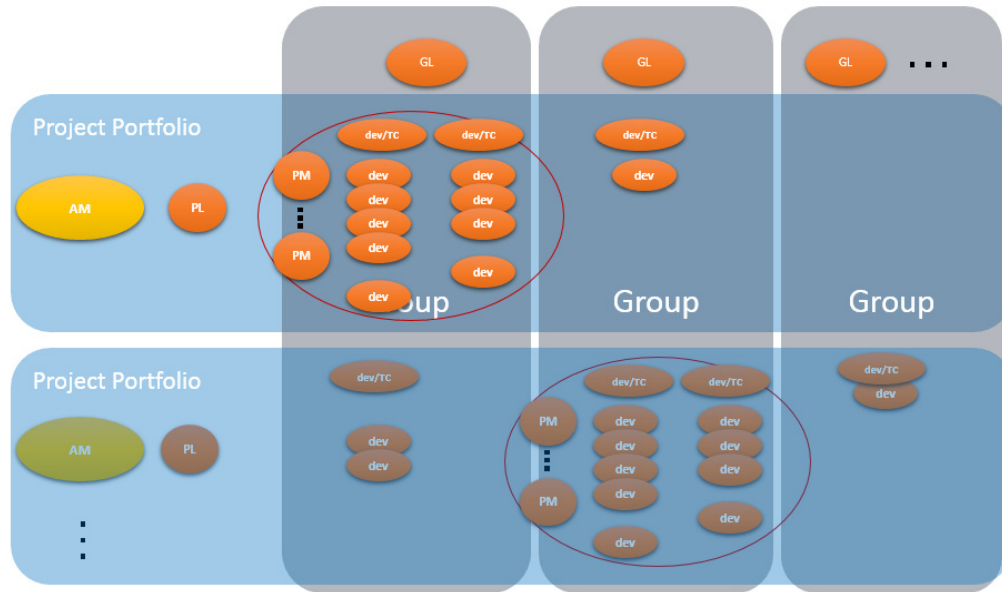


Figure 3: The matrix organization with Human Resources/Teams versus Projects/Accounts axes.

Technical Experts: Rookie, Junior, Senior, Guru Developers

This positions comprise the majority of the Cosylab workforce, they are the experts that bring the control systems to life. Becoming guru in our highly specialized domain takes years, even for the most talented, so there are distinct stages to get there, starting out as Rookie when you freshly join.

Team Coach

This is the function that is the guardian of a particular technical expertise, like FPGA Programming, EPICS Device Integration or Control System Application Development. He/she makes sure his team builds up the necessary expertise and keeps it up-to-date. They are the mentors on the various Cosy Academies. Having a full workload of relevant work for his team is the best training, so he is also a liaison between the group leader and the developers to keep their individual “order book” full and in-sync with the project’s overall work needs.

Group Leaders

The group leader, or GL is the actual “boss” or resource manager of the people in his group. He is also responsible for the strategic direction of his group and appropriate staffing, i.e. he is responsible for hiring people to his group.

PMs, PLs and AMs

These roles form the horizontal, project side of the matrix organization (see further). Project Managers manage sub-projects in one technology. Project Leaders are project portfolio managers, they manage a turnkey project as

a whole. Account Managers take care of the customer facing. In a young fast growing organization of around 100 people, where multi-year projects are all in different stages, the same people combine a position with several functions and/or roles. Nevertheless they, and the rest of the organization are aware of the different “hats” and clearly distinguishes between them, depending on the context. It requires some flexibility, agility from both sides, yet it is highly transparent and efficient.

CONCLUSION

Cosylab has grown quite a lot in the past years and as we have grown, so has our competences and capacities to offer and build a control system at a fixed price. Not a platform but a solution with responsibility for control system up-time. The control of construction cost and quality are a result of Cosylab’s efficient organisation, supported by rigorous reporting, measurements and drill-down analyses. Currently major control system integration work is ongoing around the world at SLAC, GSI/FAIR, ESS, ITER and others. These institutions put their trust in Cosylab to deliver on the promise.

REFERENCES

- [1] <http://www.eli-laser.eu/>
- [2] <https://www.maxlab.lu.se/maxiv>
- [3] <http://www.tango-controls.org/>

MANAGING THE FAIR CONTROL SYSTEM DEVELOPMENT

Ralph C. Bär, Frédéric Ameil

GSI Helmholtz Center for Heavy Ion Research, Darmstadt, Germany

Abstract

After years of careful preparation and planning, construction and implementation works for the new international accelerator complex FAIR (Facility for Antiproton and Ion Research) at GSI have seriously been started. The FAIR accelerators will extend the present GSI accelerator chain, then being used as injector, and provide anti-proton, ion, and rare isotope beams with unprecedented intensity and quality for a variety of research programs.

The accelerator control system (ACS) for the FAIR complex is presently being designed and developed by the GSI Controls group with a team of about 50 soft- and hardware developers, complemented by an international in-kind contribution from the FAIR member state Slovenia.

This paper presents requirements and constraints from being a large and international project and focusses on the organizational and project management strategies and tools for the control system subproject. This includes the project communication, design methodology, release cycle planning, testing strategies and ensuring technical integrity and coherence of the whole system during the full project phase.

FAIR PROJECT

FAIR, the new Facility for Antiproton and Ion Research is a new international accelerator facility for the research with antiprotons and ions. It is being built at GSI in cooperation of an international community of countries and scientists. The accelerator facilities significantly extend the present GSI accelerator complex, then being used as an injector for the FAIR machines.

In October 2010, nine countries signed the international agreement on the construction of FAIR under international law. FAIR will be financed by a joint international effort of so far 10 member states. The Federal Republic of Germany together with the local federal state of Hesse will provide the major part of the budget. International partners in Europe and overseas will substantially contribute as well, about 30% of the construction costs, some of them already being shareholders of FAIR. The countries will contribute both in kind, by supplying facility components, and in cash.

FAIR will be realized in several modules. The funding for the modularized start version, 1 billion Euro in 2005 prices, has been acquired. The FAIR start version [1] comprises the superconducting SIS-100 ring, CR, HESR, SFRS, Proton-linac and about 3.5 km of beam line.

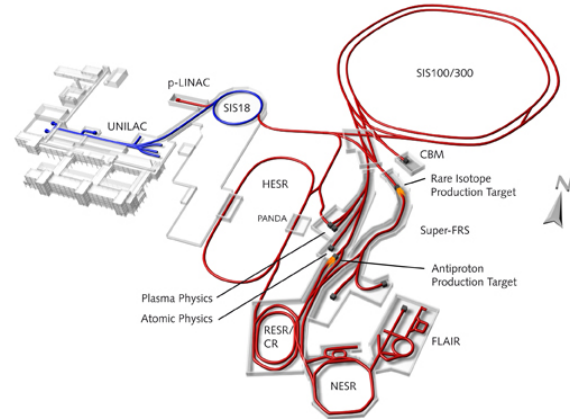


Figure 1: Schematic overview GSI (blue, existing) and FAIR (red, to be built) accelerators and beam lines.

FAIR PROJECT MANAGEMENT

Considering the substantial scope of the FAIR project (with its three sub-projects accelerators, experiments and civil construction), the technical and organizational complexity as well as the high financial investments needed, the FAIR council and mainly the German funding agency has required for a high and professional standard project management.

Consequently, in the past years GSI has established a project organization and management that is adequate for a project of this dimension. In 2012 GSI has been completely reorganized in order to fully focus on the FAIR construction phase. Besides the line-hierarchy, a matrix-like project organization was introduced:

- Project Leader FAIR@GSI (PL)
- Machine Project leaders (MPL)
- Work Package Leaders (WPL)

For the accelerator subproject, a Project Leader (PL) was appointed and a Project Management Office was built up. For the individual machines (e.g. SIS-100, CR, HESR, etc.) 7 Machine Project Leaders were introduced. For the technical subsystems (e.g. power converters, RF systems, vacuum, beam instrumentation, magnets, etc.) about 100 Work Package Leaders (WPL) have been installed and assigned to the respective machines.

Integrated Project Planning

To manage time schedules and resources of the GSI contribution to FAIR, Integrated Project Planning group has been established and project planning tools introduced and customized for FAIR. For resource-loaded project planning, an MS Project 2010 server environment has

been set up and is intensively used. Project plans for FAIR are organized in 3 levels (see Figure 2):

- Level 1: Master Schedule for the Accelerator Subproject (all machines)
- Level 2: Major Milestone Plans for every FAIR Machine or singled out Major Technical System
- Level 3: Detailed Plans for every work package

Plans are loaded with resources (financial data and generic type human resources, e.g. engineers) and are linked with each other by either hard-links (milestones will move automatically) or soft-links (information on moved milestones only). Links allow identifying major delays in the whole project, establishing the critical path and helping to adjust the resources in order to optimize with the actual capacity. In order to handle the complexity, certain rules for links have been introduced.

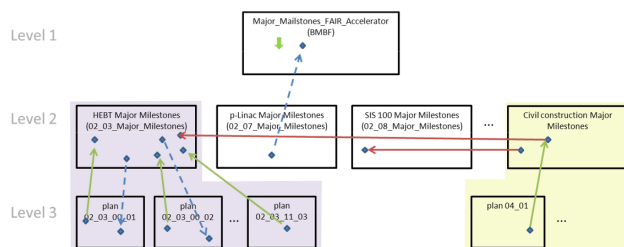


Figure 2: Hierarchy of project plan and links.

CONTROL SYSTEM ISSUES

Since this paper is not intended to describe general FAIR project planning activities we will focus on aspects of the control system development.

In-kind Contribution

For many reasons the ACS for FAIR has been declared as an in-kind contribution to be provided by the FAIR host-lab, GSI. However, about 20% of the ACS cost book value will be complemented as an in-kind contribution by the member state of Slovenia. ACS subprojects with clearly defined interfaces have been mutually agreed to be contributed by developers from the Slovenian consortium. Although based on detailed technical specifications, working out concepts and implementation requires a high level of communication and familiarity with the general ACS architecture and GSI development team. The approach to have a permanent Slovenian project manager/developer resident at GSI has turned out to be a very successful model. Regular (weekly) Jour-Fixe meetings with the ACS project lead team are established to discuss problems, progress and clarify open questions.

Organization

While all technical systems are usually assigned to a specific FAIR accelerator, the ACS is considered an orthogonal aspect to the standard matrix organization. Reason for this is that the ACS is considered a “Common System” for all machines, i.e. there is only one ACS for

all GSI and FAIR accelerators. Consequently, in project management, the ACS development is considered an accelerator-like project, is managed by a MPL and is represented by a Major Milestone plan (see Figure 2) that in turn has 10 level 3 plans for

However, since project meetings and communication is usually organized “per machine”, the role of Machine Controls Coordinators (MCC) have been introduced in the controls group. A MCC serves as a contact person for all matters concerning a specific FAIR machine and organizes activities within the controls group.

Design Team

During the design and implementation of a large and complex soft- and hardware project special attention must be paid that a clean design and system architecture is being worked out and followed during implementation. We decided to establish a Controls Core Team (CCT) to create, continuously refine and modify functional specifications and the architectural blueprint of the ACS. This team, constituted of 4 senior developers, ensures the integrity and coherence of the whole system during the full project phase, defines architectural principles and provides formal and technological guidelines wherever necessary.

Project Planning

The development effort of the ACS covers many aspects of hardware and software design and about 50 engineers, scientist and technicians are working presently in the control system group in GSI.

Such a large and complex development project may lead to a too high complexity regarding planning. Thus for practical reasons project planning for the development of the ACS is being done at two different levels of detail. The project planning for development of the ACS needs indeed to target two objectives.

- Integrated planning for each Work Package for the full project period (2013 to 2020) required by the project coordination.
- Detailed planning for all activities

In order to keep the integrated project planning complexity as low as possible the planning of the development of the ACS is organized following an iterative approach. Thus a release chain was introduced with two major releases per year with fixed due dates and a content being adapted to the overall FAIR machine schedule, in other words time schedule remains and content is modified. This cycle length of half a year was considered as optimal.

Then needed milestones for each machine are derived from the release chain and provided for other plans in order to link against them. Furthermore this allows a flexible planning especially considering uncertainties in the FAIR civil construction work.

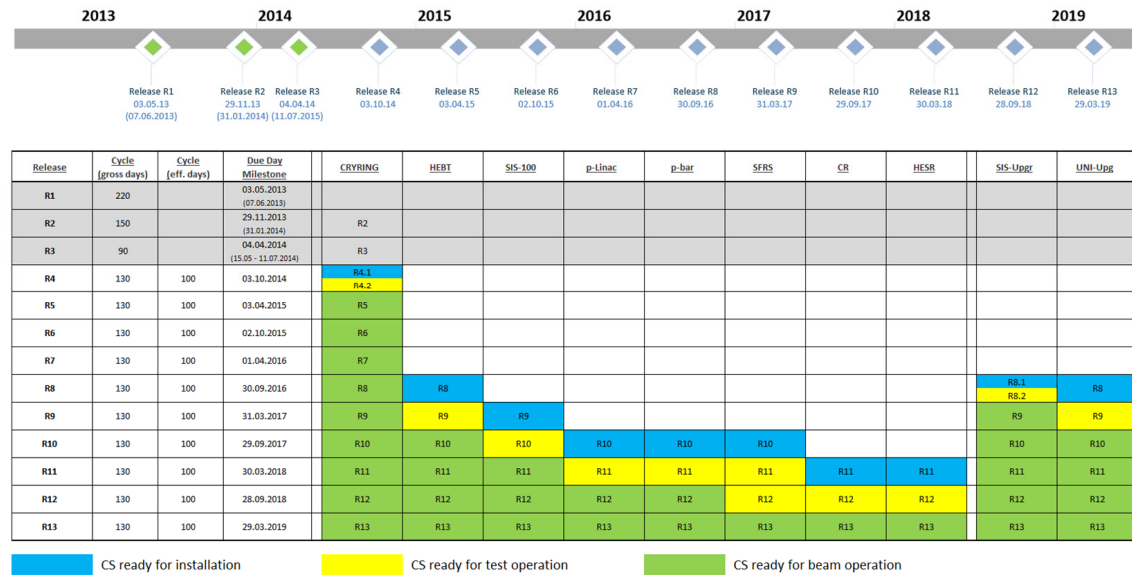


Figure 3: Schedule and release chain until 2019/2020.

Beside this a detailed planning is performed. This is done one release ahead, a detailed planning over a longer period being not realistic. This planning is activity based detailed resources loaded. All required activities for the achievement of the release content are planned and loaded with necessary resources.

Such an iterative and two stage approach represents several advantages both for the development and the planning aspects:

- The system functionality can be extended from release to release (iterative development paradigm)
- Avoid an unrealistic and unmanageable planning for the whole project
- The overload of the resources capacity can be easily avoided
- Tracking of progress allows project to be steered (e.g. modify priorities, move functions to later releases, etc.)

DEVELOPMENT METHODOLOGY

Based on the main requirements for the FAIR control system, an overall system architecture have been set up and a vertical slice through all layers of the control system has been built with basic functionality. Then, in further iteration cycles, components are replaced by newer versions, implementing more functionality, and additional components are added to the base system. By these iterations, the ACS will be extended to the full specified functionality.

In each single iteration cycle leading to a release as described above, requirements and designed functionality can be refined and adjusted, the functionality is implemented and the achieved components as well as the overall functionality are tested. Iteration holds for the system as a whole as well as for single components.

TEST STRATEGY

Presently, the heavy-ion storage ring CRYRING, being a Swedish in-kind contribution to FAIR and before decommissioned in Stockholm, is being installed behind the existing GSI Experimental Storage Ring (ESR) [2]. CRYRING can decelerate, cool and store heavy, highly charged ions that can come from ESR down to a few 100 keV/nucleon. It is equipped with its own injector line that will even allow CRYRING operation while the full GSI accelerator chain is shut-down until mid of 2017 for necessary FAIR upgrade and civil construction works.

As CRYRING has been dedicated as a test ground for the FAIR accelerator control system (as well as for a variety of other technical subsystems), the next step is to set up the control system for re-commissioning CRYRING in the next months to come. The main intention is to test and validate fundamental concepts, technologies and gaining experiences under real conditions in order to identify possible design flaws, limitations, and assure the quality of the control system components involved. While in the beginning the control system and its building blocks will only provide basic features, the intention is to add more and more functionality in the next control system releases to come.

REFERENCES

- [1] O. Kester et al., "Status of the FAIR Accelerator Facility", WEPRO060, IPAC 2014, Dresden, Germany.
- [2] F. Herfurth et al., "The Low Energy Storage Ring CRYRING@ESR", THPM1HA01, COOL2013, Murren, Switzerland.

STATUS OF INDUS-2 CONTROL SYSTEM

Pravin Fatnani[#], Anurag Bansal, Rajesh Kumar Agrawal, Kirti G. Barpande, Amit Chauhan, Sampa Gangopadhyay, Pankaj Gothwal, Ashesh Mangaldas Gupta, Musuku Janardhan, Bhavna Nitin Merh, Rohit Mishra, Chandrashekhar Purushottam Navathe, Kutubuddin Saiffee, M Seema, Yogendra M Sheth, Bakshi Sanjai Kumar Srivastava, Rishipal Yadav

Raja Ramanna Centre for Advanced Technology (RRCAT), Indore, India

Abstract

Indus-2 is a 2.5 GeV Synchrotron Radiation Source at Indore, India. With 8 beamlines operational, several more under installation & commissioning and 5 insertion devices planned, the machine is operated in round the clock mode for users. With implementation of orbit, tune and bunch feedback systems and many new systems in planning, machine is constantly evolving and so is the control system. The control system software is based on WINCCOA SCADA running on windows PCs and also integrates with other software modules in Labview and Matlab. The control hardware is a combination of VME based control stations interconnected over Ethernet and Profibus. Some recent system enhancements include parameter deviation alarms, transient data capture system, database improvements and web services. Paper takes a stock of the control system and its evolution along with new systems recently introduced.

INTRODUCTION

Over the years, Indus-2 has been consistently graduating to newer milestones. Among various other enhancements, these relate to addition of new front ends and beam lines, regular operation of various beam lines, round the clock operation of machine, improvement of orbit stability through implementation of orbit control systems, tune feedback system to stabilise the machine tune points, improvements in ramping and cycling procedures, faster data logging, automation and monitoring of various auxiliary systems like LCW, compressed air system, cavity precision chillers, solid state RF amplifiers, addition of vacuum chamber temperature alarms, bunch by bunch feedback system to counter instabilities at higher beam currents, addition and remote operation of diagnostic beamlines, machine and sub-system diagnostics etc. Now the machine operates with global slow orbit feedback (SOFB) system controlling the orbit to within 30 microns. With tune feedback system implemented, bunch by bunch feedback system integrated with the control system, both local and global fast orbit feedback (FOFB) systems demonstrated, the system is now being prepared for regular use of combined operation of SOFB along with FOFB. Average horizontal orbit drift correction using RF frequency correction will also be integrated. Two undulators will be received and installed soon.

The preparations for system integration of undulators and its control system with machine control system are on the anvil. The control system for Indus accelerators is designed, developed, and maintained by the Accelerator Control Section, RRCAT.

OVERVIEW

The control system works on client server model and enables functional and physical separation and placement of hardware and software modules across the entire range of control system components. Using WinCCOA [1] SCADA on Windows client and server machines interconnected over ethernet switched network to two layers of VME controllers via Profibus, the distributed Indus-2 control system (Fig.1) monitors about 10000 I/Os in all. User Interfaces are mostly built around the SCADA for managing the complex requirements in an integrated manner. The intermediate, supervisory layer and the lower equipment interface layers are based on VME controllers running a multitasking real time operating system. Ethernet (100 MBPS) and PROFI bus (750 K baud) communications are used between L1-L2 and L2-L3 respectively. The modular control system hardware is designed around VME bus.

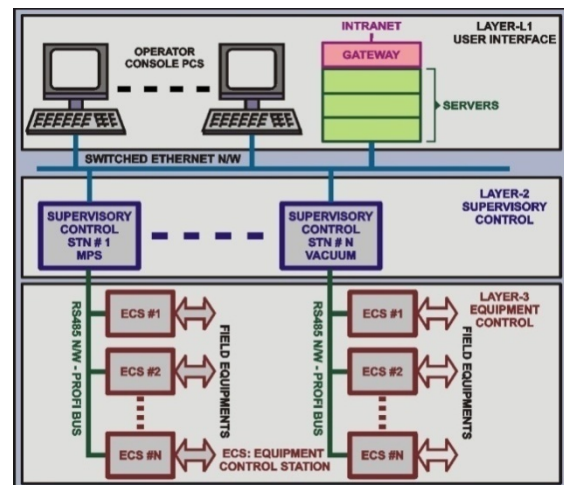


Figure 1: Indus-2 Control System Architecture.

Various enhancements in the individual control systems leading to performance improvements in past few years and in recent times are briefly described below.

EVOLUTION

Machine Safety Interlock System (MSIS)

MSIS monitors about 400 critical signals of machine components and takes action during abnormal and potentially harmful conditions to safeguard the machine. It takes decisions for allowing different power supplies to run, injection to happen, or trip the power supply or kill the beam by tripping the RF system. The system ensures that all safety shutters are closed at the time of injection, all valves are open, water flow is normal and temperatures at various locations are within range. It also takes action if any of the assigned beam line front end signals have abnormality. It also takes care for the conditional bypassing of any faulty status or device. It is built using home grown VME infrastructure and carefully designed to be reliable and fail-safe. It is a distributed system built using VME controllers but without the Profibus for interconnection on RS 485 bus. It rather uses specific custom communications for tighter performance and guaranteed reliability. Many signals from beam line front ends were later included with MSIS for improved safety of beam line components and machine. This system is runs in 24x7x365 mode.

Central Alarm Handling System

Alarm handling system for Indus-2 keeps watch on all machine parameters and raises alarms whenever abnormal conditions are detected. Alarms have been categorized into two categories, normal and critical which have distinctly different audio annunciation tones in the control room. Recently hysteresis has also been introduced to alarm values. Towards the policy of empowering the sub-system experts to themselves set and change the system parameter limits, an alarm configuration module is now made. System experts can now manage the parameter limits on their own.

Parameter Deviation Alarms

Proper machine operation depends on complex relation of a number of online alterable critical parameters which should remain within some tolerance limits. Therefore need was felt for a Parameter Deviation Alarm System which would raise alarms in such well defined cases. PDA has been implemented for magnet power supply parameters.

System Diagnostics

Extensive system and data diagnostics are made available which help to minimize the machine down time by quicker diagnostics of system malfunctions. System diagnostics built from the initial days are - L3 and L2 layer controller status, Bus-error status of L-3 CPU, L-3 CPU running state, L1-L2 communication status, API running status, DAC readback by ADC for end-to end

reference signal confirmation (in MPS system) and board temperature info for high stability analog I/O boards. Later application level diagnostics facilities were also developed, like, cycling analysis and verification system, ramping data capture and analysis system (Fig.2) using fast data capture through onboard ADC and local data storage, transient data capture system using separate faster ADC modules and web based beam trip analyzer etc.

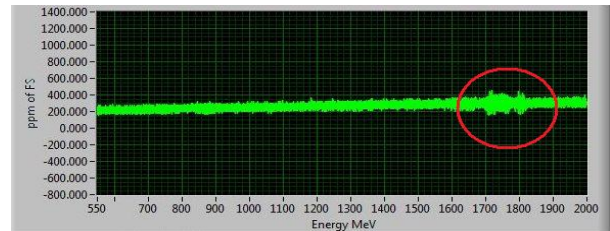


Figure 2: Ramping data capture system showing up malfunction at 100 PPM level in high accuracy power supply readback signal.

SCADA - WINCCOA

In the three-layered architecture of Indus control system, the WINCCOA API manager [2] plays a crucial and interacts with both the WINCCOA database and L2, the VME Supervisory layer. In order to access the PVSS database it uses the PVSS API, a C++ class library, whereas in order to access the L2, custom functions have been built. Any external software can be integrated in a PVSS System via class libraries provided by the API. API managers have been developed for various sub-systems of Indus-2. These cater to multiple clients by following a multi-threaded design. As the work progressed, the developed API managers were put to real test. The system did require changes and fine tuning to cater to new requirements and to remove some bugs. Some operations, like ramping clock generation, earlier done using panel scripts were later moved to API. For a special requirement like energy ramping, the magnet power supplies API manager code was modified such that it retains the state of the last operation. The various states maintained are INIT, RAMP-ON, RAMP-PAUSE, RAMP-RESUME and RAMP-OVER. So even if the API Manager is re-run during ramping process, it does not affect the clock generation and ramp operation.

Timing System

Timing system controls every phase of the electron beam extraction and injection through various machine components, like Microtron, Transport line-1, Booster, Transport lines 2 & 3 and Indus-2. It provides for proper bunch filling modes, control of various trigger pulses, generating the required delays and amplitudes of pulsed power supplies etc. Since booster (31.613 MHz) and Indus-2 (505.6 MHz) rf frequencies are different, a coincidence generator board was designed and used for proper injection into Indus-2. The bucket separation in Indus-2 ring is 1.99 ns and delays are generated with sub-nano second resolution and jitter less than 2 ns. Noise issues initially faced with the timing system were

overcome by re-design of some boards and avoiding ground loops through use of optical fiber based analog reference signal delivery system. Initially three bunch filling modes were built but recently new bunch filling patterns were required and these have been provided now.

Data Logging

About 10,000 machine parameters are continuously logged in the central database. Logging of all operator interactions and system events is also done, which helps in correlating the information that is crucial in case of system malfunction. The data logging system in use till recently made SCADA write directly to MS SQL database. It supported logging at varying time intervals of one second to one minute. Although this strategy worked satisfactorily till some time back, it did not scale up well as the number of control parameters increased and began to strain network and storage capabilities. So the data logging system has been upgraded [3] and now it supports logging of all the machine parameters at the same rate at which all the parameters are acquired from the field i.e. at 1Hz.

The new data logging scheme adopts 'data table per data type' approach [4] i.e. same data-type parameters values are stored in a single table. Besides this, in this new data logging scheme [5], the time series data are first put into text files by the SCADA and later bulk inserted into the data base using Java programme and stored procedures. Table partitioning using sliding window scenario concept [6] is an important ingredient. The Java application also manages temporal synchronization and serve as watchdog for any application failure. With these and associated major modifications at software level, the system achieves fast data logging of all the parameters at 1 Hz.

Web Based Information Management Tools

Some very useful and convenient web based tools have been developed and deployed. Indus Online provides the live, historical and statistical data to users and system experts, Fault Information System allows tracking and emailing the faults occurring in sub-systems, Machine Status Information System is used to display the live machine status at various locations of Indus Complex, Flogbook [7] helps logging and emailing the machine faults by shift crew and solutions and comments by system experts. Elogbook allows recording of shift operation details in electronic form. Eplanner is a package extensively used for machine activities management.

Orbit Feedback Correction Systems

Two orbit correction schemes are being implemented. The *global Slow Orbit Correction (SOFB) Scheme* has been implemented. It brings down the natural occurring beam variations in both horizontal and vertical planes to within $\pm 30\mu\text{m}$. The *global Fast Orbit Feedback (FOFB)* system is being implemented in phases. In its first phase the FOFB system with 16 BPIs and 16 fast correctors is

developed for both the planes. In the second phase this scheme will be extended to the full 56 BPI and 32 corrector version for both the planes. FigureXX gives the block diagram for global FOFB system at Indus-2. The system successfully brings down the natural occurring beam variations up to 50Hz in Vertical plane to $\approx \pm 3\mu\text{m}$ (Fig.3).

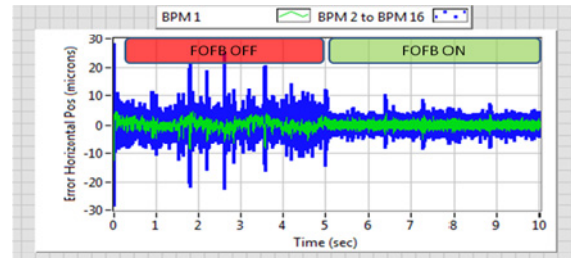


Figure 3: Measured beam position on different BPIs with Global FOFB system OFF and ON for vertical plane.

SUMMARY AND FUTURE PLANS

Indus 2 control system continues to evolve to support the ever growing new requirements and enhancements and thereby support the evolution of this national facility. API managers have been running with no crash events being reported. The load of the over all SCADA system has been nearly constant and lies between 17-21% with API manager load being maximum 2%. Possible extensions to the API are developing a generic API which caters to all future up-gradations of the Indus control system. In near future, the system will see development and integration of controls for insertion devices and beam based alignment systems.

REFERENCES

- [1] WINCC OA is a SCADA package from Siemens.
- [2] Bhavna N Merh, "API Manager Implementation & its use for Indus Accelerator Control", 9th Personal Computer and Particle Accelerator Conference, 2012, Kolkata, <http://www.jacow.org>
- [3] Rohit Mishra et al., "Data Logging System Upgrade For Indus Accelerator", 9th Personal Computer and Particle Accelerator Conference, 2012, Kolkata, <http://www.jacow.org>
- [4] R. Billen, C. Roderick, "The LHC Logging Service", UKOUG Conference 2006 14-17 November 2006, Birmingham, UK
- [5] E. Goman, S. Karnaev, O. Plotnikova, E. Simonov, "The database of the VEPP-4 Accelerating facility parameters", Proceedings of PCaPAC08, Ljubljana, Slovenia, <http://www.jacow.org>
- [6] "Partitioned Tables and Indexes in SQL Server 2005" [http://msdn.microsoft.com/enus/library/ms190787\(v=SQL.90\).aspx](http://msdn.microsoft.com/enus/library/ms190787(v=SQL.90).aspx)
- [7] BSK Srivastava, "Flog Book: Concept to Realisation", 9th Personal Computer and Particle Accelerator Conference, 2012, Kolkata, <http://www.jacow.org>

FIRST OPERATIONAL EXPERIENCE WITH THE !CHAOS FRAMEWORK

C. Bisegni, P. Ciuffetti, G. Di Pirro, L. Foggetta, F. Galletti, R. Gargana, E. Gioscio, G. Mazzitelli, A. Michelotti, A. Stecchi, INFN-LNF, Frascati, Italy
L. Catani, INFN-Roma Tor Vergata, Roma, Italy

Abstract

The !CHAOS framework for control systems has been designed for a wide range of different applications in terms of performance, size and complexity of the system to control and the technologies used for implementing its services. Sub-components and core-services of the !CHAOS framework have been already tested in control applications and are currently in use as part of control systems of accelerators at LNF.

Recently, the !CHAOS framework has been used for upgrading the control system of the DAFNE Beam-Test Facility (BTF), a part of the DAFNE accelerator complex at Laboratori Nazionali di Frascati.

This document presents the development and the first operational experience of the BTF control system setup, the solutions adopted to fix bugs and optimise the performance and reports on the current stage of development of the !CHAOS framework.

INTRODUCTION

One of the pillars of the !CHAOS framework is the high abstraction of services, devices and data such to obtain a high scalability of the system and an extreme flexibility in terms of its possible applications (see Fig. 1).

Two main technologies are at the basis of the !CHAOS framework: the non-relational databases known as key/value database (KVDB) and distributed memory object caching systems (DOC).

The distributed memory object caching systems provide in-memory key/value store for small chunks of frequently requested sets of information. Main features is the fast response to clients' requests because the load is distributed to a scalable cluster of cache servers.

The key/value database technology, compared to relational databases (RDMS), offers high throughput, scalability and flexibility.

By implementing these two technologies the !CHAOS framework has been designed as a collection of services for clients that either need to publish and store the data they produce or need to read data and information for displaying it or for running calculation such as measurements or feedback algorithm.

While this solution might look as quite conventional for the DAQ service, where the KVDB is a simple replacement of a relational database, it is not customary that a DOC can be used as a mean for the real-time flow of data from the front-end controllers to the display clients or feedback

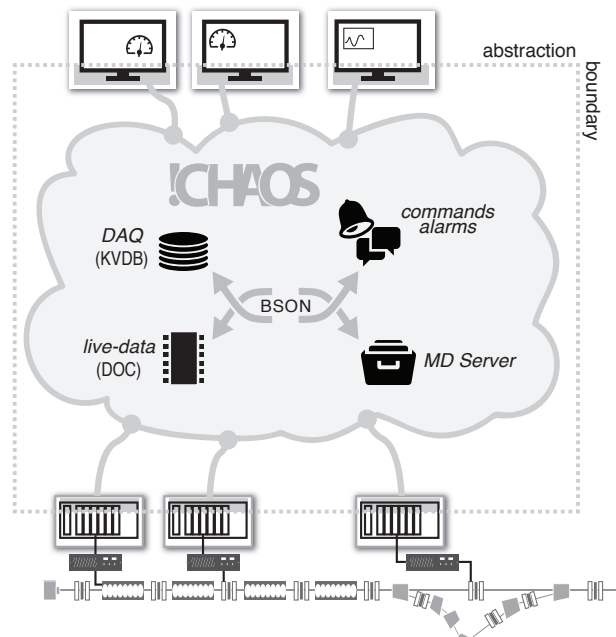


Figure 1: The !CHAOS model of control system.

software, in other word as alternative to the commonly used client-server communication.

Indeed, in !CHAOS data flow between front-end controllers and operator's consoles is managed by the DOC service that first receive the device data from the front-end controller and then provide the same data to any display application that need it.

In spite of its centric topology the !CHAOS services can be easily scalable for sizing the performance to the required throughput and to avoid any bottleneck by distributing a single service over several computers. Moreover, automatic failover is possible by redirecting to other servers the workload of a failed one.

By taking advantage of this feature the !CHAOS control system can be easily scaled according to both the different size of the accelerator infrastructure and the performance required, thus avoiding any potential bottleneck that may be expected as the weakest link of the star-like communication topology.

The data-pushing strategy allows to further extend the abstraction boundary at the front-end. The Controller's development has been simplified and standardised by introducing the *Control Unit*, a manager and a supervisor of the software modules implementing the device's specific control procedures.

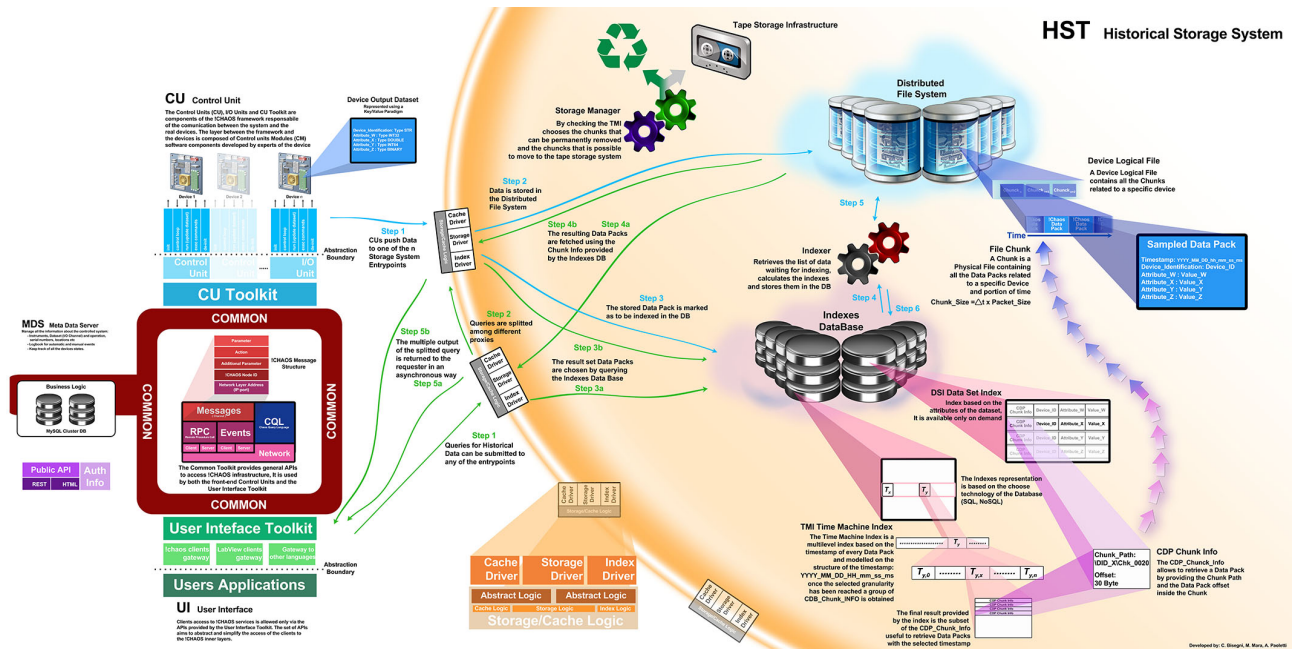


Figure 2: Overview of the !CHAOS Scalable Information Service (SIS) and Scalable Data Service (SDS).

CURRENT STATUS OF DEVELOPMENT

During the last months the !CHAOS group activities focused on three main tasks:

- design and implementation of !CHAOS Scalable Data Service (SDS)
- development of control system for upgrading the controls of the DAFNE Beam-Test Facility
- design of a non-HEP applications of !CHAOS

Chaos Data Service

The current version of the !CHAOS framework offers two main services:

- Scalable Data Service (SDS) which implements the live and historical data storage. The Scalable Data Service includes the distributed shared memory where the Control Units (CU) push data to and the clients fetch data from.
- Scalable Information Service (SIS), designed for the management of meta-data;

Presently there is a simplified version of the SIS — named Meta Data Server (MDS) — that is dedicated to collect and redistribute information about the devices connected to the infrastructure

The !CHAOS Scalable Data Service is the result of the optimisation of the former live and historical services presented in previous publications [1, 2].

This service, that includes all APIs for inserting and querying datasets into the !CHAOS Data Cloud, has been redesigned in such a way to be horizontally scalable for achieving higher insert and indexing rate. The SDS is still in a early stage of development and some functionalities (data recover, ageing, customisation of queries and indexing) have not been fully implemented. Nevertheless, current ver-

sions allowed to test scalability and availability of the design. The info-graphics on Fig. 2 shows the layers of components implementing the Chaos Scalable Data Service:

- the proxy/entry point for data insert or query (write to both live or historical service, search, get last data from caching, etc...)
- the indexer for archived data
- the maintainer for data ageing, data recovering, etc.

Each SDS entry point can run any, or all, of the three services. In particular, the Scalable Data Service provides the distributed shared memory where the Control Units (CU) push data to and the clients fetch data from.

!CHAOS ON DAFNE BEAM-TEST FACILITY

The need to revamp the control system of the DAFNE Beam Test Facility (BTF) accelerator at Frascati gave the opportunity for a complete test of the !CHAOS components developed so far.

The DAFNE BTF is a transfer line branch that is not part of the injection chain of the main rings. Instead, it is used for machine studies or as a test facility for devices and equipments, mainly detectors. It means the work on the BTF Controls did not interfere with the machine operations.

From the controls point of view, driving the beam through the BTF transfer line (see Fig. 3) means operating a number of magnets, to be specific 4 dipoles, 6 quadrupoles and 2 H/V correctors. Actually, the first and the second pulsed dipole (DHRTB001 and DHPTS001) are critical also for the beam transport to the damping ring and to the spectrometer, so we decided to leave them under the control of the main DAFNE Control System.

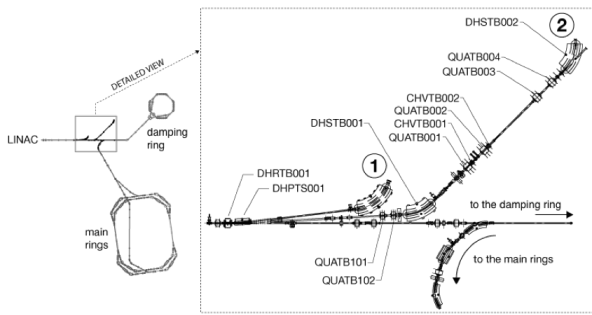


Figure 3: The BTF transfer line.

The H/V correctors are powered by 8 power supplies modules driven by a main PS unit. As consequence, taking control of even a single channel means taking the control over its main unit as well. Therefore, since the main unit housing the two BTF correctors (CHVTB001 and CHVTB002) also powers other correctors along the DAFNE transfer line, we had to leave them under control of the main DAFNE Control System.

As consequence, the set of components to be controlled for the trial run of !CHAOS involved 8 power supplies in total, installed in two different locations (5 in the modulator room and 3 in the damping ring room). The power supplies are all from the same manufacturer (OCEM) and implement the same communication interface and protocol. Originally, the power supplies were connected in daisy-chain through serial lines (RS 422, 4-wires) ending up in serial device servers (MOXA NPort 5650) that allow to monitor and control them from any network location. In order to release the power supplies from the main DAFNE Control System and make them available for the !CHAOS CUs, we laid two new temporary serial lines and connected them to two MOXA servers.

Control Unit Description

The !CHAOS framework provides the CUToolkit for the development of a Control Unit (CU), that is the abstraction of a device driver. In the BTF control system, the devices to be primarily controlled are the OCEM power supplies and consequently the corresponding !CHAOS controlled is named OCEM CU.

The OCEM CU can be viewed as a device server connecting the !CHAOS infrastructure to the physical device. It exposes to the !CHAOS control system a set of “attributes” and “commands” that are common to the power supply class. The device’s I/O data are sent to the !CHAOS distributed cache and are available for reading to client applications like GUIs and Control algorithms. Each OCEM CU serves just one power supply; multiple devices can share the same serial bus.

A Read and Write transaction with the OCEM power supply takes around 200-300 ms. Hence reading all the power supply attributes (5 attributes) takes approximately 1s to complete. Reading N OCEM power supplies connected in daisy chains takes about N seconds. Since the commands

are executed just after the update of all datasets, it’s fairly possible that a new command is processed after N seconds.

CU on a Virtual Machine

The first deployment of the Control Unit software in a real life application gave the opportunity for a heavy debugging of the !CHAOS libraries. The most tricky bug we discovered affected the timing of the threads by the scheduler of CU commands. The bug appeared as consequence of the more complex setup of virtual machines that was used, compared to the development environment. It altered the “normal” kernel schedule of the !CHAOS threads. The result of the bug was that, in some conditions, commands to the CUs were unexpectedly lost and, as it is typical for bugs dealing with temporisation, it was very hard to find.

Another issue, that affected the global performances, was due to the nature of the controlled power supplies that implements an extremely slow and clumsy communication protocol. As already seen in the previous paragraph, performing an update or executing a command on the CUs might take a time of the order of few seconds.

To overcome this problem during the operations we decided to not update all the attributes at the same rate but to prioritise the readout of the magnet’s current and polarity respect to other I/O values such as status, alarms and voltage. In such a way, we reduced the update time and command latency of about a factor 2.5.

CU on ARM Board

The !CHAOS libraries has been seamlessly re-compiled for ARM architectures. Cheap Single Board Computer have been used to replace the multi-drop serial lines driven by a MOXA server with a point-to-point connection that dramatically reduced the HD/SW latency of the communications with the power supplies.

We used a Beagle Bone Black equipped with an expansion board (cape) — developed on purpose — that provides four independent RS422 serial ports for a direct connection to four OCEM power supplies (see Fig. 4). The new design

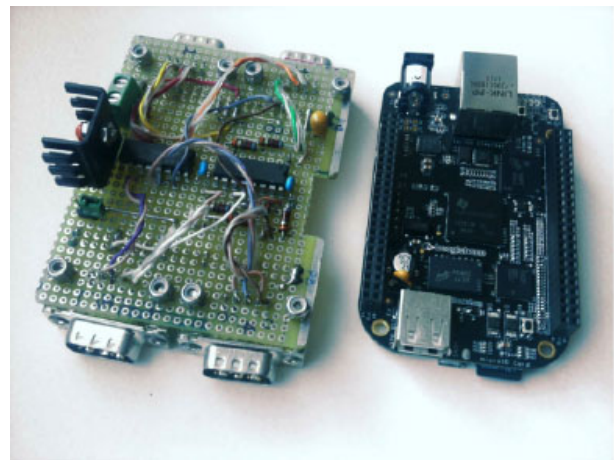


Figure 4: The Beagle Bone Black and the expansion board with 4 serial ports.

reduced the time latency in communicating with the power supplies by a factor four at a fraction of cost of the MOXA.

NON-HEP APPLICATIONS OF !CHAOS

The objective of the project "!CHAOS: a cloud of controls" is the development of a prototype of a nation-wide infrastructure aimed at offering high performance services for the management and the collection of physical data from devices and multifunctional sensors distributed over LAN and WAN. The project is being developed by a collaboration of INFN research groups lead by INFN-LNF and industrial partners such as National Instruments and the ADF Solaris, an ESCO company.

The project started on July 2014 and is expected to complete the prototype by the end of 2015. In particular, the project has the following specific objectives:

- development of a scalable platform for the control of large scale distributed collection of sensors, complex devices, and SoS, based on the !CHAOS framework's services and open source and open hardware;
- include services such as historicisation, data storage and analysis, distribution and presentation of polymorphic data;
- ensure high throughput, scalability and reliability;
- maximise compatibility with off-the-shelf devices and drivers;
- reduce costs and development time for end users;
- demonstrate the feasibility of a national platform open, accessible, scalable and reliable to control polymorphic sensor/devices/SoS.

The case-study for the prototype is offered by the ESCO Company. Energy Savings Companies provides a broad

range of energy solutions for energy consuming systems. The !CHAOS partner ADF Solaris is specialised in the designs and implementation of energy savings projects for buildings.

Generally, the ESCO company starts by performing an analysis of the property, designs an energy efficient solution, installs the required elements, and maintains the system to ensure energy savings during the payback period. The monitoring of the energy balance of the building, before and after the optimisation provides the information that allows to evaluate the savings in energy costs, that are used to pay back the capital investment of the project.

CONCLUSION

Although the !CHAOS overall design, as well as prototypes of the main services, have been completed we are still exploring new solutions. The new !CHAOS Scalable Data Service (SDS) for instance, merged into a single entry point the live and historical data storage. An all-!CHAOS control system is operational showing compatibility with many OS and HW. A non-HEP application aiming at completing the prototype of a nation-wide Cyber-Physical System infrastructure is under development.

REFERENCES

- [1] L. Catani *et al.*, "Introducing a new paradigm for accelerators and large experimental apparatus control systems", Physical Review Special Topics – Accelerators and Beams 11/2012; 15(11):10.
- [2] 6th Extremely Large Databases Conference, September 10-13, 2012 Stanford, California, USA,
<http://www-conf.slac.stanford.edu/xldb2012/>.

CONCEPTUAL DESIGN OF THE CONTROL SYSTEM FOR SPring-8-II

R. Tanaka, T. Matsushita, T. Sugimoto, JASRI/SPring-8, Hyogo, Japan
T. Fukui, RIKEN/SPring-8, Hyogo, Japan

Abstract

The SPring-8 storage ring was inaugurated 17 years ago in 1997. The storage ring is an 8-GeV synchrotron that functions as a third-generation light source, providing brilliant X-ray beams to a large number of experimental users from all over the world. In recent years, discussions have been held on the necessity of upgrading the current ring to create a diffraction-limited storage ring at the same location. Now, a plan to upgrade the storage ring, called SPring-8-II, has been launched. First, new beam optics capable of storing beams of 6 GeV was designed using a five-bend magnet system to obtain smaller electron beam emittance that would produce coherent X-rays that are brighter than those produced by the current ring. The design of a control system that would meet the performance requirements of the new ring has also started. Equipment control devices are based on factory automation technologies such as PLC and VME, whereas digital data handling with high bandwidths is realized using telecommunication technologies such as xTCA. In this paper, we report on the conceptual design of the control system for SPring-8-II on the basis of the conceptual design report proposed by RIKEN.

INTRODUCTION

SPring-8, which is a third-generation light source, has been in service for more than 17 years. Out of the 140,000 users in the SPring-8 community, approximately 4,500 users come to the site for synchrotron radiation experiments every year. In the past, productive scientific results have been obtained and shared with the community. Recently, an XFEL facility, SACLA, was constructed at the same site as SPring-8 and inaugurated for public use in 2012. The SPring-8 site is now unique in the sense that it is the only location to have both the SR light source and XFEL.

Experiments at SPring-8 provide good measurements of static phenomena with crystal samples because of incoherent X-ray beams. On the other hand, SACLA can measure fast-moving dynamical phenomena even for non-crystal samples such as thin-film proteins, with 10-fs pulses destroying the samples. This difference represents the characteristic features of the two light sources. On the basis of the proposed conceptual design report (CDR) [1], we can infer that a wide gap exists between the two machines.

To narrow the present gap, a new project involving a new storage ring is planned; that is, the current storage ring will be replaced by a new storage ring at the same location. In the proposed project, a 1-GeV linac, which currently serves as the injector for SPring-8, will be

replaced with an 8-GeV SACLA linac. The CDR says, “We know how it happens but we do not know why it happens. We should provide a tool to offer answers to the question, why”. This is the motivation of the SPring-8-II project.

The control system plays an essential role in the working of large accelerator facilities currently operational in the world. The controllability and operability of the facility strongly depend on the architecture and implementation of the control system; hence, the current control system of SPring-8 will be upgraded suitably to fulfil the performance requirements of the SPring-8-II storage ring, as described in this paper.

SPring-8-II Project

The CDR of the SPring-8-II project is available on the Web [1]. The basic idea is to replace the current 8-GeV storage ring with a low-emittance 6-GeV storage ring at the same location by reusing the present machine tunnel. The C-band linac of SACLA will be used as a full-energy injector for the new ring. To transport electron beams to the new ring, a beam transport line, XSBT, is constructed, as shown in Fig. 1. The X-ray beamlines for the present undulators will be retained. The blackout period (construction period) is expected to be one year or less.

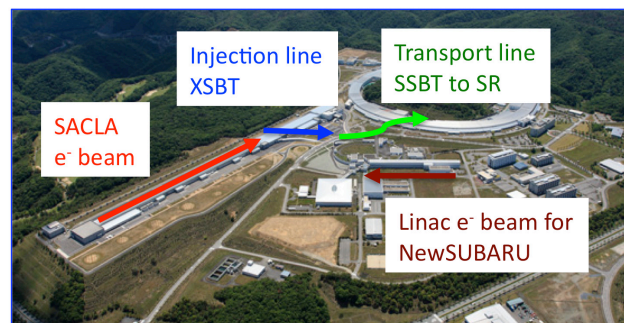
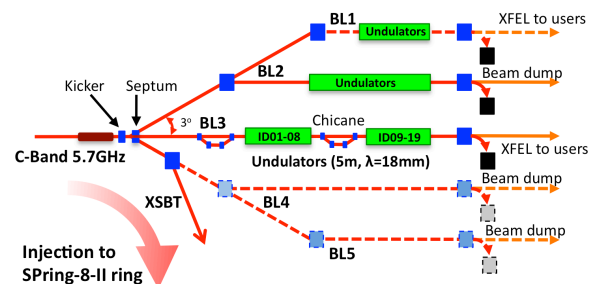


Figure 1: Injection line to SPring-8-II by XSBT at SACLA (upper), and beam transport route (lower).

CONTROL DESIGN OVERVIEW

The control system for SPring-8-II is based on the concept of “*right platform in the right place.*” The system will be constructed by using heterogeneous platforms. While beginning the construction of the current storage ring, the control system design group chose the VME and PLC platforms so that the system could be developed efficiently within a short construction schedule by a small number of group members. Considering this scenario, we now have a rich history in developing equipment control devices. Therefore, we can choose the *best-fit devices* out of a variety of device options, while considering the device availability/sustainability in the future.

Implementation Concept

Control processing performance is usually classified as fast control and slow control. The features necessary for slow control are reliability, simplicity, and cost effectiveness; therefore, we can use PLC and VME platforms and reuse well-established software developed previously. However, how fast processing capabilities beyond the bandwidth supported by the VME backplane can be developed is a question that needs to be answered soon.

Today, great progress has been made in the electrical and information technology fields. Some of the emerging platforms can be applied to accelerator control, such as xTCA (ATCA, MTCA), which is a family of platforms proposed by the Telecommunications Carriers Association. The higher processing capability of digital data/signals is important for accelerator control, especially low-level RF (LLRF) control.

Accelerator Control System

An example of RF control that consists of a composite platform is shown in Fig. 2. The LLRF controls the phase and amplitude modulation of the low-power RF through feedback loops; therefore, we use xTCA for digital processing after the analog-to-digital signal conversion. High-power parts such as the klystron power supply will be controlled by a PLC interconnected by FL-net to a VME, or by a PLC directly connected to the Ethernet.

At this moment, we have not decided which xTCA platform will be used: ATCA or MTCA. A decision will be made after the R&D phase, which will start soon.

The beam operation of the new storage ring will be difficult because of the narrow dynamical beam aperture and the sensitivity of the circulating beam to residual error sources such as misalignment and magnetic field leakage. Combination control involving the beam position monitor (BPM) and St-magnet power supply (PS) should have sufficient feedback performance for the global beam feedback and COD correction to achieve stable beam operation. A shared memory network is used for fast data sharing and message communication without software interconnection, as shown in Fig. 3.

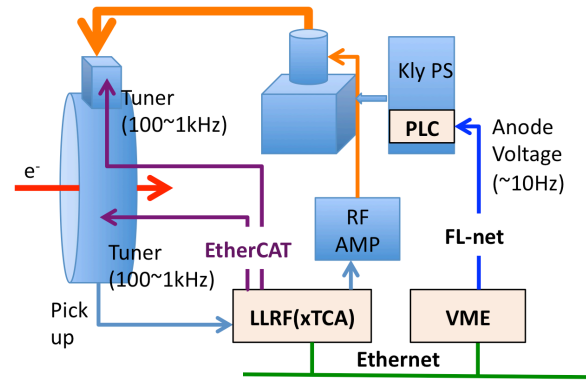


Figure 2: Control scheme of RF control system. The system consists of a PLC, VME, and xTCA designed on the basis of the “right platform in the right place” concept.

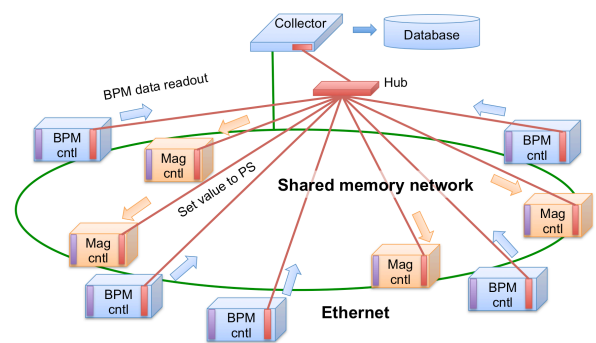


Figure 3: Schematic view of BPM readout and St-magnet PS control using shared memory network.

Beamline Control

The current beamline control consists of three parts: (1) insertion device control, (2) optics component control (frontend, transport channel), and (3) experimental station control (monochromator), as shown in Fig. 4a. The controllers consist of VMEbus systems and a PC. A separated control subsystem provides an independent construction/tuning schedule and efficient maintenance, which serves as a degree of freedom; however, as a trade-off, there will be communication overhead between controllers. Such a trade-off somewhat disturbs the synchronized operation of the insertion device and beamline components during fast experiments. In this sense, it is better to connect the controllers using a single line (loop) for communication. Currently, EtherCAT is a promising candidate that can provide real-time performance; its performance is sufficiently good for providing undisturbed synchronized operation of beamline components and a monochromator, including an insertion device (ID).

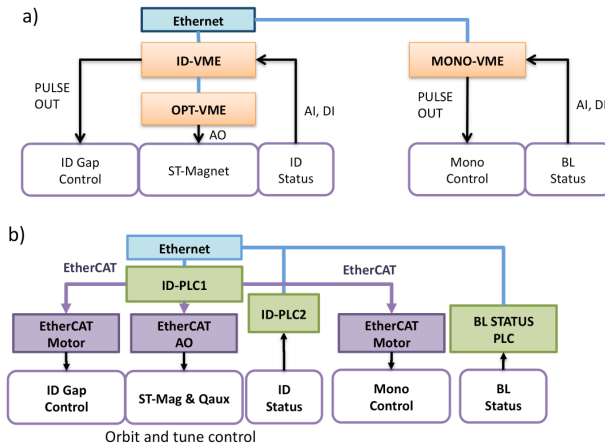


Figure 4: Schematic view of current beamline control (a), and proposed control for SPring-8-II (b).

An auxiliary PS of a Q-magnet and the St-magnet PS are controlled by the same controller that belongs to the ID section to correct the electron beam orbit and tune at the ID region.

Control Framework

The SPring-8 storage ring has been controlled successfully by using the MADOCA framework since 1997 [2]. Now, MADOCA is widely used not only for the accelerator and beamline but also for the data acquisition (DAQ) system for SACLAL experiments. Owing to the wide application and long history of MADOCA, there are requests for upgrading the functions of the framework. Therefore, we launched an upgrade project for the framework, named MADOCA-DX (Daq eXtension). The project was completed successfully in 2013, and the upgraded framework product is called MADOCA II [3]. The MADOCA II middleware is implemented using ZeroMQ and MessagePack. The MADOCA II software scheme is shown in Fig. 5.

MADOCA II supports the following features:

- 1) A variable-length message with no length limitation
- 2) Handling of binary data such as image data
- 3) Windows OS and LabVIEW interface to MADOCA
- 4) Better message transaction (~1 ms for a round trip)
- 5) Logging data management by NoSQL database

Database

In MADOCA II, machine status data (logging data) are managed by NoSQL database engines that consist of a combination of Redis and Cassandra [4]. The key-value-store (KVS) database is suitable for handling chronologically ordered row-type data. In fact, the data transaction performance is one order of magnitude better than that of a relational database we are currently using. The Redis database manages the machine status data in the online memory. The “Writer” processes that run on

relay server machines transfer the status data to the Redis and the Cassandra simultaneously as shown in Fig. 5. The computing system for the Cassandra database consists of cost-effective PCs, which provide operation redundancy.

We still expect to use the relational database management system for the configuration and alarm databases. The database management system for SPring-8-II will be a hybrid system consisting of SQL and NoSQL database engines.

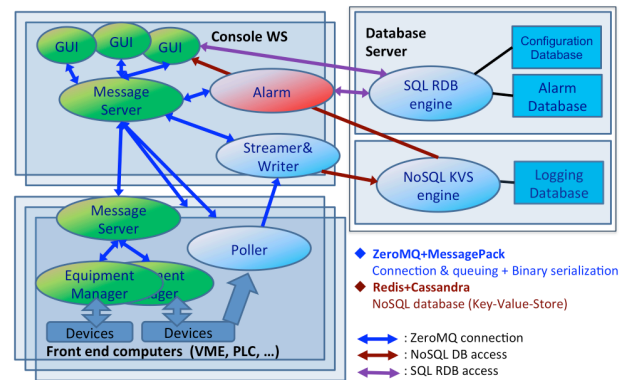


Figure 5: Software structure of MADOCA II that uses ZeroMQ and MessagePack for the message transaction. The NoSQL database engines receive machine status data from the Writer processes.

FUTURE WORK

The accelerator control system of SPring-8 has been functioning well under the MADOCA framework since 1997. Now, we have improved the framework and named it as MADOCA II.

We plan to start R&D on new technology such as xTCA. In addition, we plan to move one step ahead from the current conceptual design to a technical design, as we move toward the commissioning of SPring-8-II in the early 2020s.

REFERENCES

- [1] Conceptual Design Report, <http://rsc.riken.jp/>
- [2] R. Tanaka et al., “The First Operation of Control System at SPring-8 Storage Ring”, Proceedings of ICALEPCS 97, Beijing, China, 1(1997).
- [3] T. Matsumoto et al., “Next-generation MADOCA for the SPring-8 Control Framework”, Proceedings of ICALEPCS 2013, p.944, San Francisco, CA, USA, (2013).
- [4] M. Kago, A. Yamashita, “Development of a Scalable and Flexible Data Logging System Using NoSQL Databases”, Proceedings of ICALEPCS 2013, p.532, San Francisco, CA, USA, (2013).

COMMON DEVICE INTERFACE 2.0

P. Duval, H. Wu, DESY, Hamburg, Germany
J. Bobnar, Cosylab, Ljubljana, Slovenia

Abstract

The Common Device Interface (CDI) is a popular device layer [1] in TINE control systems [2]. Indeed, a de facto device server (more specifically a 'property server') can be instantiated merely by supplying a hardware address database, somewhat reminiscent of an EPICS IOC. It has in fact become quite popular to do precisely this, although the original design intent anticipated embedding CDI as a hardware layer within a dedicated device server. When control system client applications and central services communicate directly to a CDI server, this places the burden of providing useable, viewable data (and in an efficient manner) squarely on CDI and its address database. In its initial release variant, any modifications to this hardware database needed to be made on the file system used by the CDI device server (and only when the CDI device server was not running). In this report we shall describe some of the many new features of CDI release 2.0, which have drawn on the user/developer experience over the past eight years.

CDI AND TINE

Although the Common Device Interface (CDI) can be used outside of the TINE control system it is nonetheless strongly coupled to the TINE libraries as well as the TINE application programmer's interface (API) and the TINE naming convention and hierarchy. It is worthwhile to discuss some of these aspects in order to better understand the discussion that follows.

TINE itself does not require any specific hardware device layer in order to provide control system services. On the other hand TINE is weakly coupled to CDI in that specific CDI hooks are embedded in the TINE library. This in turn allows a TINE device server to utilize embedded CDI services for hardware access.

CDI Hardware Server

A very simple manifestation of embedded CDI services is the so-called CDI hardware server, which is essentially a generic TINE device server providing access to the hardware devices contained in the CDI database. Such a server provides no additional device control intelligence beyond that which can be configured in the database. Originally it was imagined that although the hardware server would be a very useful tool for testing hardware, developers would design device servers based on direct data acquisition via embedded services within a single framework. In practice, the hardware server itself has become the mainstay of hardware access for most TINE device servers. In most cases a device server with specific control intelligence is designed as an effective middle

layer server communicating with a front-end CDI hardware server. In many cases, however, client applications communicate directly with CDI servers and there is no additional device server in the picture at all!

At this juncture we should point out that the CDI hardware server should properly be termed a property server and not a device server. Control systems are sometimes categorized into those which provide a database-driven paradigm (such as EPICS [3]) or a device-server paradigm (such as DOOCS [4] or TANGO [5]). Although TINE falls into the device-server camp it also supports property servers. Traditional device-servers treat instances of equipment as named devices and these devices have properties which one can access via the device server. A property server on the other hand considers services and information to be designated as properties located on some host, and such a service property will likely apply to a set of keywords.

The services a CDI server offers of course include bus access properties, such as sending and receiving on a hardware bus. For such properties, the keywords correspond directly to named hardware addresses, referred to as CDI devices. Other services include bus and template information as well as database management services.

CDI SPECIFICS

CDI operates on a plug-and-play basis, making use of bus-plug interfaces to the device hardware. A bus plug is a hardware specific shared library which encapsulates the details of the hardware bus I/O behind the CDI API. The CDI shared library is told which bus plug libraries to load via a CDI manifest database.

A CDI address database provides the cross-reference information necessary to instantiate the named hardware devices that CDI will export. The use of CDI address templates can greatly facilitate this instantiation. As a database format, CDI uses comma-separated value (CSV) files, which are easy to view in any text editor and fit seamlessly into any spreadsheet application such as Excel.

A CDI address database snippet is shown in Fig. 1. Here one can see how templates make life easy. Templates are defined by specifying the bus name 'TEMPLATE' and providing both the template name and template field name separated by a colon. If a device instance such as 'PU01I' specifies a template <BLM> in its address parameters then it will automatically expand into multiple CDI I/O devices given by instance name and template field name separated by a dot. The bus itself in this case is the in-house DESY bus SEDAC. The initial entry gives the special bus name 'FIELDBUS' which

provides a method for giving a name to the bus on line SEDAC line 1, namely 'BLMs'. Not shown in the example is another special bus name BITFIELD which can be used to define the elements of a TINE bitfield which can be applied to read-back data.

NAME	BUS	LINE	ADDRESS_BASE	ADDRESS_PARAMETERS	FORM/ACCESS
SEDAC:BLMs	FIELD BUS	1	0	0 short	RD
BLM:hiWord	TEMPLATE	1	0	2 short	RD
BLM:loWord	TEMPLATE	1	0	1 short	RD
BLM:Mode	TEMPLATE	1	0	0:05 short	WR
BLM:Reset	TEMPLATE	1	0	0:03 short	WR
BLM:PowerClr	TEMPLATE	1	0	0:07 short	WR
BLM:Time	TEMPLATE	1	0	0:06 short	WR
BLM:PreScaler	TEMPLATE	1	0	0:04 short	WR
PU01I	SEDAC	1	2.8	<BLM> short	
PU01O	SEDAC	1	2.8	<BLM> short	
PU02I	SEDAC	1	2.8	<BLM> short	
PU02I_I	SEDAC	1	1.8	<BLM> short	
PU03O	SEDAC	1	2.8	<BLM> short	
PU03O_I	SEDAC	1	1.8	<BLM> short	

Figure 1: CDI address database snippet.

The database snippet shown in Fig. 1 is simplified and omits many optional columns which could be used to specify other I/O instructions including data masks and calibration rules. As noted above, a well-configured database can often establish a CDI server which provides finished, ready-to-use data, obviating the need to develop any other device server. In the past, this was usually only true when the targeted hardware was 'simple', i.e. where slow data access was sufficient and multiple clients to the CDI server could be tolerated.

With the advent of scheduling, asynchronous listeners, and the asynchronous triggering from bus plugs found in CDI 2.0, the number of cases where a CDI server alone is sufficient for control purposes has greatly expanded. We shall come to these topics in more detail below.

One further important detail concerning the property server nature of a CDI server should be mentioned. Namely the fields of the template devices alluded to above are themselves registered as properties, whose keyword lists consist of those instances making use of the template. To continue with Fig. 1 for example, the CDI server would also export a property 'Mode' and list all of the PU01I, PU01O, etc. instances as keyword devices. More importantly it would treat this property (referred to as a CDI extended property) as a multi-channel array (MCA) property, which would participate in the TINE MCA contract coercion [6] used to provide efficient data transfer, server to client. Underneath the hood, any call to a CDI extended property maps to the corresponding full CDI device name ('*instance.template-field*') and the bus property 'RECV.CLBR', which translates into 'receive on the bus and apply any calibration rules'.

Asynchronous Listeners

If a call to a CDI server always ended up making a synchronous bus i/o operation this could lead to bottle necks and inefficiencies depending on the amount of data and the read-back intervals involved. To this end, a CDI server will recognize when a client establishes an

asynchronous contract and then establish a local asynchronous static listener, which will regularly receive updates for the requested device from the hardware and report these results to the caller. In fact, when an asynchronous listener is in play, any synchronous request from a caller will return the most recently acquired data without any additional hardware I/O. This of course does not apply to SEND operations on the bus, but as most server I/O tends to be read-backs this mechanism makes the CDI server very efficient at servicing multiple clients.

A CDI server can now easily specify in its address database which properties (or template fields) should have an automatic listener applied at start time.

Scheduling

When an asynchronous listener is in place, then the CDI library can itself monitor read-back values and signal a data-change event. This is done by calling the TINE scheduler and is referred to as scheduling a property. CDI will generally monitor data at an interval provided in the address database, although it can also react to asynchronous events coming from the bus plug. Scheduling is a good way to avoid latency and deliver data to listening clients in a timely manner.

Local Histories and Alarms

To further the cause for creating a viable device server (or rather, property server) merely by applying the proper database settings, we note here that TINE local histories of any designated CDI device can be easily incorporated into the database. Raising alarms is another matter and requires a separate TINE alarm watch database [2]. However, this latter task is hardly daunting.

Remote Database Access

One of most exciting features of CDI 2.0 is the ability to remotely access and modify the database of a running CDI server by making use of exported database properties. Although reading the database is allowed by anyone, writing to the database requires traversing TINE security. Should a local database be updated, a backup is made of the most recent database and a rollback to the previous database is also easily available in CDI 2.0.

The ability to update a CDI database via the CDI server itself simplifies matters to no end when one is dealing with embedded or semi-embedded CDI servers on, say, a PC104 card. In the old days, this typically required a secure login/secure copy of the new database and a server re-start.

If enabled, the CDI server can also be remotely 'reset', whereby the wrapping TINE server unloads all CDI libraries and returns the running server to its original uninitialized state and then re-initializes.

CDI EDITOR

As seen above, a CDI database makes good use of CSV files. Although an accomplished spreadsheet user might be comfortable with this, working with detailed

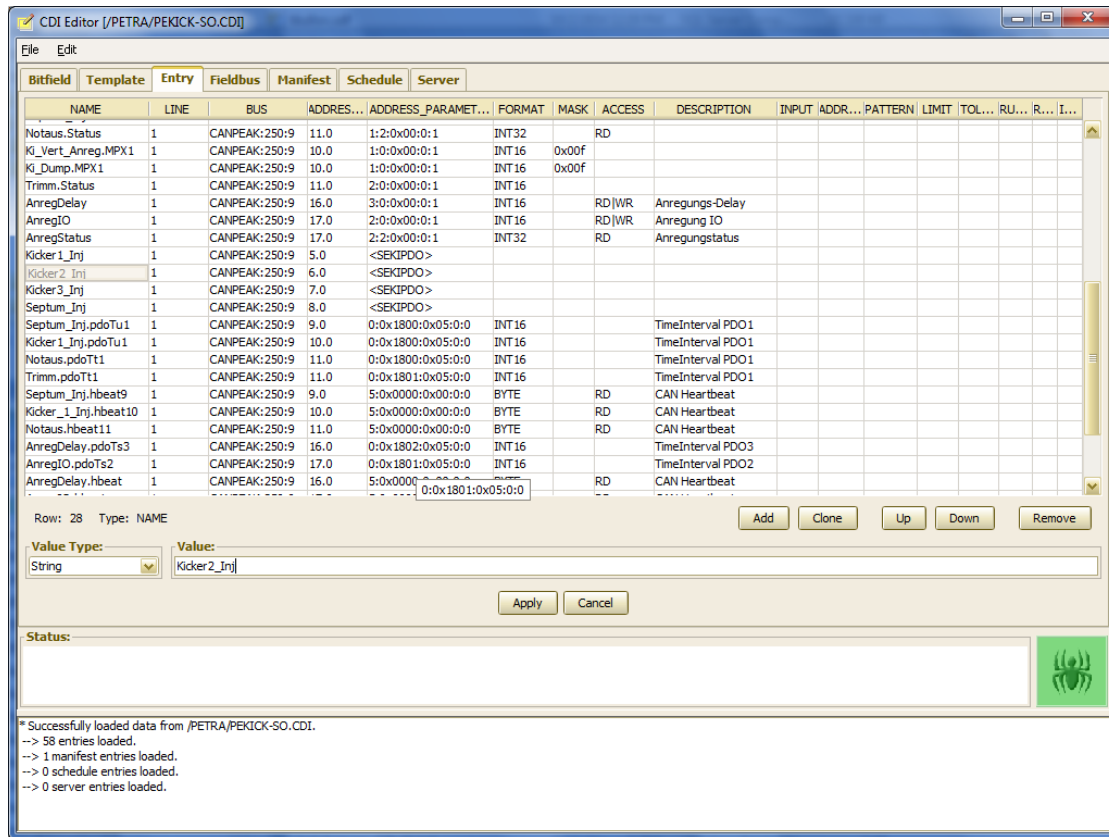


Figure 2: Example of the CDI Editor. From the title bar one can see that the database was acquired directly from a server called /PETRA/PEKICK-SO.CDI. The above view focuses on the device *entries*.

information in a large spreadsheet can be tedious and error prone. To this end, a CDI editor is now available, which shows various aspects within their contexts and, most importantly, checks for consistency where it can.

As a CDI 2.0 server offers remote access to its database, the CDI editor can either work with a local database on the local file system or acquire the database from any running CDI server.

The user can browse and edit in a clear, intuitive manner all aspects of the database and then have the option of updating to the targeted server if he has the access rights to do so.

An example is shown in Fig. 2 above.

CDI DEPLOYMENT TOOL

Another issue which has surfaced over the past eight years of CDI usage is that of distributed databases. Namely, there are situations where not only is a real device server necessary, but said device server needs to acquire data from multiple, distributed, CDI hardware servers, each containing instances of the same hardware type. A simple modification to a template then involves updating not one, but numerous CDI address databases (not an enviable task, even making use of the CDI editor).

To simplify such operations the CDI 2.0 package also features a CDI deployment tool. This utility is able to generate the multiple CDI address files from a single master address database. An additional requirement here

is that the master address database must have a column labelled TARGET, which should contain the destination CDI server address (in TINE */Context/Server* notation) for each hardware instance. The deployment tool is able to then generate a set of database files on the local file system in a directory structure based on the *Context* and *Server* information provided. Or, it is able to make use of the remote database access described above to push the specific database to the targeted server.

CONCLUSION

CDI has reached a new level of maturity and continues to be a major workhorse in TINE control systems. The feature-set of release 2.0 makes CDI considerably more versatile, and the ability to generate a hardware address database is now vastly simplified through the CDI editor.

REFERENCES

- [1] P. Duval and H. Wu, "Using the Common Device Interface in TINE", Proc. PCaPAC'06, <http://jacow.org/>.
- [2] TINE website: <http://tine.desy.de/>.
- [3] EPICS website: <http://www.aps.anl.gov/epics/>.
- [4] DOOCS website: <http://doocs.desy.de/>.
- [5] TANGO website: <http://www.tango-controls.org/>.
- [6] P. Duval and S. Herb, "The TINE Control System Protocol: How to achieve high scalability and performance", Proc. PCaPAC'10, <http://jacow.org/>.

INEXPENSIVE SCHEDULING IN FPGAS

W. W. Terpstra, D. Beck, M. Kreider, GSI, Darmstadt, Germany

Abstract

In the new scheme for machine control used within the FAIR project, actions are distributed to front-end controllers (FEC) with absolute execution timestamps. The execution time must be both precise to the nanosecond and scheduled faster than a microsecond, requiring a hardware solution. Although the actions are scheduled at the FEC out of order, they must be executed in sorted order. The typical hardware approaches to implementing a priority queue (CAMs, shift-registers, etc.) work well in ASIC designs, but must be implemented in expensive FPGA core logic. Conversely, the typical software approaches (heaps, calendar queues, etc.) are either too slow or too memory intensive. We present an approach exploiting the time-ordered nature of our problem to sort in constant time using only a few memory blocks.

INTRODUCTION

In a schedule-driven control system, pending actions include an execution timestamp. When the timestamp matches the current time, the responsible front-end controller (FEC) executes the action. Within the scope of the FAIR project, many physically distributed FECs will execute actions in concert. Actions are coordinated and distributed by a central unit, the data master, which controls beam production.

Unfortunately, the data master is quite complicated. The actions it requires a FEC to take may be delivered in an order different than the execution order. This paper describes how a FEC takes an incoming set of out-of-order actions and outputs them in sorted order. In principle, a single FEC may control many devices attached to many interfaces. However, for the purposes of this paper, we will concern ourselves only with the actions delivered by a FEC through a single interface. We also omit message processing.

Concretely, a FEC processes action tuples (a, x) , where a is the action to execute and x is the time at which it must be executed. At any given time t , the FEC has a set P_t of pending/yet-to-be-executed tuples. That is, $x \geq t$ for all $(a, x) \in P_t$. At time t , the FEC must output a if $(a, t) \in P_t$; this is illustrated in Figure 1. Obviously, it is physically impossible for a single interface to execute two actions concurrently. The data master would never ask a FEC to do something impossible. Therefore, we can assume $x = y \rightarrow a = b$ for $\{(a, x), (b, y)\} \subseteq P_t$.



Figure 1: Actions (a, x) flow from the data master to the FECs. They are stored in P_t until $t = x$ and then output.

For FAIR, the control system is required to have nanosecond precision. This means that at least the execution/output of actions must be synchronized by hardware. Furthermore, the data master distributes the schedule via gigabit Ethernet. The FECs must be capable of accepting the schedule at the full rate, which suggests hardware may be required here as well. FECs include an FPGA and therefore we choose to solve the problem of receiving the schedule, sorting it, and outputting it, all in the FPGA.

AN FPGA CRASH COURSE

We can program an FPGA to contain customized hardware. Like an application-specific integrated circuit (ASIC), we can implement any digital circuit consisting of logic gates and registers. Unfortunately, FPGAs only have a limited number of comparatively slow logic elements to implement core logic (logic gates and registers). FPGAs also include many block memories, which are essentially small SRAM chips embedded inside the FPGA. These block memories have the same density and performance as they would in an ASIC. For these reasons, a good FPGA design tries to minimize core logic by leveraging block memory.

Digital logic is generally clocked. Registers take their values on the rising edge of a clock signal. In a modern FPGA with a reasonably complicated design, the clock speed is generally limited to $<500\text{MHz}$. That means that each clock cycle takes $>2\text{ns}$. The particular sorting circuit presented here can run at 325MHz ($\approx 3\text{ns}$ period) on an Altera Arria V chip. To achieve nanosecond precision, this means we need to be able to output an action in very few clock cycles. Furthermore, we need to accept new tuples at a similar rate.

Our goal is thus to accept a tuple (a, x) on clock cycle t to compute $P_{t+1} = P_t \cup \{(a, x)\}$. Furthermore, if there is $(a, t) \in P_t$, then we output a . In other words, on every clock cycle, we need to potentially accept a new action into the buffer and output the action whose timestamp is due.

APPROACH

Sorting a set of $n = |P_t|$ numbers requires $O(n \log(n))$ comparisons. If we must sort one timestamp every cycle, that means $\log(n)$ comparisons per cycle. For FAIR, timestamps are 64-bit numbers. In a modern FPGA, comparing even a single pair of 64-bit numbers in one clock cycle would reduce the maximum performance to $\approx 125\text{MHz}$. A hardware technique called pipelining can spread this work between multiple clock cycles. In our implementation, a 64-bit comparison is done in 3 steps to achieve the target performance. Unfortunately, this makes the comparator quite expensive. While there are techniques to spread out the work of all $\log(n)$ comparisons across multiple cycles using pipelining [1], these approaches cost significant hardware.

We solve the scheduling problem in a different way, requiring only 1 comparison per cycle. The key insight is that we do not actually need to solve the sorting problem. We only need to output the current action a for time t . Sorting requires finding in each step the next *smallest* timestamp. We only need to find actions with the current timestamp.

Imagine a gigantic table T that stores actions. At any given time t , you locate row t in the table and read out the action to execute; $a = T[t]$. Every time a new tuple (a, x) arrives from the data master, you just write a into entry x in the table; $T[x] := a$.

The only problem with this simple approach is that the table must have 2^{64} entries, one for each possible time x . We must find a way to store the table more compactly. The first thing to keep in mind is that an FPGA design has finite physical resources. We must define an upper limit to the number of pending actions which the FPGA design can store. This limit can be reconfigured whenever the FPGA is reprogrammed and currently we have set it to 256. Block memory in a modern FPGA is only available in ≥ 256 -entry chunks, so anything less is no cheaper.

With only 256 entries, we cannot directly implement the gigantic look-up table. However, we can record anything that happens in the next 256 cycles. Just check if an incoming tuple (a, x) obeys $t \leq x < t + 256$. If it does, then set $T[x \bmod 256] = a$. On clock cycle t , just look up $a = T[t \bmod 256]$ and clear $T[t \bmod 256]$ for re-use later.

Unfortunately, tuples arriving from the data master may be far in the future; $x \geq t + 256$. If we wrote them into the table anyway, we would execute them too early. However, we must still record these actions somewhere! Our scheme is to instead use two tables. There is one (unsorted) table used to store all pending actions, the *pending table*. There is another table used to record references to the pending table for those actions with timestamps due in the next 256 cycles, the *calendar*. This is illustrated in Figure 2.

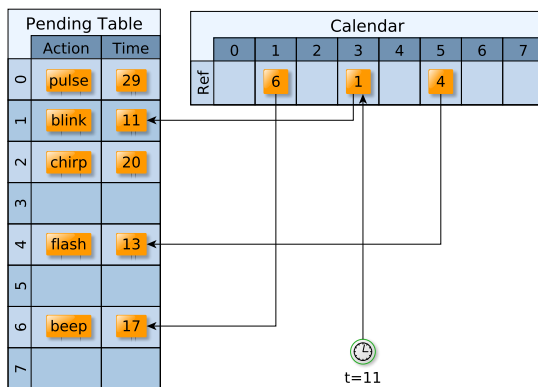


Figure 2: An example of 8-entry compressed tables.

Of course, these two tables alone do not solve the problem. Actions which execute more than 256 cycles in the future do not appear in the calendar. However, we can continuously scan the pending table's records, one record per clock cycle. If the pending record has a timestamp due within 256 cycles

($x < t + 256$), put a reference into the calendar. In fact, once we have this *scanner* process, we no longer need to write incoming tuples into the calendar at all. We need only write the tuple into the pending table and the scanner will put a reference into the calendar for us.

The reason this scheme works is that the scanner is guaranteed to fill the calendar before an action must be executed. Pick an arbitrary tuple (a, x) from the pending table. Define u as the time when the tuple was stored into the pending table. Because t grows without bound, eventually $x < t + 256$ and the scanner will put a reference in the calendar. Define time v to be the first time that the scanner does this. The scanner re-scans the records in the pending table every 256 cycles. Thus, at time $w = v - 256$, as long as (a, x) is already in the table ($u < w$), the scanner will have previously visited the record. However, since v was the first time t where $x < t + 256$, we know that $x \geq w + 256 = v$. In other words, a reference to (a, x) is placed into the calendar at time v , before it must be executed (x).

The above proof requires that the tuple is already in the pending table at time w . Recall that at time $t = v$, a reference will be placed into the calendar because $x < t + 256$, and thus $x - 256 < v$. Therefore, to ensure correct behaviour, we require tuples to be delivered early; $u < x - 256 - 256$. This implies that $u < v - 256 = w$, as required in the proof. What this means is that you must store a tuple into the pending table 512 cycles before its execution timestamp. Fortunately, in FAIR a time budget of $4\mu s$ is quite reasonable; this allows us to run as slowly as 125MHz (8ns period).

DESIGN

For a complete implementation, we must also manage the free entries in the pending table. A new incoming action must be written into an empty record in the pending table. When an action is executed, the dispatcher already removes the reference from the calendar. However, the entry in the pending table must be released as well. To facilitate allocation and release, we implement a *manager* that records free pending indexes in a *free stack*.

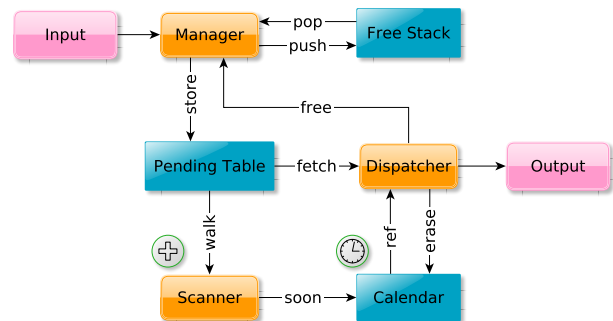


Figure 3: Block Diagram of block memories and processes.

As shown in Figure 3, we have in total three tables: pending, calendar, and free. They are controlled by three processes: dispatcher, scanner, and manager. On each cycle, the

dispatcher checks the current time in the calendar for a reference and erases whatever it finds. It then fetches the action from the pending table and outputs it. Finally, it tells the manager to free the pending entry. The scanner repeatedly walks the pending table. If the scanned record has a timestamp $< t + 256$ (the only comparator/adder in our design), the scanner writes a reference to that soon-to-be-executed record into the calendar. The manager accepts new tuples from the network and stores records into the pending table. If the manager must both allocate and free a record, it overwrites the freed record with the new tuple. If it must only free, it pushes the free pending-table index to the free stack. If it must only allocate, it pops the free stack and stores the tuple to the free index.

One very annoying constraint in FPGA design is that a block memory can only execute a single write and a single read per cycle. The free stack poses no problem here; the manager at most reads or writes one entry per cycle. The manager is also the sole writer to the pending table. The action column of the pending table is only read by the dispatcher, so this poses no problem. The timestamp column (in our actual implementation) is read by both the scanner and the dispatcher. However, two readers and one writer can be implemented by duplicating the table. What is not so straight-forward is the calendar.

The calendar is only read by the dispatcher. Unfortunately, it is both written by the scanner (to add references) and erased by the dispatcher (to remove references). On some platforms, one can implement same-address get+erase using one half of a true dual port memory. Sadly, this trick is not portable (same-port old-data) and does not work on the Arria V. Instead, we used two bank-interleaved block memories. Since t is incremented every cycle, it changes parity every cycle. Thus, we can read from the next even memory address $t + 1$ at the same time as we erase the odd memory address t .

To achieve high performance, each of the processes is heavily pipelined. Unfortunately, this creates a structural hazard. The scanner might write a reference into the calendar where the dispatcher is only halfway-done executing. To solve this, we just increase the width of the calendar slightly, and forbid the scanner from writing near the position of the dispatcher. Pending records near the scanner thus get scanned twice before being executed by the dispatcher, and only one of those times can be a structural hazard.

OUTLOOK

We presented a way to output actions scheduled according to their timestamps. While the actions arrive in unsorted order, we are still able to accept and output one action every clock cycle. Due to careful pipelining, the design can run at up to 325MHz on an Altera Arria V. The design uses only three memory blocks, one 64-bit comparator, and three

very simple processes. The low cost of this solution was achieved by formulating the real-time scheduling problem as a lazily-updated look-up table.

Our approach has many similarities to calendar queues [2]. However, unlike calendar queues, time spent inspecting empty calendar entries is not time wasted. In a calendar queue, one must retrieve the *next* scheduled action. In real-time scheduling, we need only retrieve the action scheduled for the current time. Calendar queues thus suffer the same sorts of problems as bucket- and radix-sort [3]; they depend on the distribution of execution timestamps and performance can become significantly degraded.

Unfortunately, even our approach is not completely immune to a poor timestamp distribution. If, for some reason, more than 256 pending actions need to be stored at once, the block memory used for the pending table will be exhausted. For this reason, the FAIR data master only schedules actions that will be executed in the immediate future.

Compared to a heap-based approach [1], our approach deals poorly with two actions scheduled to occur simultaneously. While this situation must be avoided in any case, as the relative order between two simultaneous actions is undefined, a heap implementation would at least output these two actions back-to-back. Our approach will instead delay one of the actions by 256 cycles. We think that the significant area and performance benefit inherent to our approach justifies this small concession.

Finally, in an ASIC, it might make sense to use a shift-register-based approach [4]. To implement this in an FPGA would be prohibitively expensive as the storage of the records must be done in registers, not block memory. Even in an ASIC, the necessary CAM-like shift-register setup will cost far more than an equivalent SRAM. Indeed, it is hard to imagine any solution exists which is significantly smaller or faster than the one we propose here.

REFERENCES

- [1] Wojciech M Zabołotny. Dual port memory based heapsort implementation for FPGA. In *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2011*, pages 80080E–80080E. International Society for Optics and Photonics, 2011.
- [2] Randy Brown. Calendar queues: a fast 0 (1) priority queue implementation for the simulation event set problem. *Communications of the ACM*, 31(10):1220–1227, 1988.
- [3] D. Knuth. Sorting by distribution. In *The Art of Computer Programming, Volume 3: Sorting and Searching*, chapter 5.2.5, pages 168–179. Addison-Wesley, 3 edition, 1997.
- [4] Chen-Yi Lee and Jer-Min Tsai. A shift register architecture for high-speed data sorting. *Journal of VLSI signal processing systems for signal, image and video technology*, 11(3):273–280, 1995.

TESTBED – AUTOMATED HARDWARE-IN-THE-LOOP TEST FRAMEWORK

P. Maslov, K. A. Meyer, K. Žagar, Cosylab, Ljubljana, Slovenia

Abstract

In a big physics facility such as ITER or ESS, the control system is typically updated at least 3 times a year. This means that prior to each minor release all components should be tested. For testing DAQ drivers, a test plan should be written, based on which a manual test is performed. The idea behind the TestBed suite is to execute tests automatically. Our TestBed is a PXI chassis which contains an embedded controller running the CODAC control system on a Scientific Linux operating system and a DAQ board capable of generating and acquiring analog and digital signals. It provides an easy-to-use framework written in Python and allows for the quick development and execution of automatic test scripts.

ARCHITECTURE

From the hardware perspective, each system under test (SUT) is physically connected to the TestBed (TB) (Fig. 1) with a connector board using a predefined pin configuration. Both SUT and TB are connected to the LAN (not shown in Fig. 1).

The software part consists of three tiers:

1. Software that provides the desired functionality of a DAQ board:
 - C executables
 - EPICS device support (NDS [1] driver + IOC)
 - *LabVIEW* interface
2. Python bindings in the form of a class that reflect the given functionality of 1)
3. Automatic test cases written by the test-plan engineer using 2).

The NI-PXI6259 functionality that is supported in the TB suite includes:

- Analog input/output (static) on a desired channel
- Analog input (waveform) on a trigger
- Analog output (waveform – sine/saw/square/from file) on a trigger signal
- Configuration of the DIO port mask
- DIO diagnostics: port mask (0 – input, 1 – output) and lines state (0 – low, 1 – high)
- Digital input/output (static) on a desired line
- Device reset

The underlying connection protocol is SSH and is provided by the Python package Paramiko (the NDS implementation utilizes the Python package CaChannel).

TESTBED BASE CLASS

The Testbed Python class (Fig.2, left hand-side) provides a set of test methods to be executed on an SUT. These methods can be extended for any DAQ board and then used to quickly write automatic test scripts.

An example of such test case script showing three tests for the NI 6259 DAQ board (Fig.2, right hand-side) is:

1. test_aio_static: generate static signal on AO0 (TB), acquire static signal from AI0 (SUT), compare results.
2. test_dio_static: generate static signal on DO0 (TB), acquire static signal from DI0 (SUT), compare results.
3. test_aio_wf: generate a sine wave on AO0 (TB) on the trigger signal PF11, acquire the waveform on AI0 (SUT) on the trigger signal PF11, compare results.

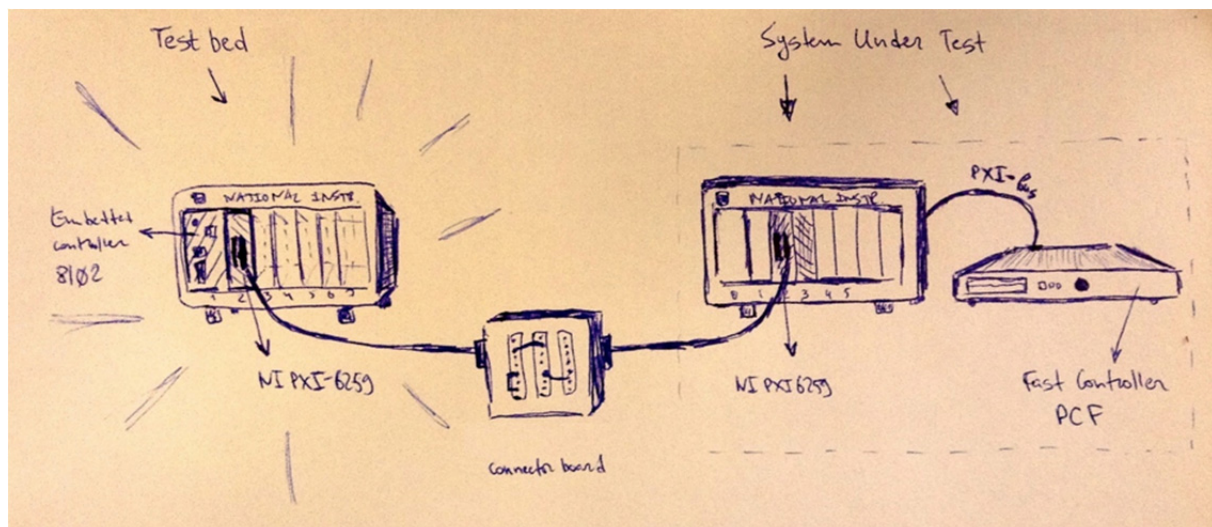


Figure 1: TestBed chassis is attached to the system under test.

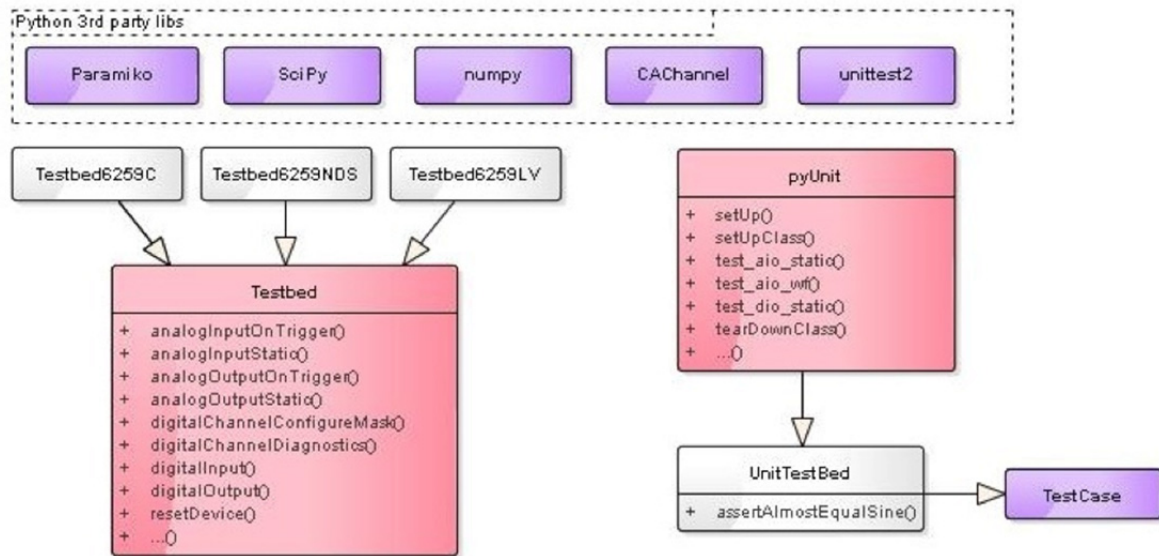


Figure 2: Python class diagram.

IMPLEMENTATIONS

C Executables

The functionality of a DAQ board can be supported by providing the C executables. In this work we have used the NI PXI-6259 Linux Device Driver [2], which provides the support library API to be used for application development.

NDS

If EPICS device support is available for the selected DAQ, then you no longer have to deal with files (saving and parsing waveforms from files, transferring files via SSH, etc.), but with EPICS Channel Access [3] (CA) – a protocol that provides remote access to records and fields managed by IOC.

The process can be simplified if the device support is written using NDS. NDS generalizes EPICS device support for data acquisition and timing devices. It provides sets of interfaces, solutions and best practices of device integration for EPICS. In this case, the Python code becomes identical for all DAQ cards. Which means that the only thing you need to do is provide an automatic test script (e.g., using the unittest2 Python package).

LabVIEW

Often, there is no Linux driver that supports the full functionality of a DAQ board. NI DAQ cards, however, come with the NI-DAQmx driver, that does support everything. In this case, the developer can implement something similar to NDS in LabVIEW, thus generalizing data acquisition cards by running an IOC and exposing NDS-like EPICS PVs to CA clients (including TestBed).

SUMMARY

- The PXI chassis, Embedded Controller and DAQ board have been selected for the TB suite.
- Scientific Linux 6.3 and ESS CODAC v4.1 were installed on the Embedded Controller.
- The CODAC pxi6259 example programs (written in C), EPICS device support (NDS) were modified to provide basic functionality – DIO, AIO, triggers, reading/writing data from/to files (PVs).
- The TestBed class was written in Python, and wraps the C and NDS functionality of the NI-PXI6259 DAQ board.
- The TestBed was tested on ITER CODAC v4.2 (Red Hat Enterprise Linux 6.3) and ITER CODAC v4.1 (Scientific Linux 6.3).
- The resulting test framework makes it possible for automatic tests to be executed with each release of the CODAC control system, thus reducing effort and ensuring complete consistency and repeatability in the testing protocol.

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] V. Isaev, "Nominal Data Acquisition Device Support for EPICS", Proc. ICALEPCS2013, TUPPC059, <http://jacow.org/>.
- [2] K. Meyer *et al.*, "Design and implementation of Linux drivers for National Instruments IEEE 1588 Timing and General I/O cards", Proc. ICALEPCS2013, THPPC056, <http://jacow.org/>.
- [3] <http://www.aps.anl.gov/epics/docs/ca.php>

LAUNCHING THE FAIR TIMING SYSTEM WITH CRYRING

M. Kreider, J. Bai, R. Bär, D. Beck, A. Hahn, C. Prados, S. Rauch, W. W. Terpstra, M. Zweig
 GSI Helmholtzzentrum für Schwerionenforschung, D-64291 Darmstadt, Germany

Abstract

During the past two years, significant progress has been made on the development of the General Machine Timing (GMT) system for the upcoming FAIR facility at GSI. The primary features are time-synchronization of 2000-3000 nodes using the White Rabbit Precision Time Protocol (WR-PTP), distribution of International Atomic Time (TAI) timestamps and synchronized command and control of FAIR control system equipment.

A White Rabbit network has been set up connecting parts of the existing facility. A next generation of the Timing Master has been developed. Timing Receiver nodes in the form factors Scalable Control Unit (standard front-end controller for FAIR), VME, PCIe and standalone have been developed. CRYRING is the first machine on the GSI/FAIR campus to be operated with this new timing system and serves as a test-ground for the complete control system. Installation of equipment starts in late autumn 2014 followed by commissioning of equipment in winter.

INTRODUCTION

The primary task of the General Machine Timing (GMT) system is the hard real-time control of the GSI and FAIR accelerator complex with sub-ns precision [1]. This is a two-step process. First, the *Settings Management* [2] distributes multiple settings to concerned Front-End Computers (FECs) via the normal network of the accelerator. Activities¹ are prepared by the Front-End Software FESA [3]. Second, the GMT generates on-time actions at the FECs. Such an action triggers a prepared activity at the FEC and provides an index referencing one of the preloaded settings.

The fundamental idea behind the GMT is the concept of time-based control. The distribution of information and timely execution of activities are decoupled. As a prerequisite, all nodes of the GMT share a common notion of time provided by WR-PTP [4] via the dedicated White Rabbit network (timing network). The central component of the GMT is the Data Master (DM) [5]. At some time the DM receives an (updated) schedule recipe for the operation of the facility from the settings managements. This recipe only contains indices to and time intervals between actions. Based on this recipe, the DM controls the facility in hard real-time. This is again a two-step process. First, the DM broadcasts timing messages including the indices, but with absolute execution time stamps, via the timing network to the Timing Receivers (TR) embedded in the FECs. The messages must be distributed with an upper bound latency. Second, messages are received by the TRs where they are filtered. Relevant messages remain pending until the specified execu-

tion time when the TR performs the action. Depending on the configuration of the TR, such an action could be digital signal generation, complex activity such as ramping a radio-frequency system, or signaling an event to the front-end software via an interrupt request (IRQ).

In 2014 the primary features have been implemented in such a way that the GMT and other components of the FAIR control system can be used coherently for the control of a real machine like CRYRING. This synchrotron has in the meantime moved from its original site in Stockholm to GSI and is presently installed in a refurbished cave behind the existing Experimental Storage Ring (ESR). While the infrastructure for the installation of the control system is presently being completed, the GMT components relevant for the recommissioning of CRYRING have been implemented and tested. This paper reports on the on-going work and summarizes the present situation.

ASTERISK

For the nodes of the GMT, the timing team at GSI supports a variety of hardware types and functionality. The interfaces of the GMT provide a common “look and feel” to the users, hiding the complexity and differences between the form factors to a large extent. Although some features (e.g. a display) may not exist on all form factors, common functionality must be presented identically at the interfaces.

To guarantee these requirements, the timing team builds releases. Such a release includes hardware, gateway (FPGA code), firmware (embedded CPU code) as well as software (host system code) in a consistent way. The first version, named *Asterisk*, has been released in July 2014. It includes all features of the GMT required for the next milestones of the whole FAIR control system.

Hardware

Asterisk includes four form factors. The Scalable Control Unit (SCU) is the standard FEC for the FAIR control system and has been developed by the hardware section of the control system department [6]. Three other form factors have been developed by the department of Experiment Electronics and are intended for usage by the department of Beam Instrumentation as well as Data Acquisition (DAQ) systems of FAIR experiments. The most important one is the PCIe module PEXARIA5, since this module represents the reference implementation based on an ARRIA V FPGA for all form factors provided through the GSI timing team. The two remaining modules, the VME board VETAR and the standalone form factor EXPLODER are still based on ARRIA II FPGAs.

¹ Example: Ramping of a magnet.

Gateware

Gateware is synthesized Hardware Description Language (VHDL) code for Field Programmable Gate Arrays (FPGAs). For the GMT, the gateware is based on a Wishbone System-on-Chip architecture [7]. This way, the functionality can be clearly separated and implemented by dedicated Wishbone master or slave devices attached to a hierarchy of several Wishbone crossbars. Each Wishbone device is identified by vendor ID, device ID as well as major and minor revision required for the implementation of a Self-Describing Bus (SDB) record. The connection to host bus system or Ethernet are provided by Wishbone masters.

Typical components addressed by users would be: A Timestamp Latch Unit (TLU) allows timestamping of incoming digital signals with 1 ns precision. An Event-Condition-Action (ECA) unit filters incoming timing messages [6]: Relevant messages are transferred to so-called action channels, where they are sorted according their execution time [8]. On-time, the ECA spits out the message data with a granularity of 8 ns to Receiving Components (RC), that are also implemented in VHDL. Examples of RCs are a message queue required for IRQ handling to the host system, or General Purpose IO (GPIO) that allows output of digital signals with a 1 ns granularity. Of course, the gateware includes the WR-PTP core required for time synchronization.

An important development has been the implementation of an on-chip CPU cluster of Lattice Micro 32 (LM32) softcores [9]. The cluster is connected to the Wishbone bus and implements shared memory and Message Signaled Interrupts (MSI) for synchronization.

Software

The main software components within *Asterisk* include a kernel driver specific for each host bus system, a generic Wishbone kernel driver and the userland Etherbone [10] API. This is sufficient to provide transparent access to the on-chip Wishbone devices from userspace. The supported interfaces are PCIe, VME, USB and UDP/Ethernet. This is complemented by software tools and libraries for specific Wishbone devices like a flash controller, the WR-PTP core or the ECA.

Data Master

The Data Master is in charge of command and control of the FAIR accelerator complex in hard real-time. It is implemented as a hybrid system. A high-end industrial PC serves as an interface to the settings management and other components of the control system. A PEXARIA PCIe module serves as the main hardware component. Its key feature is a FPGA hosting a multi-core cluster of LM32 softcores. Distinct processes on these cores are each in charge of real-time generation of timing messages of a particular part of the accelerator complex. A dedicated tool chain allows configuration of the DM with a schedule recipe in XML format and control of the operational state. Only one instance of

the Data Master will be used for CRYRING and later-on for FAIR. For more details see [5].

Timing Network

The timing network is composed of commercial White Rabbit switches. For the start of CRYRING, only top down timing messages from the Data Master are allowed next to WR-PTP. These messages are forwarded by the switches to the nodes via cut-through routing. As there is no other traffic, the latency of a switch is on the order of a few μ s.

Timing Receiver Nodes

Asterisk supports four TR types. EXPLODER is a standalone TR for digital I/O and allows configuration via USB or via the timing network. VETAR and PEXARIA are VME and PCIe modules. They can be configured via their host system bus and support IRQs. The most important for equipment control is the SCU. This is an embedded system and operated in custom crates of 3U height. It includes a SCU carrier board with on board TR, dedicated piggy boards, and a Com-Express module hosting the front-end software. As a very important feature, it provides a connection to the slave modules in the same crate via the so-called SCU-bus on the backplane. Slave modules implement I/O to external hardware like power supplies or radio-frequency systems.

INTEGRATION WITH THE OVERALL CONTROL SYSTEM

CRYRING

CRYRING is the first system that requires an integration of all components of the FAIR control system. Towards the application layer and settings management, the Data Master has implemented a prototype FESA class.

At the TRs, a first version of an interface towards FESA has been implemented. Frankly speaking, this is only a hack in FESA core. However, it already allows configuring the TR and to receive IRQ and the data associated to a timed action of the ECA. As a demonstration, a FESA class is implemented which, first, receives all ECA actions received via the host bus system and, second, publishes relevant information via the Controls MiddleWare (CMW) [11]. This information about actions triggered by the GMT is distributed and available to all applications of the control system.

The department of Beam Instrumentation has successfully integrated TRs provided by the GMT into PCIe and VME systems. Three FESA classes have been implemented demonstrating the integration of GMT, FESA and control system equipment in FECs [12].

INTEGRATION WITH DAQ SYSTEMS

The timing receivers developed for the GMT have two features that are of interest for DAQ systems.

The timestamp latch unit (TLU) allows timestamping with a granularity of 1 ns. This feature is of interest for timestamping events or even detector signals. This can be applied

to DAQ systems with high trigger rates exceeding 100 kHz. Long term tests over weeks have been passed successfully. This demonstrates the robustness of software, drivers and SoC Wishbone architecture of the TR.

Another feature is clock and timestamp fan-out provided by the form factors PCIe, VME and standalone. A 200 MHz clock is phase locked between all TRs connected to the timing network. A 100 kHz clock is phase locked to the 200 MHz clock and provides encoded timestamps that are sent in between the 100 kHz clock ticks. Due to the distributed nature of the GMT, this feature allows not only for timestamping but, when combined with dedicated electronics, for the measurement of time differences with a precision in the two digit picoseconds range. As an example, this can be used for Time-of-Flight measurements for particle identification.

For more details on the integration of GMT and DAQ please refer to [13].

ISSUES

Although the integration of DAQ and GMT systems shows high stability even on long term operation, this picture changes as soon as additional Ethernet traffic is allowed in the timing network. A loss of White Rabbit lock has also been observed by other users of White Rabbit elsewhere and has been investigated by our colleagues from CERN. It has been reported, that this issue is improved with a new release of the switches software and gateway [14].

The present implementation of front-end software typically claims a resource of the TR exclusively. This results in a failure to share unique resources like IRQs between two or more userland application. Along the same line, access to digital I/O on TRs or access to the SCU backplane bus can effectively not be shared amongst different applications. These and other issues triggered the development of a software framework, codenamed *SaftLib*, which is presently being defined.

Some important features of the GMT are not yet implemented. Examples are *robustness* of timing messages, *redundancy* of the timing network and *priority encoding* in White Rabbit switches.

CONCLUSION AND OUTLOOK

Important features of the GMT have been implemented. A Data Master exists. The SCU is integrated. TRs in the form factor PCIe and VME have been developed and integrated in the FECs of the Beam Instrumentation group as well as DAQ systems. The GMT is *ready for installation* and integration with CRYRING equipment will be started once the required infrastructure, such as power, racks, network and cable trays, has been installed at CRYRING.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of the CERN White Rabbit Team, the driving force behind the development of White Rabbit PTP. Furthermore the authors would like to thank our GSI colleagues from the departments of Beam Instrumentation, Experiment Electronics and Radio-Frequency for their help. We thank Peter Moritz (†) and Sabine Voltz (†).

REFERENCES

- [1] D. Beck et al., “The New White Rabbit Based Timing System for the FAIR Facility”, FRIA01, Proceedings of PCaPAC (2012) Kolkata, India.
- [2] J. Fitzek et al., “Settings Management within the FAIR Control System Based on the CERN LSA Framework”, WEPL008, Proceedings of PCaPAC (2010) Saskatoon, Canada.
- [3] Al. Schwinn et al., “FESA3 - The New Front-End Software Framework at CERN and the FAIR Facility”, WECOA03, Proceedings of PCaPAC (2010) Saskatoon, Canada.
- [4] J. Serrano et al., “The White Rabbit Project”, TUC004, Proceedings of ICALEPCS (2009) Kobe, Japan, 2009.
- [5] M. Kreider et al., “New developments on the FAIR Timing Master”, FPO022, *These Proceedings*, PCaPAC (2014) Karlsruhe, Germany.
- [6] S. Rauch et al., “Facility Wide-Synchronization of Standard FAIR Equipment Controllers”, WEPD48, Proceedings of PCaPAC (2012) Kolkata, India.
- [7] Wishbone B4, specification: http://cdn.opencores.org/downloads/wbspec_b4.pdf
- [8] W.W. Terspra et al., “Inexpensive Scheduling in FPGAs”, TCO301, *These Proceedings*, PCaPAC (2014) Karlsruhe, Germany.
- [9] W.W. Terpstra, “The Case For Soft-CPU in Accelerator Control Systems”, THCHMUST05, Proceedings of ICALEPCS (2011) Grenoble, France.
- [10] M. Kreider et al., “Open Borders for System-on-a-Chip Buses: A Wire Format for Connecting Large Physics Controls”, Phys. Rev. ST Accel. Beams 15 (2012) 082801.
- [11] V. Rapp et al., “Controls Middleware for FAIR”, WCO102, *These Proceedings*, PCaPAC (2014) Karlsruhe, Germany.
- [12] H. Bräuning, GSI (2014) private communication.
- [13] N. Kurz et al., “White Rabbit Applications for FAIR Experiments”, GSI Scientific Report 2013 (2014) to be published.
- [14] Eighth White Rabbit Workshop, Geneva, 2014: <http://www.ohwr.org/projects/white-rabbit/wiki/Oct2014Meeting>

TCP/IP CONTROL SYSTEM INTERFACE DEVELOPMENT USING MICROCHIP BRAND MICROCONTROLLERS*

Christopher E. Peters[#], Maria A. Power, ANL, Argonne, IL 60439, USA

Abstract

Even as the diversity and capabilities of Single-Board Computers (SBCs) like the Raspberry Pi and BeagleBoard continue to increase, low level microprocessor solutions also offer the possibility of robust distributed control system interfaces. Since they can be smaller and cheaper than even the least expensive SBC, they are easily integrated directly onto printed circuit boards either via direct mount or pre-installed headers. The ever increasing flash-memory capacity and processing clock speeds has enabled these types of microprocessors to handle even relatively complex tasks such as management of a full TCP/IP software and hardware stack. The purpose of this work is to demonstrate several different implementation scenarios wherein a computer control system can communicate directly with an off-the-shelf Microchip brand microcontroller and its associated peripherals. The microprocessor can act as a Hardware-to-Ethernet communication bridge and provide services such as distributed reading and writing of analog and digital values, webpage serving, simple network monitoring and others to any custom electronics solution.

MOTIVATION AND THEORY

The Argonne Tandem Linac Accelerator System (ATLAS) 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 research in order to perform physics analysis of the properties of the nucleus. In support of this goal, the accelerator control system and its electronics hardware have been in a state of continuous upgrade since its transition to computer control in the late 1980s [1]. In addition, the beam transport hardware and support electronics equipment has been in a constant state of change. The interfaces into the control system range from simple 0-10v analog input signals, to 50+ bit binary coded decimal, to all manner of digital communication protocols (RS-232/485, GPIB, USB, SNMP, ModBUS).

These signal types can require inefficient cable solutions, sometimes requiring hundreds of separate conductors to transmit relatively slow changing (sub ~10Hz) and relatively low accuracy requirement (2-4 significant figures) beam transport control channel points. Specifically, these types of channels stand in contrast to the high speed, high accuracy data acquisition and timing channels used by the target and detectors groups.

The result is a mismatch between the modest channel speed and accuracy requirements and the high cost of implementation. The goal of this work is not only to define a low cost control system hardware interface, but also to push that platform lower down into the custom hardware components currently developed at ATLAS (Figure 1). This will provide better focus for hardware designers to only work directly on electronics separate from the high level interface, and allows the controls group to rely more on generic and reusable interfaces.

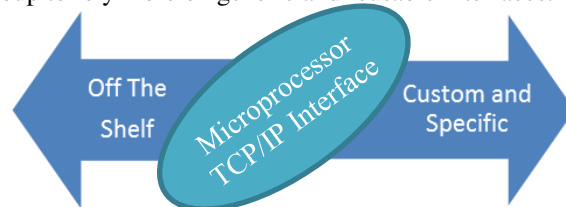


Figure 1: Microprocessor Based Interfaces Exist in the Middle of the Spectrum of Hardware Design Complexity.

MICROPROCESSOR COMPARISON TO SINGLE BOARD COMPUTERS

The goal of this work is to investigate the advantages and disadvantages of using the latest in microprocessor based platforms in a distributed control system environment. Single Board Computers (SBCs) have increasingly been viewed as a way to add various communication protocols [2] to hardware. These devices can interface with simple electronics via on-board ADCs and PWM/DACs, and also act as complex data processing platforms in their own right. This paper specifically focuses on the Microchip brand [3] and the OEM produced PIC32 Ethernet Starter Kit (PIC32 or ESK).

It is often difficult to directly compare the offerings of common SBCs like the BeagleBoard Black [4], the Raspberry Pi [5], and Industrial PCs like PC/104 [6] and microcontrollers, since the selection of each platform is often highly dependent on the application. Our goal is to determine if the more limited and focused capabilities of microprocessors are matched well to implementation into low-level custom built beam transport electronic devices.

PRODUCTS AND INITIAL EXPERIENCES

The initial steps in this research were to determine what possible COTS offerings were available to push TCP/IP communication down into the hardware design level. Initial experimentation was performed using a starter kit, however all components are available for installation on any custom developed PCB hardware.

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. *microchip.com #ChrisPeters@anl.gov

Current Baseline Product

Microchip's Ethernet Starter Kit was chosen as the test bed for this work due to the company's reputation for field support and previous experience with the authors. This kit comes in the form of a small form factor PCB board with an Ethernet port and several debugging tools (Figure 2). One such tool is an on-board USB interface for PC-USB debugging. There is also a board-to-board high density plug on the bottom, for the addition of optional expansion and I/O breakout boards.

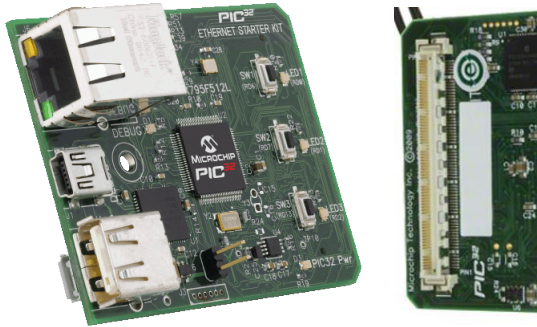


Figure 2: Microchip Ethernet Starter Kit [2] & Connector.

Preloaded Software and Services

One of the major advantages to working with products from large OEMs is in the form of bundled software libraries. In the case of Microchip, these libraries are offered open source, and with a free license. There are also prewritten API routines, similar to those which come with SBC OS distributions, which can operate UARTs, SPI, I2C, and USB busses with the minimum of code implementation from the end user. This code base is bundled along with a compiler and debugging tools into Microchip's MPLAB IDE at no additional cost [7].

IMPLEMENTATION AND SENARIOS

Web Browser/HTTP and AJAX Connections

The starter kit comes preloaded with an HTTP server. Using this server, it is possible to create HTML elements on the client side which connect asynchronously to the PIC32 via Asynchronous Java and XML (AJAX) calls. This way, a very simple but user-friendly interface can be created (Figure 3) for debugging or testing offline. User manuals or schematics can also be stored in non-volatile memory on the chip for documentation purposes.

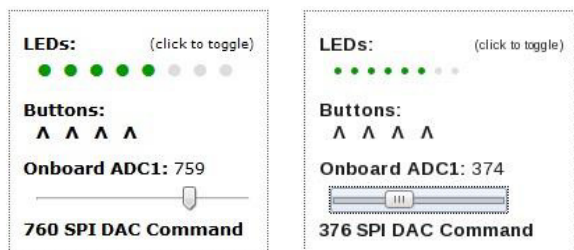


Figure 3: HTML Range Slider communicating via AJAX in (Left) Windows and (Right) Linux Firefox Browsers. The slider sets an external DAC, and reads back the ADC.

Low Level Protocols like SNMP

The starter kit also comes with standard SNMP libraries. SNMP has been demonstrated in the past to be useful in a control system environment [8], and can be additionally leveraged as a common control system protocol. At ATLAS, low level SNMP handlers were implemented into local control system libraries and were then able to communicate reliably with the PIC32. Standard operator display pages were created (Figure 4) with SNMP executing asynchronously in the background and a seamless interface to the operators was presented.

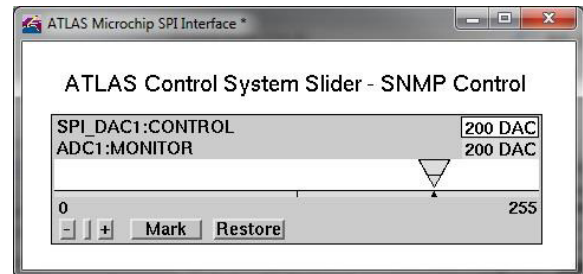


Figure 4: ATLAS Standard Slider interfaced to Microchip snmpset:MICROCHIP-MIB::SpiDAC1= INTEGER: 200 snmpget:MICROCHIP-MIB::analogPot=INTEGER: 200.

Higher Level Protocols like Channel Access

In order to demonstrate capabilities similar to those of full SBCs, it was desired to implement some level of EPICS integration into the PIC32. However, EPICS libraries are mostly C++ and contain a significant codebase dedicated to multitasking and thread-safety. Therefore, it would be difficult to directly port standard EPICS libraries to the Microchip compiler. Instead, the EPICS Channel Access Protocol Reference [9] was used to implement a very small subset of low level TCP/UDP commands which the PIC32 can respond to. This has enabled at least a first pass at integration with a higher level control system.

Cost Comparisons

The below (Table 1) provides a cost comparison for assemblies like SBCs and PIC32 development kits compared to individual component parts of a custom microprocessor hardware design. Assuming that the PIC32, physical Ethernet (PHY) chip, and Ethernet socket are in addition to an existing circuit board design, the total additional cost is under 25 USD. This does not include development costs and other electrical components.

Table 1: Development Kits and Component Costs

Component	Cost
PIC32 Ethernet Starter Kit (ESK)	70 USD
ESK I/O Expansion Board (Optional)	70 USD
PIC32MX795F512 Processor Only	11 USD
TI DP83848C PHY Chip	6 USD
MagJack Ethernet Plug SI-60000	5 USD
Beagleboard Black	55 USD
Raspberry PI Ver-B	35 USD

PERFORMANCE

Any control logic duties of the microprocessor such as input filtering, I/O transactions, alarming, etc., can only exist in addition to the existing low level interface code. There are several restrictions which must be considered before using a microprocessor for control interfaces. The first restriction is on code size (static, EEPROM) and dynamic (stack, heap) memory sizes. (Figure 5) shows the amount of remaining memory with all HTTP and SNMP services running, in addition to a large amount of test services. This leaves 342kB available for control system logic, and 95kB free for live RAM scratch space. While this would be considered impossibly restrictive in the SBC realm, microprocessor operations are normally focused and hierarchal in nature. A simple Ethernet-to-SPI bridge (TCP/IP to SPI DAC output directly inside the PCB hardware) would only require a few kB of code.

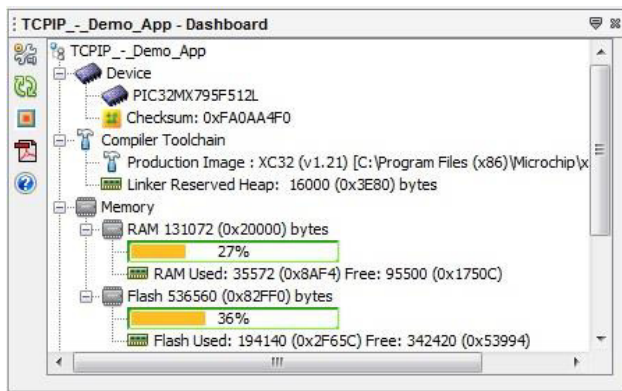


Figure 5: Test Memory Usage with HTTP, SNMP, and EPICS CA Services fully compiled, using MPLAB.

The PIC32 tested during this work contains an on-board Ethernet controller which communicates via direct-memory-access (DMA) to an external PHY chip. In this way, moving data to the Ethernet subsystem does not use CPU instructions, making transfers extremely fast. The demo board in this work was able to send TCP data at a rate of over 1 MByte per second to a remote host. This speed was reduced only slightly by actively sending simultaneous HTTP and SNMP requests and responses.

CONCLUSION AND FUTURE WORK

There are several additional steps to be undertaken in order to fully qualify the concept of a generic microcontroller-based control system interface. First and foremost, within this work only a demonstration kit has been tested. The next iteration should be installed directly inside on an existing PCB along with the required oscillators, filter capacitors and power supplies. Another option is to develop an ATLAS specific interface PCB which uses the same PIC32 processor and components, but has custom headers which are already used by the electronics development group. In this way, a control system microprocessor kit specific to ATLAS could be sourced and then re-used on any available hardware project by the Electronics groups.

ISBN 978-3-95450-146-5

Table 2: Microprocessor Advantages and Disadvantages

Feature	Advantages	Disadvantages
Low-level on board HW communication	Configurable pin-outs speak a wide variety of HW level protocols.	Challenge to create a single standard interface design.
Small, Low Cost and native real time.	Hardware Engineers can integrate directly into end products.	Initial work for custom PCBs to distribute signals
Few Running Application sections and services	Low security risk & risk of bad behaviour as a result of complex operating system.	Requires some knowledge of Firmware development.
Increases in memory and processor speed	Can support TCP/IP stack, and has OEM software libraries	Less memory and processing power compared to SBCs.

A successful end result would push control system development towards an “Internet of Things” concept [10], whereby every device in the system has an integrated Ethernet plug and its own IP address. This greatly reduces cable overhead cost and increases the capabilities by which all devices can communicate (Table 2). Common security concerns like unintended access and outdated operating system patches can be mitigated by the greatly reduced code base and the targeted Ethernet capabilities of a microprocessor. While these features are not always distinct from a single board computer, the microprocessor as a concept is much more able to be integrated directly into the actual circuit boards of control hardware devices and has the dual advantage of lower component cost and zero unneeded operating system services providing additional networking vulnerabilities.

REFERENCES

- [1] F. Munson, D. Quock, B. Chapin, and J. Figueroa, “Argonne’s ATLAS Control System Upgrade”, ICALEPCS ‘99, Trieste, Italy, October 4-8, 1999.
- [2] S. Cleva, A. Bogani, L. Pivetta, “A Low-Cost High-Performance Embedded Platform for Accelerator Controls”, Proceedings of PCaPAC2012, Kolkata, India, 2012.
- [3] <http://www.microchip.com>
- [4] <http://beagleboard.org/Products/BeagleBone+Black>
- [5] <http://www.raspberrypi.org/>
- [6] <http://www.pc104.org/specifications.php>
- [7] <http://www.microchip.com/pagehandler/en-us/devtools/mla/home.html>
- [8] C. Peters, M. Power, “Distributed Network Monitoring Made Easy - An Application for Accelerator Control System Process Monitoring”, ICALEPCS2013, San Francisco, CA, 2013.
- [9] <http://www.aps.anl.gov/epics/docs/Caproto.html>
- [10] Evans, Dave, “The Internet of Things”, Internal Cisco Whitepaper, April 2011, http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

THE ROLE OF THE CEBAF ELEMENT DATABASE IN COMMISSIONING THE 12 GeV ACCELERATOR UPGRADE*

T. Larrieu, M. Joyce, M. Keese, C. Slominski, D. Turner, JLAB, Newport News, VA 23606, U.S.A.

Abstract

The CEBAF Element Database (CED) was first developed in 2010 as a resource to support model-driven configuration of the Jefferson Lab Continuous Electron Beam Accelerator (CEBAF). Since that time, its uniquely flexible schema design, robust programming interface, and support for multiple concurrent versions has permitted it to evolve into a more broadly useful operational and control system tool.

The CED played a critical role before and during the 2013 startup and commissioning of CEBAF following its 18-month long shutdown and upgrade. Information in the CED about hardware components and their relations to one-another facilitated a thorough Hot Checkout process involving more than 18,000 system checks.

New software relies on the CED to generate EDM screens for operators on-demand thereby ensuring that the information on those screens is correct and up-to-date. The CED also continues to fulfil its original mission of supporting model-driven accelerator setup. Using the new ced2elegant and eDT (elegant Download Tool), accelerator physicists have proven able to compute and apply energy-dependent set points with greater efficiency than ever before.

BACKGROUND

The Jefferson Lab CEBAF accelerator is a superconducting recirculating linear accelerator capable of delivering continuous wave electron beams simultaneously to multiple experimental halls. Previously capable of delivering 6GeV, CEBAF recently underwent an extensive multi-year \$338M upgrade project to double its energy capacity to 12GeV and add a fourth experimental hall (fig. 1). The upgraded accelerator was commissioned successfully between November 2013 and May 2014. The CEBAF Element Database (CED) was an indispensable tool during the commissioning.

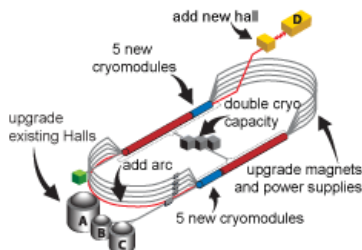


Figure 1: The CEBAF 12GeV Upgrade Project.

* Notice: Authored by Jefferson Science Associates, LLC under U.S. DOE Contract No. DE-AC05-06OR23177. The U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce this manuscript for U.S. Government purposes.

DATABASE DESIGN

The CED design is based upon a modified Entity-Attribute-Value with Classes and Relationships (EAV/CR) data model [1]. Whereas in a traditional schema adding support for new accelerator hardware would involve adding additional tables and columns to the database, the EAV/CR data model employed by the CED is introspective – defining a new class of accelerator hardware in the CED simply involves adding rows to the already-existing metadata “catalog” tables. Once defined in the catalog, existing software is fully capable of interacting with the new entities after discovering their properties from the metadata tables.

A major benefit of the EAV/CR data model is that its static schema is well suited to integration with Oracle Workspace Manager [2] to provide timestamp-based versioning of table rows, named save points, and multiple independent workspaces (akin to branches in a software version control system).

DATABASE IMPLEMENTATION

Platform

At Jefferson Lab, the CED is implemented using version 11g Standard Edition Oracle Relational Database. The database server runs Redhat Enterprise Linux 6 on x86_64 hardware. The current API library is compiled solely for 32-bit x86 architecture, though prior versions were also compiled for the Solaris 10 Sparc and HP-UX PA-RISC platforms.

Layout

To optimize performance for the different use cases, the CED database instance is distributed among three database users/schemas as shown in Figure 2. The operational schema stores the current machine configuration and is optimized for performance. The historical schema provides efficient storage and access to (read-only) historical save points (snapshots). And the development schema is used to create workspace branches where data can be edited and prepared before being promoted to the operational schema.

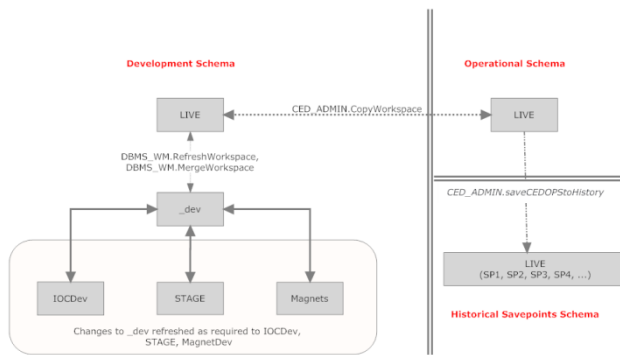


Figure 2: The CED three schema layout.

The details of the three schema layout are transparent to non-administrative users who need simply include an optional workspace or savepoint name when interacting with the database.

Configuration Control

As a rule, users do not directly edit data in the operational schema. Instead, updates are prepared in a development workspace and merged into OPS upon request. A database administrator receiving a merge request will audit the proposed changes for validity and then promote the changes to the operational schema. Administrators make use of a web-based management tool to automate the auditing and merging process as well as the creation of new savepoints.

API

The generalized nature of the EAV/CR data model employed by CED requires a robust API to interpret the contents of the metadata tables and translate the abstract database storage into recognizable attributes useful to programs and users. The API is also responsible for enforcing much of the data validation and domain logic embodied in the metadata catalog.

For the CED, a shared library was written in C++ to be the sole intermediary for applications that will access its information. Native versions of the API are available not only to C++ programs, but also to scripts written in Perl, PHP, and Tcl. The script language versions are generated automatically from the original library via the open source SWIG (Simplified Wrapper and Interface Generator) tool [3,4].

On top of the core API, two general purpose user interfaces to the CED have also been built around the C++ library and its scripting language derivatives: a RESTful web interface and a full-featured command-line tool usable from JLAB Linux workstations.

CED Event Server

The CED Event Server provides a means to link control system actions to database updates. Whenever rows are modified in the operational schema, a database trigger notifies the event server of the changes. Based upon rules in its configuration file, the event server can take actions including: telling long-running daemon processes to refresh their configurations, sending email to system

owners, making logbook entries, or calling the ced2Epics utility to update pvs in the control system. The CED Event Server will even notify users if an auto-generated screen on their desktop needs to be refreshed because of a database update.

HOT CHECKOUT

Readiness

The immense scope of the work completed during the 18 months the lab was shut down necessitated a thorough system whereby all new, refurbished, and even supposedly unmodified systems could be verified as ready to be operated safely.

The CED was used as the underpinning for a web-based Hot Checkout (HCO) tool [5]. This tool tracked HCO readiness status on a daily basis in the months leading up to commencement of beam operations. Geographic information from the CED granted users of the HCO tool an ability to determine readiness status region by region.

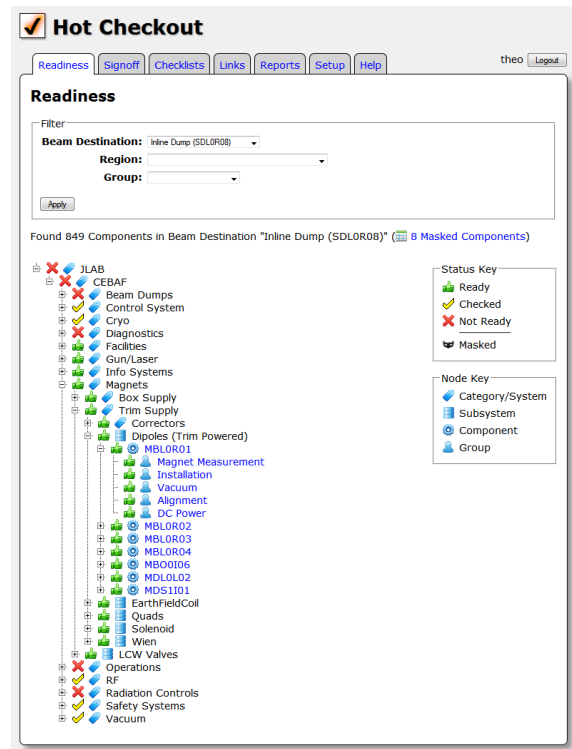


Figure 3: Web-based Hot Checkout application.

CED-linked Hardware Labels

New hardware labelling was devised that utilizes the CED to provide detail that can't fit on the label itself. In addition to the key details such as name, weight, power supply, etc., the new labels contain a web QR code that can be scanned with camera-equipped tablets or smart phones to call a CED-linked web page. For example, the web page accessed from a Dipole label (see figure 4) would allow the user to find out which other nearby

hardware shares the same cooling water valves or power supply.



Figure 4: Hardware label with QR code linked to the CED.

EDM SCREEN GENERATION

The display manager used in the CEBAF control room is EDM (Extensible Display Manager). For many large hardware systems, including magnets and vacuum, the process of building and maintaining screens by hand has been replaced by programmatic screen generation. A large percentage of the screens used by CEBAF operators no longer exist as static .edl files, but are generated on-the-fly using data from the CED. This process liberates EPICS programmers from the burden of creating screens for new device instances and also ensures that the screens always match the current operational configuration.

In addition to the screens that are placed on menus for operators, a command line tool permits users to create task specific on-the-fly screens for many systems based on CED filters. For instance users may request screens that show non-contiguous regions or that group elements in a non-standard way (e.g. by power supply, by rack, by building, by controlling IOC, etc.).

ELEGANT DOWNLOAD TOOL

The ced2Elegant and Elegant Download Tool (eDT) software tools were developed at JLab as the mechanism whereby the design set points stored in the CED can be used to perform accelerator set up. First, the ced2Elegant utility is executed to retrieve design BdL values from the CED and produce an Elegant [6] lattice. Then eDT is rung to calculate new energy-specific BdL values by using Elegant to compute the momentum at each element in the lattice. At the end of the process, a BURT "snap file" containing the BdL values to be sent to the control system is written out. This process is faster and more robust than the previous method of manual scaling of an historical machine save followed by substantial operator tuning in order to achieve a stable orbit. The database-driven approach also lends itself to ongoing improvement – for example, a recent enhancement to eDT compensates for synchrotron radiation losses that occur at higher energy configurations.

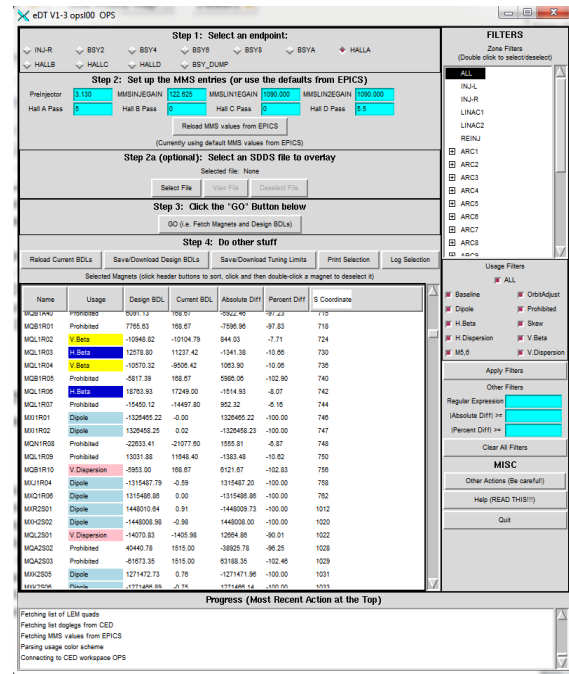


Figure 5: The Elegant Download Tool uses design data from the CED to generate downloadable energy-specific machine settings.

CONCLUSION

The CED has proven its utility during installation and commissioning of the CEBAF 12GeV upgrade project. Its flexibility and the growing suite of tools that have been built upon it ensure that it will play a central role in efficient and safe future operation of the upgraded accelerator.

REFERENCES

- [1] Nadkarni, MD, Prakash M.; Marengo, MD, Luis; Chen, MD, Roland; Skoufos, PhD, Emmanouil; Shepherd, MD, DPhil, Gordon; Miller, MD, PhD, Perry (1999), "Organization of Heterogeneous Scientific Data Using the EAV/CR Representation", Journal of the American Medical Informatics Association 6 (6): 478–493, PMID 10579606.
- [2] <http://www.oracle.com/technetwork/database/twp-appdev-workspace-manager-11g-128289.pdf>
- [3] Beazley, David M (1996), "SWIG : An Easy to Use Tool For Integrating Scripting Languages with C and C++", Proceedings of the Fourth Annual Tcl/Tk workshop, Monterey, CA July 6-10.
- [4] <http://www.swig.org/>.
- [5] Baggett, K. (2014, October), "New and Improved": The JLab (state-of-the-art) HCO System, WAO2014, Workshop on Accelerator Operations in Mainz, Germany, 2014.
- [6] http://www.aps.anl.gov/Accelerator_Systems_Division/Accelerator_Operations_Physics/elegant.html

InfiniBand INTERCONNECTS FOR HIGH-THROUGHPUT DATA ACQUISITION IN A TANGO ENVIRONMENT

T. Dritschler^{*}, S. Chilingaryan^{*}, T. Farago[†], A. Kopmann^{*}, M. Vogelgesang^{*}

^{*}Institute for Data Processing and Electronics

[†]Institute for Photon Science and Synchrotron Radiation

Karlsruhe Institute of Technology, Germany

Abstract

Advances in computational performance allow for fast image-based control. To realize efficient control loops in a distributed experiment setup, large amounts of data need to be transferred, requiring high-throughput networks with low latencies. In the European synchrotron community, TANGO has become one of the prevalent tools to remotely control hardware and processes. In order to improve the data bandwidth and latency in a TANGO network, we realized a secondary data channel based on native InfiniBand communication. This data channel is implemented as part of a TANGO device and by itself is independent of the main TANGO network communication. TANGO mechanisms are used for configuration, thus the data channel can be used by any TANGO-based software that implements the corresponding interfaces. First results show, that we can achieve a maximum bandwidth of 30 Gb/s which is close to the theoretical maximum of 32 Gb/s, possible with our 4xQDR InfiniBand test network, with average latencies as low as 6 μ s. This means that we are able to surpass the limitations of standard TCP/IP networks while retaining the TANGO control schemes, enabling high data throughput in a TANGO environment.

INTRODUCTION

PC-based systems have made significant advances in stability and computational power and have become viable for usage as control systems even in large installations. Driven by this influx of ‘off-the-shelf’ PC systems, the ESRF started to develop its own remote control system called TANGO [1] in 2003. TANGO provides transparent and uniform access to all devices on the control network. It uses the CORBA communication layer to send function calls to any remote device that provides a TANGO interface.

Since TANGO was created mainly as a control system for the synchrotron community and is distributed as an open source system, it found broad acceptance across the European synchrotron community and is now actively being developed by a large consortium, lead by ESRF. Eventually, the ANKA synchrotron at KIT has also decided to extensively use TANGO as one of the control system for their beamlines.

The new IMAGE beamline is being constructed at ANKA to allow ultra fast 3D tomography with near real-time monitoring and fast image-based control loops. The beamline development aims to permit investigations of the internal morphology and structural changes in small living organisms

in 4D (3D + time) with micrometer spatial resolution and sub-second time resolution [2]. The required resolution is achieved with high-speed cameras providing over a thousand frames per second and with streaming bandwidth ranging from a few hundred Megabytes up to multiple Gigabytes per second. The image-based control is made feasible by the UFO-framework [3]. Running on GPU-based computational servers, it is able to process a few Gigabytes of tomographic data per second. The efficient delivery of this data to the computation nodes, though, is a challenge.

To comply with ANKA standards, the control system for the IMAGE beamline [4] is based on TANGO to communicate with the beamline hardware and the controls for the pixel sensors/cameras are exposed using TANGO modules. Since TANGO transports all its data using CORBA with standard TCP/IP communication, its bandwidth is not only limited by the speed of the network adapter, but also by the performance of the TCP/IP stack, especially in terms of latency [5]. Also, the effective bandwidth is further limited by the particular implementation of CORBA (omniORB [6]) used by TANGO.

In this paper, we present our approach to extend the already existing TANGO infrastructure with a secondary high-bandwidth data channel. We use InfiniBand interconnects to avoid latency and bandwidth limitations of standard TCP/IP over Ethernet. A TANGO enabled camera driver was developed that allows to transport camera data through a secondary data channel based on InfiniBand RDMA communication. With this new data channel and the TANGO control system, coupled together, a control interface was created that replicates the remote TANGO camera interface into a local system while using InfiniBand as transfer for the camera data. This effectively combines high throughput data with the flexibility of TANGO while feeling and behaving like a purely local camera system.

DATA TRANSPORT PERFORMANCE

We evaluated the performance of TANGO and omniORB under certain scenarios. For this, a TANGO device server was written that generates pictures of 512x512 pixels (265 KB data) with a static gradient pattern and a region of random noise data in the center. These pictures can then be pulled from the server over the standard TANGO interface. We connect to this device server with a device proxy client that continuously pulls the picture attribute from the device server for a certain number of iterations and creates a mean value of the observed throughput by dividing the transferred

amount of data by the elapsed time. The amount of iterations were chosen to ensure that at least 1 GB of data gets transferred in total.

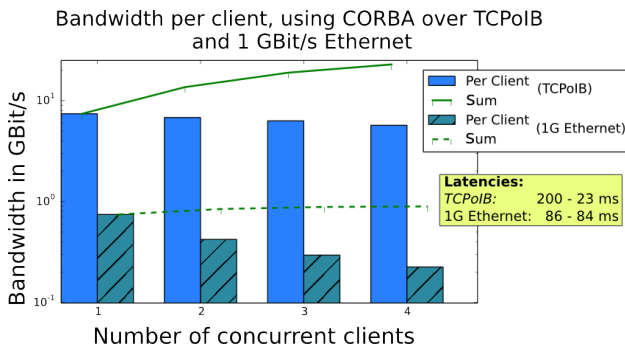


Figure 1: TANGO with omniORB performance using TCPoIB and 1 Gbit/s Ethernet.

We ran the test on a standard 1 Gbit/s Ethernet network with one switch. Our results showed, that omniORB efficiently uses the available bandwidth in case of a single client with up to 75% (750 Mbit/s) of the theoretical bandwidth and with up to 90% in sum (900 Mbits/s) in case of multiple clients. However, since standard Gigabit Ethernet does not provide enough bandwidth in order to transport our camera data with the required data rate of several Gigabytes per second, we need faster network adapters. Also, with latencies in the order of 85 ms for the standard Ethernet TCP/IP communication of omniORB, high frequency data transfer for high-framerate cameras is impossible.

Since InfiniBand interconnects are already widely used by the IMAGE beamline infrastructure we also investigated the performance of omniORB over InfiniBand interconnects. Modern InfiniBand adapters can easily reach bandwidths of up to 32 Gbit/s (4xQDR) with latencies as low as 2 μ s when using *native* InfiniBand communication protocols. However, since omniORB is not able to use native InfiniBand, we ran a performance test with a 4xQDR network using a TCP over InfiniBand stack (TCPoIB) instead. We present the results of our performance tests in Fig. 1. Here we can see that the efficiency of the data transmission is severely limited by the performance of TCPoIB, reaching only 7.4 Gbits/s out of the possible 32 Gbits/s (23%) with latencies ranging from 200 ms to 23 ms, in case of single client access. Per client efficiency is further limited with increasing amounts of concurrent clients. These results suggest, that *native* InfiniBand should be used as a dedicated data channel instead, to alleviate the problem of both bandwidth and latency.

SOFTWARE ARCHITECTURE

In order to separate between data acquisition and processing, and thus allow for easily interchangeable acquisition systems, the UFO infrastructure follows a distributed design. This makes it necessary to remote control the acquisition system over the network. To solve this problem, two pieces of software are already existing:

libuca

libuca is a unified camera control library with a generic interface to control different types of cameras on a local system. The specific implementations for communication and control of each camera type are encapsulated in the form of plugins. Each plugin is derived from a *libuca* camera base class that provides a well defined interface to perform the most common operations of any camera, such as ‘start/stop recording’ and ‘start/stop readout’ (in case of a cameras with an internal buffer). Besides this basic functional interface, each plugin provides a set of *properties* that can be queried and listed from the calling application. These properties are used to represent internal states and settings of each camera, like exposure time, sensor dimensions, region of interest, etc.

UcaDevice

A TANGO Server wrapper around *libuca*, called *UcaDevice*, was created to integrate supported cameras into the TANGO environment. In order to do so, *UcaDevice* creates an instance of the selected Uca camera plugin. It enumerates all properties published by the plugins and for each property a TANGO attribute is dynamically created and exposed into the TANGO network. A set of operations mirroring the basic Uca camera interface is published as well.

With these two pieces of software it is possible to remotely control the data acquisition over a TANGO network. However, we can not use the same TANGO communication scheme to also transport the acquired camera data due to the performance problems shown in the *Data transport performance* section. In order to extend the infrastructure by a high-throughput data channel that is able to handle the required data rates and low latencies we have extended *UcaDevice* with a dedicated data transport channel based on *native* InfiniBand communication. Our solution employs RDMA (Remote Direct Memory Access) and sustains throughput in the order of several Gigabytes per second. RDMA allows to transfer data directly into the destination machines RAM without additional I/O operations, which reduces latency significantly. To achieve this, we have created an InfiniBand communication library, called *KIRO*, and integrated it into to existing software infrastructure.

KIRO Library

The *KIRO* library is a networking library that provides an InfiniBand server and client class. The server locks a preallocated memory block for RDMA-Read access. Each connecting client, once access is granted, can read the locked memory region at any time without further involvement of the server. The RDMA communication is based on InfiniBand communication primitives and has virtually no transport or protocol overhead.

We added a *KIRO* server component to the *UcaDevice* wrapper and added new TANGO attributes to its interface which describe the address information of the *KIRO* server. The data acquisition mechanism of *UcaDevice* was changed

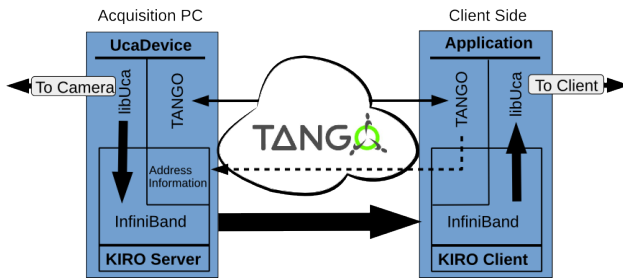


Figure 2: Overview of the software architecture for the InfiniBand data channel.

so it not only provides camera data over the TANGO interface but also via the KIRO server.

In addition, a libuca plugin for any generic TANGO/KIRO-enabled camera was created. The plugin connects to a given TANGO server and, once a connection is established, inquires the remote TANGO attributes to check for an available KIRO InfiniBand server address and port attribute and tries to establish this connection as well. All available attributes are queried from the TANGO server and are translated into internal libuca properties. This efficiently clones the discovered TANGO interface into the local libuca environment and provides a transparent remote interface for the libuca application. The remote camera then feels and behaves as if the camera was connected locally. If the KIRO connection setup was also successful, the plugin proceeds to use the established InfiniBand connection to pull data from the camera, while retaining the TANGO connection to remotely control the connected camera. Otherwise, the standard TANGO attribute for data transmission is used.

The architecture of the overall system is shown in Fig. 2. With this software architecture, we were able to establish a secondary data channel based on native InfiniBand communication between the camera and the processing PC. The UcaDevice server is used to generate a TANGO interface for the locally connected camera which is controlled via an appropriate libuca plugin. It also provides an instance of the KIRO Server to transport camera data.

On the receiving side, our client application loads libuca with the aforementioned KIRO-plugin. It gets directed to our UcaDevice TANGO server and clones the TANGO interface back into a local libuca interface while also establishing a KIRO InfiniBand connection for data transfer.

PERFORMANCE COMPARISON AND CONCLUSION

With this architecture we were able to achieve a bandwidth of 30 Gigabit/s out of the 32 Gigabit/s that are theoretically possible with our 4xQDR InfiniBand test network and average latencies of 6 μ s (see Fig. 3).

This shows that, in case of single client access, InfiniBand efficiency surpasses that of TCP/IP easily while also retain-

ing a significantly smaller latency. It does, though, also suffer from some limitations in case of concurrent client access. However, since the data channel using the InfiniBand connection is designed to be a dedicated data channel only for data transfer and is independent of the control communication, concurrent access becomes less of an issue for both communication channels.

This concludes that we are able to use the full efficiency of modern InfiniBand interconnects in a fully TANGO controlled environment while still retaining standard control schemes over TANGO with high-throughput and very low latencies.

Bandwidth comparison between CORBA over TCPoIB, and InfiniBand

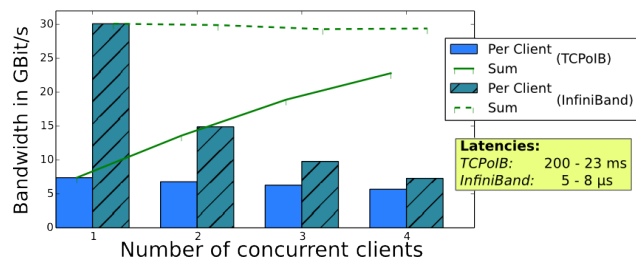


Figure 3: Comparing performance of omniORB over TCPoIB with InfiniBand.

REFERENCES

- [1] A. Götz et al., "TANGO is a CORBA based Control System", In Proc. ICALEPCS2003, Gyeongju, Korea, MP705, (2003).
- [2] Tomy dos Santos Rolo et al., "In vivo X-ray cinematography for tracking morphological dynamics", In Proceedings of the National Academy of Sciences of the United States of America (2014), pp. 3921–3926, doi: //10.1073/pnas.1308650111, <http://www.pnas.org/content/111/11/3921.full.pdf+html>, <http://www.pnas.org/content/111/11/3921>
- [3] M. Vogelgesang et al., "UFO: A Scalable GPU-based Image Processing Framework for On-line Monitoring", In Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications (HPCC-2012) & The 9th IEEE International Conference on Embedded Software and Systems (ICCESS-2012) pp. 824–829, doi: 10.1109/HPCC.2012.116
- [4] M. Vogelgesang et al., "When Hardware and Software Work in Concert", In Proc. ICALEPCS2013, San Francisco, CA, USA, TUPPC044, (2013).
- [5] Mellanox Technologies Inc., "InfiniBand and TCP in the Data Center", Document Number 2008WP, (2008).
- [6] D. Grisby et al., omniORB: Free CORBA ORB., <http://omniorb.sourceforge.net/>

PICOSECOND SAMPLING ELECTRONIC FOR TERAHERTZ SYNCHROTRON RADIATION

M. Caselle, M. Balzer, M. Brosi, S. Chilingaryan, T. Dritschler, V. Judin, A. Kopmann, A.-S. Müller, J. Raasch, L. Rota, L. Petzold, N. J. Smale, J., L. Steinmann, M. Vogelgesang, S. Wuensch, M. Siegel, M. Weber KIT, Karlsruhe, Germany

Abstract

The ANKA storage ring generates brilliant coherent synchrotron radiation (CSR) in the THz range due to a dedicated low- α_c -optics with reduced bunch length. At higher electron currents the radiation is not stable but is emitted in powerful bursts caused by micro-bunching instabilities. This intense THz radiation is attractive for users. However, due to the power fluctuations, the experimental conditions cannot be easily reproduced. To study the bursting CSR in multi-bunch operation an ultra-fast and high-accuracy data acquisition system for recording of individual ultra-short pulses has been developed. The Karlsruhe Pulse Taking Ultra-fast Readout Electronics (KAPTURE) is able to monitor all buckets turn-by-turn in streaming mode.

KAPTURE provides real-time sampling of the pulse with a minimum sampling time of 3 ps and a total time jitter of less than 1.7 ps. In this paper we present the KAPTURE system, the performance achieved and the integration in the ANKA control system.

INTRODUCTION

At the ANKA synchrotron light source, up to 184 electron buckets can be filled with a distance between two adjacent bunches of 2 ns corresponding to the 500 MHz frequency of the accelerating RF system.

Since a few years, special user operation with reduced bunch length in the order of a few picoseconds has been available to research communities. In this mode, coherent synchrotron radiation is generated for electro-magnetic waves with a wavelength in the order of or longer than the electron bunch length. Due to this, we observe a strong amplification of the radiation spectrum in the THz band. Moreover, above a certain current threshold, a coherent modulation of the longitudinal particle distribution (microbunching) occurs due to CSR impedance [1]. This particle dynamic effect changes the characteristics of the CSR tremendously. The microbunching structures fulfil a coherence condition for shorter wavelengths. This leads to an instantaneous increase of the radiated THz power. Observations in the time domain show bursts of radiation that occur with different periodicities depending on the bunch current. The characteristics of the bursting patterns are unique for different sets of accelerator parameters [2].

The KAPTURE (KArlsruhe Pulse Taking and Ultrafast Readout Electronics) system opens up the possibility to monitor the THz radiation of all bunches in the ring over a principally unlimited number of turns, realising a new type of measurement at ANKA. In this paper we present

the KAPTURE system and the integration in the ANKA environment.

KAPTURE SYSTEM

The KAPTURE system records individual pulses continuously with a sub-millivolt resolution and a timing resolution in the order of picoseconds. KAPTURE is a flexible system and can be easily configured for the requirements of any synchrotron facility.

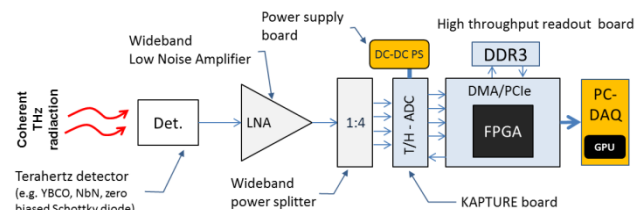


Figure 1: KAPTURE system for the detection of coherent THz radiation generated at ANKA.

The architecture of the KAPTURE system is shown in Fig. 1. It consists of a Low Noise Amplifier (LNA), a power splitter, a picosecond pulse sampling stage called “KAPTURE board”, a high throughput readout board and a high-end Graphics Processing Unit (GPU). The signal from the detector is fed into a LNA and then divided in four identical pulses by a wideband power splitter. Matching of the high bandwidth required to design a novel wideband power divider architecture [3]. The LNA gain compensates the insertion loss due to the power divider with a minimal additional noise. The KAPTURE board consists of four parallel sampling channels each operating at 500 MS/s [4]. Each sampling channel receives one of the four pulses from the power splitter and acquires it with one sampled point. The sampling time between the channels is settable with an accuracy of 3 picoseconds. The final result is that each detector pulse will be sampled with 4 sample points at a programmable sampling time between 3 and 100 ps. The basic concept and the architecture of the picosecond KAPTURE board have been reported previously [5]. The high throughput readout board uses a new bus master DMA architecture connected to PCI Express logic [6] to transfer the digital samples from the KAPTURE board to a high-end GPU server. For continuous data acquisition a bandwidth of 24 Gb/s (12 bits @ 2 ns * 4 digital samples) is necessary. The DMA architecture has been developed to meet this requirement with a high data throughput of up to 32 Gb/s. The GPU computing node is used for real-time reconstruction of the pulse from the 4 digital samples. Afterwards, the peak amplitude of each pulse and the time

between two consecutive pulses are calculated with picosecond time resolution. The GPU node performs also an on-line Fast Fourier Transform (FFT) for a frequency analysis of the CSR fluctuations.

The internal organization of the KAPTURE system and its components are shown in Fig. 2. The detector signal is connected to the LNA by a wideband V-connector and then propagated to the power divider by a tee-bias device.

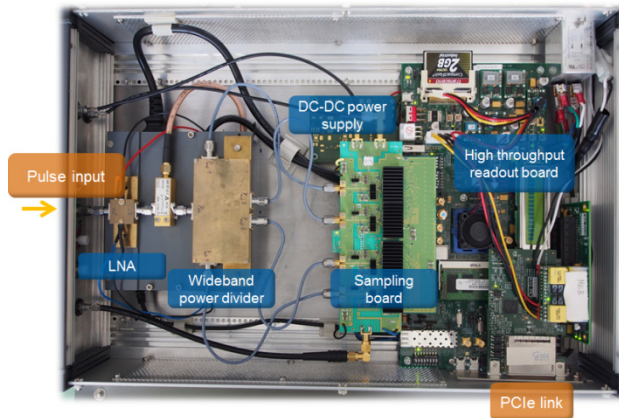


Figure 2: View inside the KAPTURE system.

The high data throughput readout board is equipped with a Virtex 6 FPGA that receives the digitized samples, tags them with the current bunch number, and sends the data to the GPU server via the PCIe data link.

PERFORMANCE

The KAPTURE system is designed for continuous sampling of very short pulses (minimum FWHM of a few tens of picoseconds) with typical amplitude in the order of some tens of millivolts. Due to ultra-fast and low amplitude pulses, KAPTURE has been designed with special precautions regarding component selection and layout technologies. The RF technologies and the circuits have been reported previously [3]. The RF/microwave analog front-end is operating at an analog bandwidth of DC - 50 GHz. For a better control of the characteristic impedance of the transmission lines, especially at high frequencies, special wideband coplanar waveguide transmission lines have been designed using a via fence technique. To achieve this performance special substrates have been used, Duroid 5880 for the analog front-end and Roger 4003 for digitalization circuits respectively [3,4].

The picosecond time delay chip, used to distribute the sampling signals at the four channels can be programmed by FPGA with a picosecond resolution.

KAPTURE is flexible and can operate in both real-time and equivalent sampling modes. In the equivalent time sampling mode KAPTURE is able to acquire a periodical waveform with a sampling rate that exceeds 300 GS/s and a total observation time of up to 2.2 ns. The low noise layout guarantees a Gaussian time jitter distribution with a measured standard deviation (Std-Dev) of 1.7 ps [5].

The time characterization of the KAPTURE system is a crucial part in the evaluation of the performance. For this

purpose, very short pulses with a FWHM of few tens of picosecond and a time jitter of few picoseconds are required. Due to limitations of commercial pulse generators, we have performed the characterization using short pulses generated by an YBCO detector [7] illuminated with coherent synchrotron radiation. The input pulse was previously measured with a real-time oscilloscope and shows a FWHM of 42 ps as well as an amplitude of 45 mV. In addition, we configured the KAPTURE system to operate in equivalent sampling mode with a sampling time of 3 ps, thus achieving a sampling rate of more than 300 GS/s. The pulse shape acquired with the KAPTURE system is in agreement with the measurements done with the real-time oscilloscope; all results are reported in the previous work [3]. KAPTURE is designed to work in real-time sampling mode where each pulse is acquired by four sampling points. Fig. 3 shows a dataset acquired in real time acquisition mode by a fast cryogenic YBCO detector. For each pulse the four sampling points S1, S2, S3 and S4 are acquired with a time distance of 15 ps respectively between the first sample S1 and the second S2, 9 ps between S2 and S3 and 15 ps between S3 and S4, while the time distance between two consecutive pulses is 2 ns.

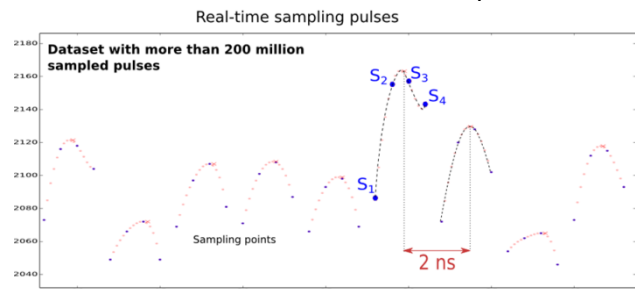


Figure 3: Details of reconstructed pulses, the x-axis shows the sampling points and the y-axis the ADC counts.

The real-time data analysis is based on a GPU co-processor. The GPU is used to reconstruct each pulse and measure its amplitude and time of arrival. It also performs an on-line Fast Fourier Transform (FFT) for a frequency analysis of the CSR fluctuations.

To sustain a continuous data acquisition a bandwidth of 32 Gb/s is necessary. A high-throughput readout system is used to transfer the sampled data from the digitizer stage to the high-end GPU-DAQ system. The readout architecture and the FPGA firmware are presented in [3].

The KAPTURE system, combined with different terahertz detectors (e.g. YBCO, NbN, Zero Biased Schottky Diode), has been proven to be an indispensable tool for analysis of CSR fluctuations in single- and multi-bunch mode. This method with a very high potential opens up new diagnostics possibilities such as the instantaneous measurement of the bursting threshold and chromaticity effects on longitudinal particle dynamics. Several important physics results have been published in the field of synchrotron terahertz radiation like the CSR behaviour at different current ring condition, the bunch-to-bunch interactions, the dependency of bursting threshold and momentum compaction [8, 9].

INTEGRATION OF THE KAPTURE SYSTEM AT ANKA

The KAPTURE system has been designed to acquire datasets for a long observation time. The normal current decay in the synchrotron ring offers the possibility to study the CSR emission at different storage ring currents. For long-term observations taking several hours we implemented a data reduction method in KAPTURE. Only every 10th turn is saved, thus limiting the instability dynamics frequency to 270 kHz. Moreover, only one second out of 10 seconds is stored, resulting in 1 Hz frequency resolution. With these settings we are able to observe simultaneously two different effects: the dynamics of the high frequency changes in the kHz-range due to electron bunch phase space rotations and the slowly changing beam-current depending on the bursting behaviour.

To keep the information between each individual dataset and the beam condition consistent, the KAPTURE acquisition system has been integrated in a hierarchical software architecture shown in Fig.4.

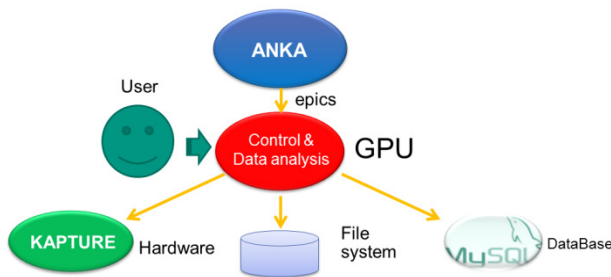


Figure 4: Integration of the KAPTURE system at ANKA.

This architecture integrates the control system, the DAQ and the on-line data quality check sub-systems in a single environment. The “Control and Data analysis unit” is the main software component. It receives the ANKA beam configurations/conditions by means of EPICS [10] and configures the KAPTURE hardware for the desired data readout (Control mode). This unit is running at the GPU server, it receives the readout data from KAPTURE with a data rate of 32 Gb/s and performs the real-time data analysis (Data analysis mode). A data quality check is included in the units in order to trigger specific actions in case of wrong data format. Optionally, an on-line FFT can be used also for a fast verification of the quality of the data acquired. The “Control and Data analysis unit” provides a high level and lightweight Graphics User Interface (GUI) that offers access to most of the system configuration in a controlled fashion and also offers basic functionality to view and analyse the data.

A MySQL database has been integrated to tag each individual dataset with the current accelerator parameters values. In this way, each measurement is automatically correlated with the exact beam condition that has generated the CSR measured by KAPTURE. This is

useful for an easy and fast identification of the interesting files for the off-line data analysis.

CONCLUSION

In this paper we have presented the KAPTURE system, designed for sampling bunch-by-bunch CSR pulses with fast THz detectors over long observation times. The system is able to measure amplitude and timing of individual pulses with a rate of 500 MHz. A low noise design and a wide dynamic range resulted in a timing accuracy of less than 3 psec. We have also presented the integration of the KAPTURE system in the ANKA synchrotron machine environment. The KAPTURE system allows scientists to study a variety of physical phenomena at the ANKA storage ring.

REFERENCES

- [1] G. Stupakov and S. Heifets, “Beam instability and microbunching due to coherent synchrotron radiation”, Phys. Rev. STAB, 2002, Vol. 5, Nr. 5, DOI:10.1103/PhysRevSTAB.5.054402.
- [2] M. Schwarz et al., “Bursting Patterns of Coherent Synchrotron Radiation in the ANKA Storage Ring”, WEPEA011, Proc. IPAC2013, Shanghai, China, <http://jacow.org/>.
- [3] M. Caselle et al., “An Ultra-fast Digitizer with Picosecond Sampling Time for Coherent Synchrotron Radiation”, 18-FEFD-4 (1119), 19th Real-Time conference IEEE-RT, Nara, Japan (2014).
- [4] M. Caselle et al., “Ultra-fast data acquisition system for coherent synchrotron radiation based on superconducting terahertz detectors”, WEOBB202, Proc. IPAC2013, Shanghai, China, <http://jacow.org/>.
- [5] M. Caselle et al., “An ultra-fast data acquisition system for Coherent Synchrotron Radiation with terahertz detectors”, JINST 9 C01024 doi:10.1088/1748-0221/9/01/C01024.
- [6] L. Rota, M. Caselle et al., “A New DMA PCIe Architecture for Gigabyte Data Transmission”, PS4-9, Proc. 19th Real-Time conference IEEE-RT, Nara, Japan (2014).
- [7] P. Thoma et al., “High-speed Y-Ba-Cu-O Direct Detection System for Monitoring Picosecond THz Pulses”. IEEE Transactions on terahertz science and technology, vol. 3, NO.1 (2013).
- [8] M. Caselle et al., “Commissioning of an Ultra-fast Data Acquisition System for Coherent Synchrotron Radiation Detection”, THPME113, Proc. IPAC2014, Dresden, Germany, <http://jacow.org/>.
- [9] M. Caselle et al., “A picosecond sampling electronic “KAPTURE” for terahertz synchrotron radiation”, MOCZB1, Proc. IBIC2014, Monterey, California, <http://jacow.org/>.
- [10] EPICS (Experimental Physics and Industrial Control System), <http://www.aps.anl.gov/epics/index.php>

INTEGRATION OF INDEPENDENT RADIATION MONITORING SYSTEM WITH MAIN ACCELERATOR CONTROL

N. Kamikubota[#], N. Yamamoto, J-PARC, KEK & JAEA, Tokai-mura, Ibaraki, Japan
T. Iitsuka, S. Yoshida, Kanto Information Service, Tsuchiura, Ibaraki, Japan

Abstract

The radiation monitoring system of J-PARC was constructed as a part of safety facilities. It is isolated, and has been operated independently from the main accelerator control.

In 2013, integration of the radiation monitoring system with the main accelerator control was discussed. In order not to affect to the original safety system, standard TCP/IP network connections and accesses to the central database are not allowed. During 2013-2014, we added new hardware to the existing systems, and developed device-level data-link layers to enable one-way data transfer to the accelerator control system.

In 2014, radiation monitoring data can be supervised from the accelerator control system. We understand that this is a significant improvement to realize safer operation of J-PARC accelerators and experimental facilities.

INTRODUCTION

An accelerator facility is always associated by related facilities. For example, safety facilities (a personal protection system, a radiation safety system, etc.), utilities (water cooling system, electricity distribution facility, etc.), and so on. In J-PARC accelerator complex, the radiation safety system was constructed by non-accelerator group, and had been operated independently. It has an isolated network, dedicated terminals for data view and history retrieve. Behind the fact, there exists a strong policy: "safety systems must be independent".

In J-PARC, the radiation monitoring system is a part of the safety system. When accelerator operators and/or staff members wanted to know radiation levels of monitoring posts, they had to visit the radiation safety office. Checking of radiation monitoring data from the central control room was not possible.

SCHEME FOR DATA-SHARING

J-PARC Accelerator Complex

J-PARC (Japan Proton Accelerator Research Complex) is a high-intensity proton accelerator complex. It consists of three accelerators: a) 400-MeV linac (LI), b) 3-GeV Rapid Cycling Synchrotron (RCS), and 30-GeV Main Ring (MR). Addition to them, there are three experimental facilities: d) Material and Life Science Experimental Facility (MLF), e) Neutrino Experimental Facility (NU), and f) Hadron Experimental Facility (HD). J-PARC was constructed and has been operated jointly between two institutes: JAEA and KEK [1,2].

[#]norihiko.kamikubota@kek.jp

ISBN 978-3-95450-146-5

Accident of Hadron Experimental Facility

On May 23, 2013, we had a serious accident at the Hadron Experimental Facility. Radioactive materials were released out of the radiation controlled area. The dose level was estimated less than 0.17 uSv on the site boundary closest to the Hadron Experimental Facility. Moreover, 102 workers were exposed by uncontrolled radioactive materials. Out of them, 34 persons were found to receive radiation dose in the range of 0.1-1.7mSv, all below the legal limit [3,4].

A team of third-party experts (the External Expert Panel) was established in order to review and inspect the accident. In August, 2013, the official report, including preventive measures against recurrence of similar accidents, was submitted to the Nuclear Regulation Authority of Japan [5]. It reflects the recommendations of the Panel. In the report, lack of sharing of radiation monitoring information was pointed out.

Policies and Plans for Implementations

We discussed a method to enable sharing of radiation monitoring data between the radiation monitoring system and the accelerator control system. In order to keep the independency of the safety system, TCP/IP network connections are not allowed. In addition, accesses to the central database of the safety system are prohibited, to avoid increases of system loads. We accepted above policies as same as before.

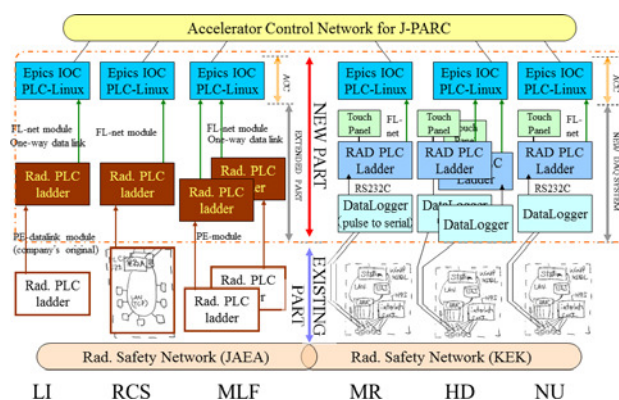


Figure 1: Overview of the data-sharing scheme.

We investigated the existing radiation monitoring systems. Our idea is that data-links to the device-level layers are possible without influences on the original safety system. The overview of the implementation scheme is shown in Figure 1.

Key issues are:

- For JAEA (LI, RCS and MLF), an extension of the existing PLC-based front-end system is considered. A dedicated data-link (PE data-link) is used to export monitoring data to the extended part.
- For KEK (MR, NU and HD), the existing system is a CAMAC-based DAQ system. Introducing new DAQ system is considered. Raw pulse signals to CAMAC are divided and are fed into the new DAQ system.
- Between the accelerator control and above two new systems, another data-link layer, FL-net, is used for one-way data transfer. Here FL-net is a device-level communication network defined by a Japanese consortium [6]. This layer is significant to guarantee independency of the safety system as before.
- A PLC-type EPICS IOC, which is a standard of J-PARC MR control [7,8], is used to receive shared-data of radiation monitors. In total, there are six EPICS IOCs, each corresponds to one of accelerators and experimental facilities.

IMPLEMENTATIONS

Figure 2 shows implementation details of both JAEA and KEK cases. For JAEA, the extended part was developed by the company which is in charge of the existing system. For KEK, new DAQ system was developed under a collaboration of us (accelerator staff) and radiation safety staff. It consists of: (a) “Data Logger”, a commercial product which converts pulse signals to periodical serial string outputs, (b) “RAD PLC Ladder”, accepts serial inputs, calculate radiation monitoring levels, and generate alarm I/O signals (HIHG and HIHI levels), and (c) “Touch panel”, enables to configure the DAQ parameters and to view monitoring data on-site.

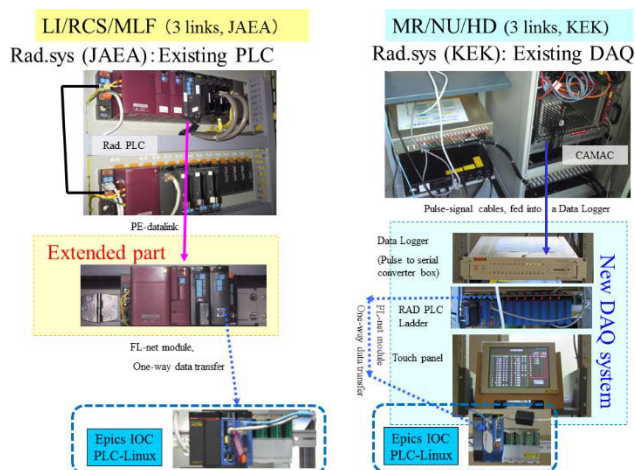


Figure 2: Implementations at JAEA and KEK.

In fact, operation of J-PARC had been suspended for seven month due to the accident. Beam operation restarted in December, 2013 [9]. During the month followed by the suspended term, we proceeded the

implementation one by one, as show in Figure 3. In October, 2014, alarm threshold values for MR and NU are in a test phase. The implementation for HD is just on-going phase, and to be completed in November, 2014.

	LI	RCS	MLF	MR	NU	IID
Implementation	Dec. 13	Jan.14	Feb.14	Jun.14	May.14	On going
Monitoring data	○ OK	○ OK	○ OK	○	○	Not yet
Alarm data	○ OK	○ OK	○ OK	△ (In test)	△ (In test)	Not yet

Figure 3: Rough history of implementations.

OPERATION EXPERIENCES

Demonstration of Radiation Monitoring by Accelerator Control

Figure 4 demonstrates 2-shift (~16 hours) trends of gas monitors on June 20, 2014. A standard EPICS tool, StripTool, is used. It shows behaviour of radiative gas levels correspond to the beam operation. In addition, data of two different sites, MR and NU, are merged in a single graph seamlessly.

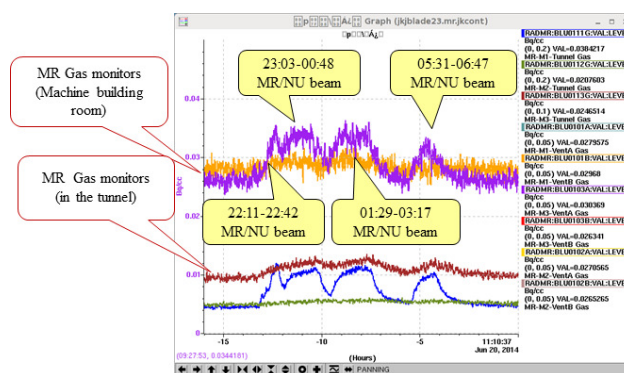


Figure 4: Demonstration of monitoring radiation data using an accelerator control tool.

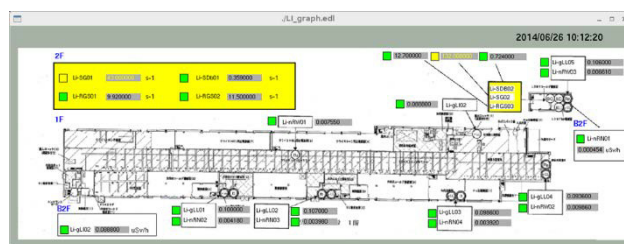


Figure 5: Map-style alarm-level indications for J-PARC LI.

Figure 5 is a map-style screen for the linac accelerator. Alarm levels, HIGH in yellow and HIHI in red, would be shown with color indications. Figure 5 is the screenshot on June 26, 2014, a few hours after accelerators stop, accordingly radiative gasses were still in the machine ventilation room at the “2F” floor level.

In fact, this screen was requested by accelerator operators. They need to identify the alarm location(s) as soon

tool, edm, is essential to develop this screen in a very short term (roughly a week).

Available Terminals

Before the accident, only a few dedicated terminals in the radiation safety office were available to check radiation monitoring information. Now all the terminals of the accelerator control can be used. Total number could be about 50 in the central control room. Additional pieces of terminals are also available in accelerator's local rooms and in experimental facilities.

CONCLUSION

We integrated the radiation monitoring system, which was constructed and has been operated independently, into the main accelerator control. Operators and commissioning staff members can check radiation monitoring data much easier than before. This work improve safety of J-PARC accelerators against possible damages using high intensity proton beams.

We thank the radiation safety staff members and accelerator operators of J-PARC, for their positive collaborations and discussions. Their contributions are indispensable to the present work.

REFERENCES

- [1] <http://j-parc.jp/index-e.html>
- [2] S. Nagamiya, "Introduction to J-PARC", Prog. Theor. Exp. Phys. (2012), 02B001.
- [3] "Accident of J-PARC Hadron Experimental Facility"; <http://j-parc.jp/HDAccident/HDAccidente.html>
- [4] "J-PARC News – June 2013 (Issue #98)", http://j-parc.jp/en/news/2013/J-PARC_News-e1306.html
- [5] "Outline of the 3rd report on the radioactive material leak at the Hadron Experimental Facility of the Japan Proton Accelerator Research Complex (J-PARC)", Aug. 2013; http://j-parc.jp/en/topics/HDAccident20130812_02.pdf
- [6] <http://www.jema-net.or.jp/English/businessfields/standarization/open/summary/>
- [7] N. Kamikubota et al., "J-PARC Control toward Future Reliable Operation", ICALEPCS2011, Grenoble, France, Oct. 2011, MOPMS026, pp. 378-381; www.JACoW.org.
- [8] J.-I. Odagiri et al., "Application of EPICS on F3RP61 to Accelerator Control", ICALEPCS2009, Kobe, Japan, Oct. 2009, THD005, pp. 916-918, www.JACoW.org.
- [9] T. Koseki and K. Hasegawa, "Present Status of J-PARC – after the Shutdown due to the Radioactive Material Leak Accident", IPAC'14, Dresden, May 2014, Germany, THPME061, pp. 3373-3375; www.JACoW.org.

LabVIEW PCAS INTERFACE FOR NI CompactRIO

G. Liu*, J. Wang, K. Xuan, C. Li, NSRL, USTC, Hefei, Anhui 230029, China
K. Zheng, Y. Yang, National Instruments China, Shanghai 201203, China

Abstract

When the NI LabVIEW EPICS Server I/O Server is used to integrate NI CompactRIO devices running under VxWorks into EPICS, we notice that it only supports "VAL" field, neither alarms nor time stamps are supported. In order to overcome these drawbacks, a new LabVIEW Channel Access Portable Server (PCAS) Interface is developed, and is applied to the Hefei Light Source (HLS) cooling water monitor system. The test results in the HLS cooling water monitor system indicate that this approach can greatly improve the performance of the NI CompactRIO devices in EPICS environment.

INTRODUCTION

The Hefei Light Source (HLS) is a dedicated second generation VUV synchrotron radiation light source. It was designed and constructed two decades ago. In order to improve the performance of the HLS, especially to obtain higher brightness and more straight sections, an upgrade project was started from the end of 2009. The cooling water monitor system was reconstructed in the project. The CompactRIO (Compact Reconfigurable Input Output) product from National Instruments (NI) is adopted to the cooling water monitor system because it is designed for harsh environment, and the software is easy to duplicate and deployed.

The NI CompactRIO devices are running under VxWorks. When the NI LabVIEW EPICS Server I/O Server [1] is used to integrate the NI CompactRIO devices into EPICS, we notice that it only supports "VAL" field, neither alarms nor time stamps are supported. In order to overcome these drawbacks, a new LabVIEW Channel Access Portable Server (PCAS) [2] Interface is developed.

The LabVIEW PCAS Interface is a software library which provides channel access server ability to LabVIEW code and emulates the way how an IOC manages process variables. It is based on the PCAS which implements the underlying channel access functions. There are two ambitious aims in designing the LabVIEW PCAS interface: (1) it supports all platforms where LabVIEW exists, including Windows, Linux and VxWorks; (2) it supply common records with common fields, similar usage pattern as IOC.

DEVELOPMENT

The LabVIEW PCAS Interface employs the Channel Access Portable Server distributed with EPICS base. The portable server comprises of server library and server tool.

The server library refers to the software that lies beneath the C++ class-interface to the portable server. The developer only needs to know how to use the interface in order to create a server tool. A server tool is a specific application written by a developer using this interface.

There is no record in the portable server, all PVs are standalone data. In order to provide users an IOC-like interface, we encapsulate PVs in the format of <record name>.<field name>, e.g. Temp.VAL.

Figure 1 is the flowchart of a server application, including the VI interfaces and the background thread. The VI interfaces are functions we provide to users. The background thread implements all the supported CA server interfaces called by CA clients. The records store all data used by the server application. In the background thread, the Channel Access is handled by the portable server. We implement the callbacks required by underlying components.

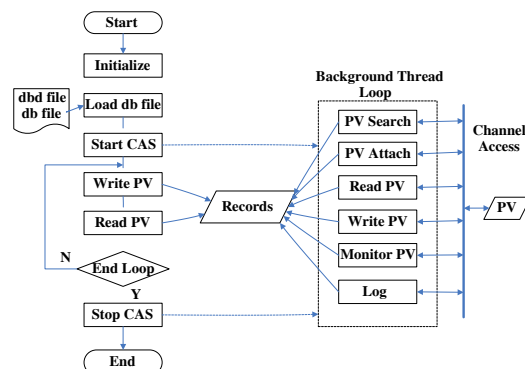


Figure 1: Flowchart of the LabVIEW PCAS Interface.

There are many record types defined by EPICS base, but we only support some common records with common fields up to now in order to limit the work load. Table 1 shows the supported record types and field. Here, "all" refers to the record type ai/ao, bi/bo, stringin/stringout and waveform.

Table 1: Record Types and Fields Supported by the LabVIEW PCAS Interface

Record Type	Fields
all	VAL/STAT/SEVR
ai/ao	HIHI/LOLO/HIGH/LOW HHSV/LLSV/HSV/LSV HYST/ADEL/MDEL
bi/bo	ZSV/OSV/COSV
waveform	NELM
ai/ao/waveform	HOPR/LOPR/PREC/EGU

*gfliu@ustc.edu.cn

USAGE

As mentioned earlier, the LabVIEW PCAS Interface is a software library. This library consists of several LabVIEW VIs and shared object libraries. The LabVIEW VIs are platform independent and the shared object libraries are compiled for each supported platform. They can be called by VIs through the "Call Library Node".

Figure 2 shows how the LabVIEW PCAS Interface is used. The server application first creates a CA server with a given server name. A .dbd file which includes all the

database definitions, and a .db file to configure records and PVs that it handles are loaded into the server. The "Run CAS" VI is used to start the server application. The CA client can access the server application after it is started. During the running time, the "Write PV" and "Read PV" VIs are used to write the PV value or get the PV value respectively. If a PV is changed by the server application or by a client with 'caput', at least one monitor is set on that PV, and an event is posted by the server application. The "Stop CAS" VI is used to stop the server application.

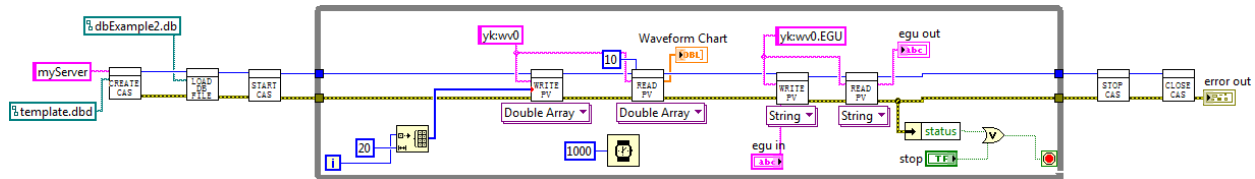


Figure 2: Usage of the LabVIEW PCAS Interface through LabVIEW code.

APPLICATION

The signals to be monitored in the HLS cooling water monitor system are the cooling water temperature, magnet temperature, cooling water pressure and cooling water flow. There are totally about 1000 signals, and these signals are distributed over the whole light source area. The CompactRIO products from National Instruments are adopted. The hardware structure of the cooling water monitor system is shown in Figure 3.

The CompactRIO is a real-time embedded industrial controller made by National Instruments for industrial control systems. It is a combination of a real-time controller, reconfigurable IO Modules, a FPGA module and an Ethernet expansion chassis. The National Instruments CompactRIO devices used for the cooling water monitor system are NI cRIO 9073, NI 9217 and NI 9208.

The NI cRIO 9073 is an 8-slot chassis with a 266MHz real-time processor, a 2M gate reconfigurable FPGA and a 10/100BASE-TX Ethernet port. The embedded processor runs the WindRiver VxWorks real-time operating system. The NI 9217 is a module which can detect 4 channels 3/4-wire RTD signals. The NI 9208 is a module which can detect 16 channels current signals with the range +/- 21.5mA.

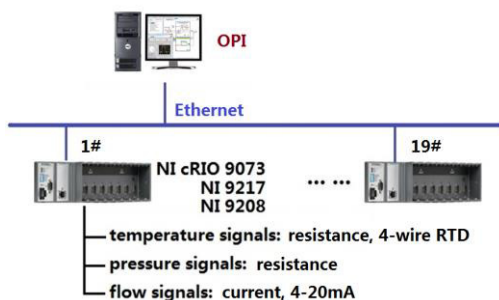


Figure 3: Hardware structure of the HLS Cooling Water Monitor System.

A LabVIEW program package is developed, and installed in each NI cRIO 9073 with a configuration file. In this way it is easy to operate, maintain and deploy in a large-scale. Figure 4 shows the software list downloaded to each NI cRIO 9073. As shown in Figure 5, the configuration file includes the information of all chassis and the installed modules. The configuration file can be generated step by step with a LabVIEW VI for all online NI cRIO 9073 chassis, or it can be edited with any text editors. The configuration file will be checked automatically by the software when the NI cRIO 9073 restarts. The USER1 LED will flash when any conflict or mismatch occurs.

In order to monitor the status of the NI cRIO 9073, two time related PVs are added to monitor the current time and startup time respectively.

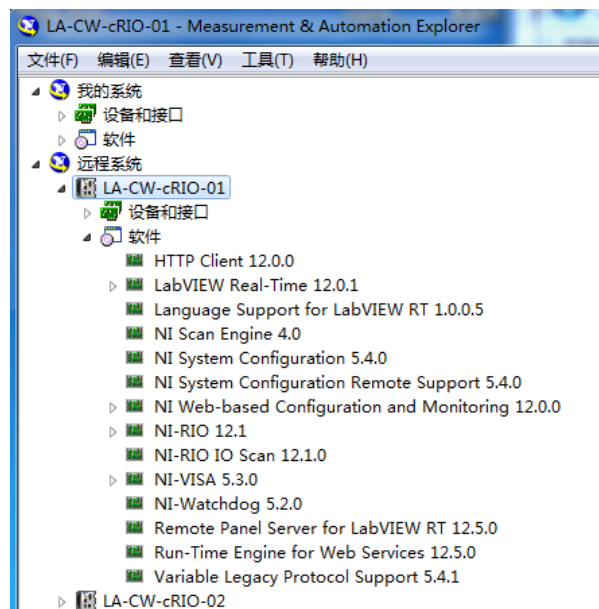


Figure 4: Software downloaded to the NI cRIO 9073.

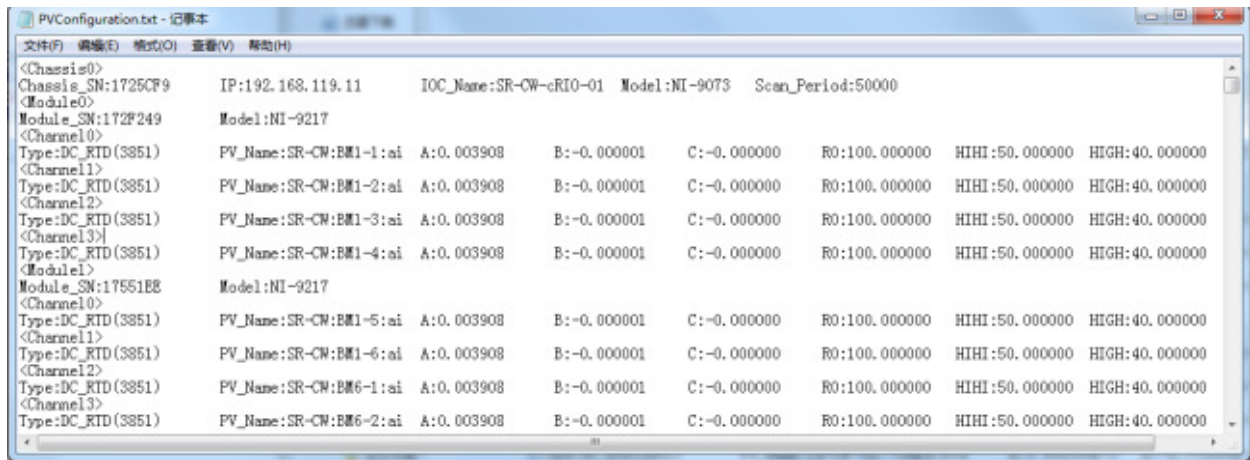


Figure 5: Configuration file.

We integrate the HLS Cooling Water Monitor System into EPICS with the LabVIEW PCAS Interface. Figure 6 shows the interactive interface with commands, alarms and time stamps. The system is stable for over one month.

```
ioc@gflliu_test1 ~ $ caget LA-CW-cRIO-01:Startup_TimeStamp
LA-CW-cRIO-01:Startup_TimeStamp 2014/09/04 19:57:03
ioc@gflliu_test1 ~ $ caget LA-CW-cRIO-01:Current_TimeStamp
LA-CW-cRIO-01:Current_TimeStamp 2014/10/08 09:11:43
ioc@gflliu_test1 ~ $ caget LA-CW:LQ1-1:ai.HIGH
LA-CW:LQ1-1:ai.HIGH 25
ioc@gflliu_test1 ~ $ caget LA-CW:LQ1-1:ai.HIHI
LA-CW:LQ1-1:ai.HIHI 30
ioc@gflliu_test1 ~ $ camonitor LA-CW:LQ1-1:ai
LA-CW:LQ1-1:ai 2014-10-08 09:12:03.960376 25.613 HIGH MINOR
LA-CW:LQ1-1:ai 2014-10-08 09:12:04.960380 25.5983 HIGH MINOR
```

Figure 6: Interactive interface with commands.

The common EPICS extensions tools, e.g. probe, StripTool, edm, Channel Archiver, RDB Channel Archiver, CSS BEAST, work well with this interface. However, ALH is not compatible with it. The warning message “aCaNewAlarmEvent failed” pops up when ALH starts. ALH has the same behaviour when it is tested with the EPICS excas. So the incompatibility should originate from the PCAS.

CONCLUSION

The test results in the HLS cooling water monitor system show that the operation of the LabVIEW PCAS Interface is stable and its behaviour is similar to that of a normal IOC. The LabVIEW PCAS Interface can be regarded as a good solution to integrate the NI CompactRIO devices into EPICS.

REFERENCES

- [1] NI Developer Zone, "Interactively Configuring EPICS I/O Servers", <http://www.ni.com/white-paper/14149/en>.
- [2] Philip Stanley, "Channel Access Portable Server: Reference Guide", http://www.aps.anl.gov/epics/extensions/cas/CAS_Reference.pdf.

HLS POWER SUPPLY CONTROL SYSTEM BASED ON VIRTUAL MACHINE

J. Wang*, G. Liu, K. Xuan, C. Li, NSRL, Hefei, China

Abstract

The Hefei Light Source (HLS) is a VUV synchrotron radiation light source. It is upgraded recently to improve its performance. The power supply control system is a part of the HLS upgrade project. Five soft IOC applications running on the virtual machine are used to control 190 power supplies via MOXA's serial-to-Ethernet device servers. The power supply control system has been under operation since November 2013, and the operation results show the power supply control system is reliable and can satisfy the demands of slow orbit feedback with the frequency of 1Hz.

INTRODUCTION

The Hefei Light Source (HLS) is a VUV synchrotron radiation light source. It is upgraded recently to improve its performance. As a part of the HLS upgrade project, all the power supplies are rebuilt, and the power supply control system is correspondingly reconstructed.

There are 190 power supplies totally. They are divided into about ten types, and used for dipole magnet, quadrupole magnet and sextupole magnet, etc. All these power supplied are designed with the unified control interface. Five soft IOC applications running on the

virtual machine are used to control these power supplies via MOXA's serial-to-Ethernet device servers.

The power supply control system has been under operation since November 2013, and the operation results show that the power supply control system is reliable. The communication time is less than 50 ms, it can satisfy the demand of the slow orbit feedback with the frequency of 1Hz.

HARDWARE

The power supply control system is developed under EPICS environment, its hardware structure is shown in Figure 1.

Five softIOCs are running on the virtual machines, which is built with VMware. They communicate the power supplied via MOXA's serial-to-Ethernet device servers Nport 6650-16. All the power supplies has the unified interface, i.e. serial port with plastic fibre connection, the baud rate is 115.2kbps. A photoelectric converter with 16 ports is specially designed for MOXA Nport 6650-16, and is used between the serial device servers and the power supplies.

All the IOC applications are put on a NFS server, each softIOC is used as NFS client to share the IOC applications.

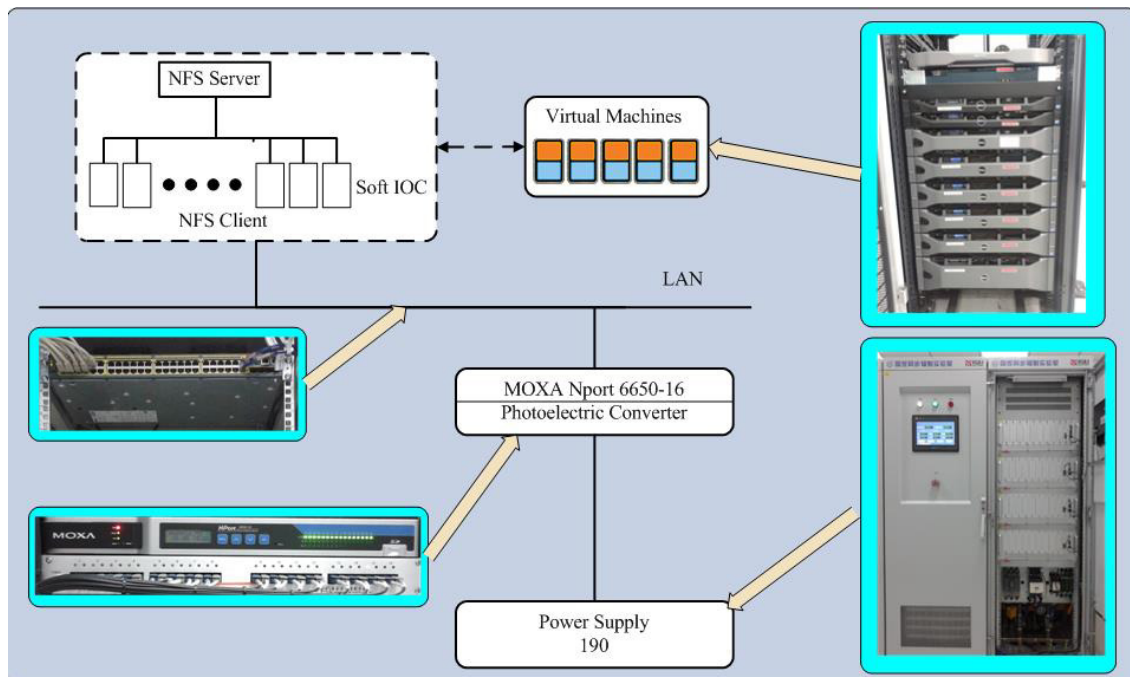


Figure 1: Hardware structure of power supply control system.

SOFTWARE

Figure 2 shows the software structure of the HLS power supply control system. The softIOC and OPI are all running under Linux environment.

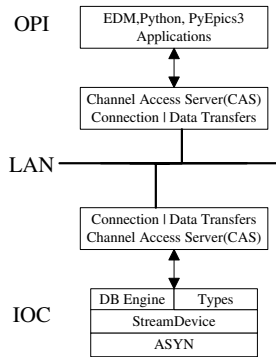


Figure 2: Software structure of the HLS power supply control system.

The module StreamDevice [1] and ASYN [2] are the interface between the softIOCs and the power supplies. The communication protocol is customized, it's an ASCII protocol with check sum. The module iocStats [3] is used to monitor the status of softIOC, and the module autosave [4] is used to support bumpless IOC reboot.

The human machine interface is developed with EDM [5], as shown in Figure 3. The setting is saved/restored with PyEpics3 [6].



Figure 3: OPI of the HLS ring correct magnet power supply control system.

PERFORMANCE

Among the HLS power supplies, the ring correct magnet power supplies are dynamic with 1Hz feedback frequency. So the respond time of the control system is the key parameter. There are 64 correct magnet power supplies totally, each half is for horizontal or vertical slow orbit feedback.

The respond time is tested with Wireshark, which is running on the softIOC. The softIOC sends consecutively the command of current setting to 32 correct magnet power supplies with 1Hz frequency. The test lasts for about 1080 seconds. Figure 4 is the application screen of Wireshark. The data obtained by Wireshark are analysed and the results are shown in Table 1. The time from the 1st sent package to the 32th received package is less than

50ms, it can satisfy the demand of the slow orbit feedback with the frequency of 1Hz.

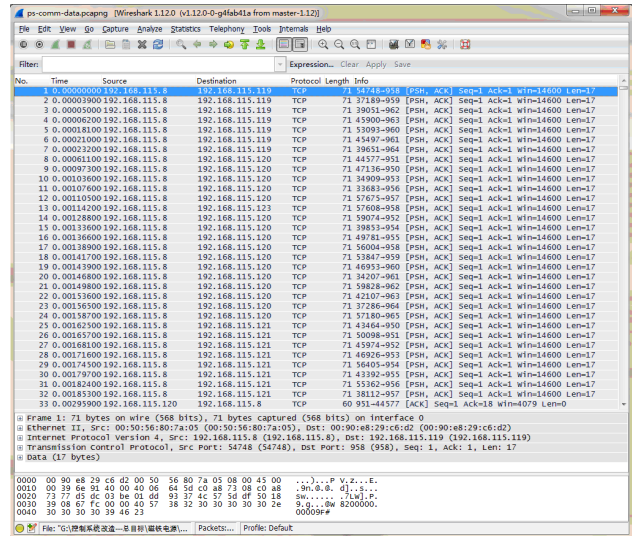


Figure 4: Application screen of Wireshark.

Table 1: Response Time Statistics

Time	Min.(ms)	Max.(ms)	Aver.(ms)	Std.(ms)
T1*	0.634	6.247	1.416	0.483
T2**	36.733	45.852	40.235	1.287

*T1: the time from the 1st sent package to the 32th sent package.

** T2: the time from the 1st sent package to the 32th received package.

The interface of the slow orbit feedback system is shown in Figure 5. The effect of the slow orbit feedback system is shown in Figure 6 and Figure 7.

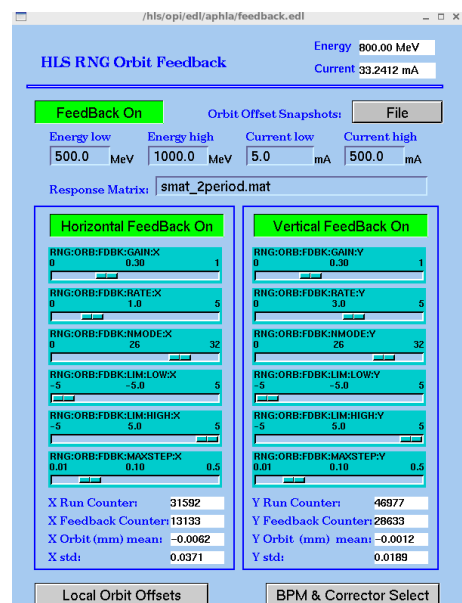


Figure 5: The interface of slow orbit feedback.

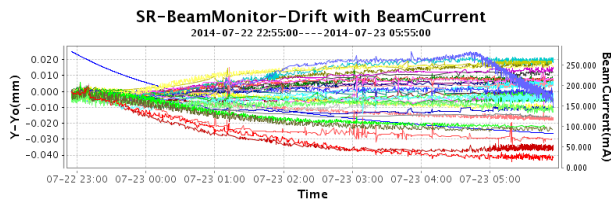


Figure 6: Vertical beam position drift when the slow orbit feedback system is off.

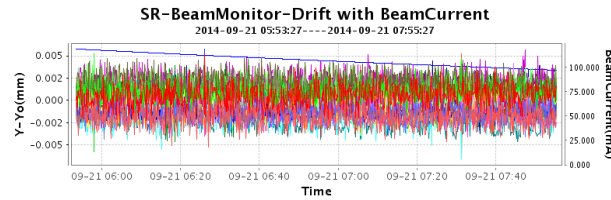


Figure 7: Vertical beam position drift when the slow orbit feedback system is on.

CONCLUSION

All the power supplies of HLS are controlled with the softIOCs on the virtual machine, and Ethernet is used instead of the fieldbus, the communication time is less than 50 ms, it can satisfy the demands of the slow orbit feedback with the frequency of 1Hz.

REFERENCES

- [1] <http://epics.web.psi.ch/software/streamdevice>
- [2] <http://www.aps.anl.gov/epics/modules/soft/asyn>
- [3] <http://www.slac.stanford.edu/comp/unix/package/epics/site/devlocStats>
- [4] <http://www.aps.anl.gov/bcda/synApps/autosave/autosave.html>
- [5] <http://ics-web.sns.ornl.gov/edm/>
- [6] <http://cars9.uchicago.edu/software/python/pyepics3>

THE SOFTWARE TOOLS AND CAPABILITIES OF DIAGNOSTIC SYSTEM FOR STABILITY OF MAGNET POWER SUPPLIES AT NOVOSIBIRSK FREE ELECTRON LASER

S. S. Serednyakov

Budker Institute of Nuclear Physics SB RAS, Novosibirsk, Russia

Abstract

The magnetic system of Novosibirsk free electron laser (FEL), which contains a lot of magnetic elements, is fed by power supplies of different types. The time stability of the output current of these power supplies directly influences the coherent radiation parameters and operation of the entire FEL facility. In this connection, we developed a system for diagnostics of the power supplies state, integrated into the common FEL control system. The main task of this system is to analyze the output current of the power supplies and calculate their time stability characteristics. Besides, this system is capable to determine the amplitude and frequency of output current ripples, if any, for a particular power supply and display obtained results. This paper describes the main architecture and some other capabilities of this system, and presents examples of its usage.

INTRODUCTION

A high-power FEL based on a multiterm energy recovery linac (ERL) [1] is under construction now at Budker Institute of Nuclear Physics. The first and second phases of the project have already been commissioned and are in operation.

The magnetic system, one of most important systems of the ERL, consists of numerous magnetic elements of different types. All these elements are fed by DC power supplies of different output powers and currents. The stability of operation of the FEL and its main parameters depends on the time stability characteristics of these power supplies. In this connection, there was developed a set of particularized software tools for monitoring of the output currents of all the power supplies. The architecture of the current system is influenced by the factors below.

1. The architecture of the software of the control system for the magnetic system and the installation as a whole.
2. The type and operation characteristics of the communication bus between the control PC and the power supply control devices.
3. The capabilities and main operation modes of these power supply control devices.

HARDWARE AND ARCHITECTURE OF CONTROL SYSTEM OF MAGNETIC ELEMENTS

The magnetic system and its control system of Novosibirsk FEL are described in detail in [2]. All power supplies are controlled by modules embedded into power supply racks and communicating with the central IBM-PC via a CAN bus interface. Power supplies of different types are governed by different controllers. For low-power current sources with an output current of up to 17 amperes, pairs of multi-channel controllers are used, controlling up to 16 power supplies – 16-channel DAC and 40-channel ADC devices. For high-power current sources, each power supply is commanded by a separate controller with 1-channel DACs and 5-channel ADCs. All controllers are connected to one CAN bus line [Fig. 1].

Table 1: Parameters of the Power Supply Controllers

Device	CANDAC16	CANADC40	CDAC20
Qty	24	24	13
PS Number	16	16	1
PS I _{max}	17	17	2500
ADC channels	---	40	5
DAC channels	16	---	1

Above is presented a comparative table of main controller parameters (Table 1.). As seen, all controllers containing an ADC module have multiple input channels. In the regular operation mode, these modules are working in the multi-channel mode, i.e. serial measurement of voltage in all input channels and sending of the values measured to the control PC via the CAN bus interface. In this case, the ADC works in the cyclic mode: once the measurement of the last channel is completed, a new measurement cycle begins from the first channel. The architecture of the set of commands for all these controllers and the CAN bus protocol allow operation of numerous controllers in such mode.

For diagnostics purposes, another measurement mode – “single-channel mode” – was realized. When switched to this mode, the controller starts serial measurements of a single selected channel, until it receives from the central PC a command “STOP” or a command “start new measurement cycle”. As mentioned above, switching

from one mode to another does not influence the operation of the rest modules and their data transfer to the central PC.

METHODS OF POWER SUPPLY DIAGNOSTICS

The main methods of diagnostics of the power supply output current are described below.

1. "Adjustment accuracy": comparison of the current value specified by the user with a real pre-set value.
2. "Long time stability": comparison of the current value measured after the user specified the latest new value with the value measured at the actual moment of time.
3. "Ripples diagnostics": Oscillographic measurements of the output current and processing of the waveform obtained for detection of possible current ripples.

The first two methods of diagnostics are used in the background mode during all the time of control software operation. In the main window of the application, a string with the names of power supplies whose current value difference exceeds the allowed warning or alarm value is marked yellow or red. These types of diagnostics are applied only to data obtained in the usual operation mode and do not require switching the ADC to another measurement modes. No need in any complicated mathematical processing of data.

The third method of diagnostics shall be switched on by the user and requires some manipulation with the modes of ADC operation and some mathematical processing of the data.

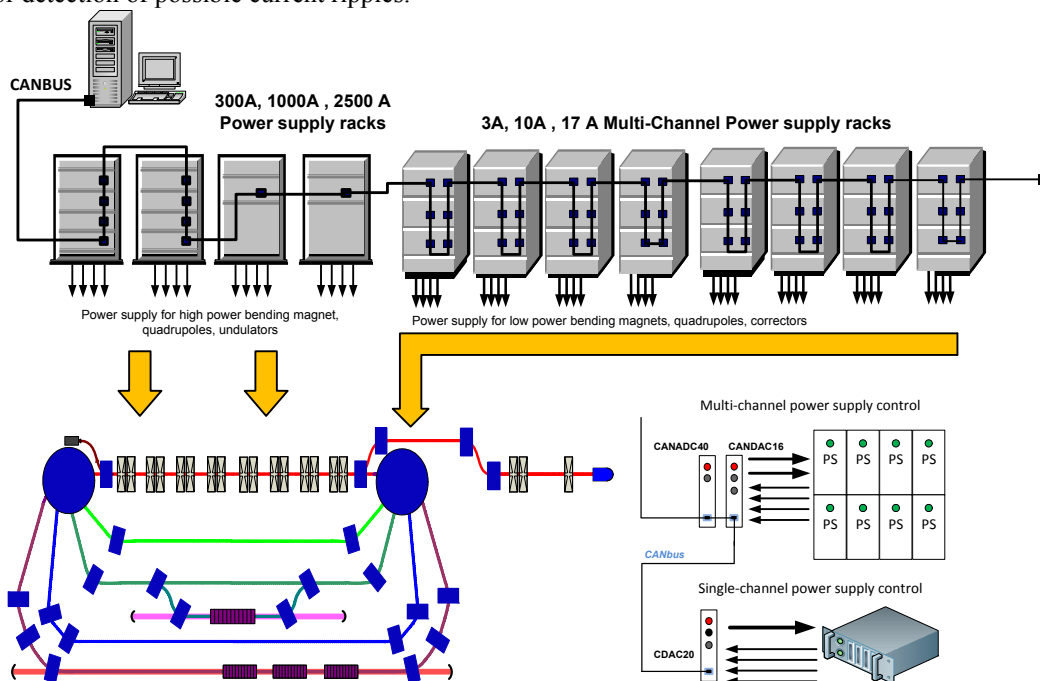


Figure 1: Main scheme of the system.

When this diagnostics procedure is started, the control application initiates the cyclic processing of the power supplies in the order of their placement in the power supply racks. Below is given the sequence of the cycle of the current ripples diagnostics method.

1. The ADC shall be switched to the single-channel mode. In so doing, the input channel is selected to which the current output of the power supply of interest is connected.
2. Collecting of data. Waiting for the ADC to accomplish a set number of measurements. The measured values are collected in one waveform. The number of measurements shall be specified by the user.
3. Processing of the obtained waveform of the measured current.

4. Stop of the current measurement mode of the ADC. If there are other power supplies, next to the just-scanned one, with current outputs connected to the running ADC channel, then the procedure is repeated from step 1 with next power supply and corresponding ADC input channel. Otherwise, the current ADC is switched to the usual (multi-channel) mode, and the application starts working with next power supply, i.e. it defines the required ADC and the input channel number and then starts from step 1.

Besides, it is possible to start this diagnostic procedure for a single power supply specified by the user.

As mentioned above, the user can specify the number of measurements, as well as the time for one measurement with the ADC, from 1 to 160 milliseconds. These two parameters allow one to specify the duration of the

measured oscillogram and its sampling frequency. Different values of these parameters could be useful for different types of output current diagnostics.

PROCESSING AND OUTPUT OF DATA MEASURED

When the measured current waveform for the power supply in question has been obtained, the following mathematical processing is executed:

1. The average current value for the entire waveform is calculated.
2. The mean square deviation and maximum deviation values for the entire waveform are calculated.
3. The spectrum of current ripples is calculated using the fast Fourier transform.

After execution of these operations, all the obtained values can be presented in the following graphs:

1. A dialog window with the graphs of the measured current waveform and its spectrum (Fig.2). This latter graph can be used for a direct view of the measured current waveform and its frequency characteristics.

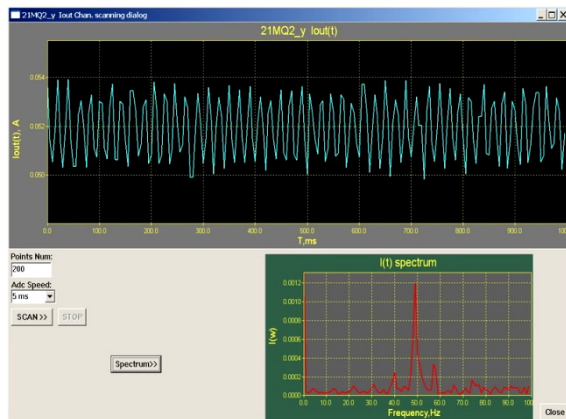


Figure 2: Graphs of the waveform of the measured current and its spectrum.

2. A dialog window with a total comparative graph of the calculated mean square deviation and maximum deviation values for all the power supplies (Fig.3). The comparative plot in this window is very useful for finding power supplies with the highest ripple values. Every power supply in the upper plot is presented with a red bar, showing the maximum deviation, and a blue bar, showing the mean square deviation. Besides, the user can launch repeatedly a scanning of each power supply and view the obtained waveform on the graph below.
3. A dialog window with a graph of all the calculated values (the mean square deviation and maximum deviation values) for a selected power supply for the time when these values were obtained (Fig.4). Since in this method the scanning of the power supplies is executed in the cyclic mode, the control application stores every calculated ripple value (after completion of the scanning) with its timestamp. As result, the user is able to view the

history of these values for each power supply during all the time of operation of the control application.

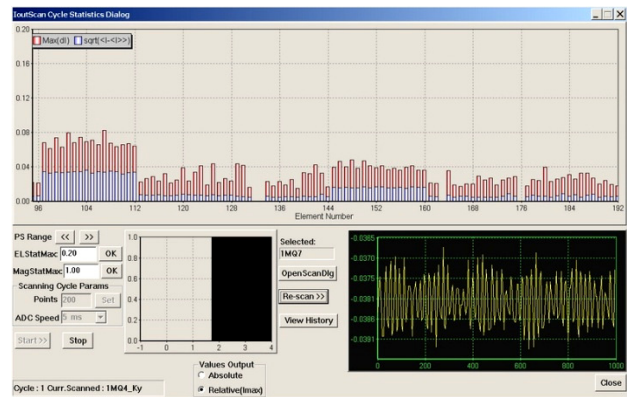


Figure 3: Comparative graph of calculated values for all power supplies.

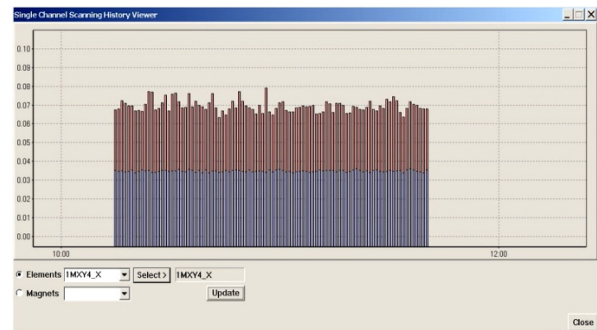


Figure 4: History of ripple values during all the operation time of the control application.

CONCLUSION

The above-described software modules give an advanced tool for diagnostics of the timewise behavior of the output current of any power supply. With this tool, the user can both investigate the output current of a single selected power supply, as well as its frequency characteristics, and start a scanning cycle for all power supplies to find those with maximum output current ripples. Storing and displaying of the calculated ripple-value history allow the user to control the quality of operation of a power supply during all the period of the facility operation.

REFERENCES

- [1] N.A. Vinokurov et al., "Novosibirsk free electron laser facility: Two-orbit ERL with two FELs", TUOD01, Proc. FEL2009, <http://jacow.org/>.
- [2] Yu. M. Velikanov et al., "Control system for magnet power supplies for Novosibirsk Free Electron Laser", MOPMU021, Proc. ICALEPCS2011, <http://jacow.org/>.

S. Rubio-Manrique, G. Cuní, D. Fernandez-Carreiras, Z. Reszela, A. Rubio,
CELLS-ALBA Synchrotron, Barcelona, Spain

The PyPLC Tango Device Server provides a developer-friendly dynamic interface to any Modbus-based control device. Raw data structures from PLC are obtained efficiently and converted into highly customized attributes using the python programming language. The device server allows to add or modify attributes dynamically using single-line python statements. The compact python dialect used is enhanced with Modbus commands and methods to prototype, simulate and implement complex behaviours. As a generic device, PyPLC has been versatile enough to interact with PLC systems used in ALBA Accelerators as well as to our Beamlines SCADA (Sardana). This article describes the mechanisms used to enable this versatility and how the dynamic attribute syntax allowed to speed up the transition from PLC to user interfaces.

ALBA[1], member of the Tango Collaboration[2][3], is a third generation Synchrotron in Barcelona, Europe. It provides light since 2012 to users through its 7 beamlines, with 2 more under construction.

Programmable Logic Controllers from several vendors (B&R, Pilz, ...) are used for acquisition, protection and motion within our Tango Control System[4]. PLC's are the main component of equipment and personnel protection systems, but they are also used in accelerators and beamlines for vacuum/temperature diagnostics and motion control. The most complex system managed by PLC's is the Equipment Protection System (EPS)[5].

The EPS is an autonomous system ensuring the safe operation of all elements in ALBA accelerators and Beamlines. It generates both interlocks and operation permits, following the logics previously defined between the Control section and the Accelerators and Experiments divisions and programmed by Control engineers.

EPS uses 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 movable elements: temperatures, vacuum sensors, position encoders and switches, electrovalves, ...

The Modbus protocol and Tango devices are also used to control PLC's in the RF circulators, bakeout controllers, water cooling system, air conditioning in the experimental hutches and overall Personnel Safety

Systems, on which the Tango Control System have just read access. The same control interface is used to communicate with all this subsystems, with certain customization depending on the control needs.

An interface between Tango Control System and our PLC-based subsystems was needed for three main purposes:

- Supervision of autonomous systems based on PLC's (EPS, PSS).
- Configuration of the EPS settings and interlock thresholds during commissioning.
- Integrate critical and diagnostics signals into our Control services like Archiving, Taurus UI, Alarms, Beamlines SCADA (Sardana) [6][7].

To achieve a successful integration of the PLC signals into our control system it was needed to automate the creation of Tango Attributes in the PLC device servers. The commissioning work-flow required a regular update of I/O and variables lists in the PLC's, and existing UI's and services like archiving had to be capable to keep pace with each update of the attribute list.

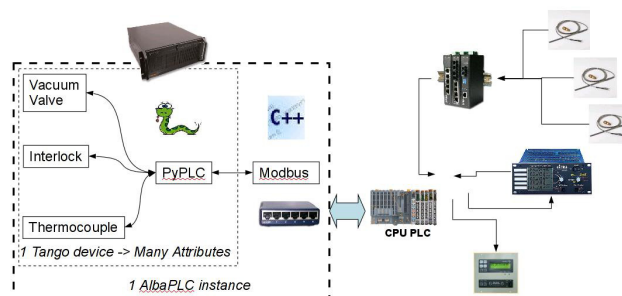


Figure 1: PyPLC Device Architecture.

The first device server developed at ALBA for communicating with PLC's was a C++ device server, ModbusPLC, running on top of the Modbus Device developed at the ESRF; that already implements all basic Modbus commands. The ModbusPLC C++ Tango device allowed to create and remove attributes depending on Tango property values [8], exporting as many integer attributes as 16 bits registers were mapped in Modbus.

This implementation presented several drawbacks:

- Hides the diversity of signals available from the PLC (digital inputs, flag registers, integers, 16 bits floats, 32 bit values spanning multiple registers).
- Too rigid for showing complex elements (needed many attributes to represent a 3-position valve).

- Triggered 1 Modbus command per attribute read, for a typical beamline PLC with hundreds of attributes it meant a full refresh every 30 seconds or so.

PyPLC, a Dynamic PyTango Device Server

PyTango, the Python binding of Tango, is the common framework of development at ALBA [9]. The PyPLC python device server overcomes the limitations of ModbusPLC, providing the high versatility of python [10] above the robustness of the reliable Modbus C++ server (Fig.1).

Advantages of PyTango device servers in python are:

- Python is a dynamically typed language, making devices prototyping and extension much faster.
- Objects and libraries are mutable, multiple inheritance and classes are modifiable in runtime.
- Device servers can be executed in any OS with no need of compiling or packaging.
- Tools like SWIG or Boost allow to use C++ APIs libraries from python.
- In fact PyTango is just a layer above Tango C++, which means that any feature/patch of mainstream Tango is also available in PyTango.
- Python allows to execute string formulas as python code, enabling complex Dynamic Attributes declaration on runtime.

Taking profit of these advantages, we developed a DynamicDS[11] template for Tango Devices that provided dynamic attribute type creation, customized calculations, configurable state composing and attribute-grouping. All these features being achieved through an easy syntax accessible to machine operators and scientists.

PyPLC inherits from DynamicDS template [12]. Creation of attributes is done writing new attribute formulas into the DynamicAttributes property of the Tango database, a process that can be done manually or automated by scripts.

Table 1: PyPLC Attribute Formulas, Int Array and a Writeable Boolean Flag.

TEMPERATURES=
DevVarLongArray(Regs(7800,100))
DIO_01= bool(
READ and Flag(80,7) or
WRITE and WriteFlag(81,7,int(VALUE)))

To enable the use of Modbus commands in the attribute formulas, the PyPLC exports all the commands needed for accessing Modbus variables (Read/Write Input/Holding Registers) and PLC variable type (Bit/Coil/Flag/Int/Long/Float/IeeeFloat/Double). Those methods can be used inside attributes (Table 1) or commands (Table 2) declaration to access any type of variable mapped in Modbus addresses.

Table 2: PyPLC Command Formulas

Open_PNV01=(WriteBit(193,2,1),1)[-1]
Close_PNV01=(WriteBit(193,1,1),0)[-1]

Optimizing Modbus Communications

The Modbus devices used at ALBA (mostly B&R plc's) show several limitations in the implementation of their Modbus communications. The maximum amount of registers to be acquired in a single modbus exchange is 120. And those communications take between 120 ms and 160 ms independently of the number of addresses read on each command.

To avoid these problems, PyPLC enhances the Modbus Tango Device providing smart mapping of the PLC memory. It allows to optimize ethernet/ serial communications and setup selective address refresh when needed. The memory areas are read and allocated in the device server as arrays, each of them dedicated to a certain type of data (boolean, int, float, double). Attributes of the PyPLC device will not access the Modbus communications but this arrays mapped in memory instead.

Actually the refresh of a typical ALBA PLC is about 3 seconds for a PLC with 2000 registers mapped to 300 attributes. Although faster refresh of certain areas of the memory can be setup up to 300 ms.

EXTENDING PYPLC

Exporting PLC Variables to PyPLC

When programming the PLC's, the controls engineer load variables information retrieved from our cabling and controls MySQL database [13]. Visual basic macros are used to update the PLC program using pre-defined logic blocks and the set of signals connected to each CPU and peripherals.

Those same Visual basic macros generate .csv files that are used by the EPS Taurus UI and PyPLC device server to load variables lists and Dynamic Attributes declaration.

Thanks to that, Taurus User Interfaces and PyPLC attributes are updated whenever the PLC program is modified, with no need of editing the source code.

Many analog and digital signals in ALBA beamlines have been exported to the experiment control framework, Sardana. The Sardana suite allows to add Tango controlled hardware as experimental channels, being able to use them as I/O or experimental data (e.g. temperatures and vacuum pressures). Limited motion control have been implemented using PyPLC, which allowed direct control from Sardana macros used by scientists.

Extending PyPLC Functionality

PyPLC provides a common interface to any device that uses the Modbus protocol, specific States and Status messages can be linked to PLC values using the same syntax of dynamic attributes. But, certain subsystems require a more specific interface to express intermediate states like warning, external interlocks, complex error codes, attributes managed by multiple registers (e.g. multiple-stage pneumatic elements).

This is achieved subclassing PyPLC into new Tango Devices that extend its functionality, in the same way that PyPLC inherits from DynamicDS. Those classes are AlbaPLC (customized to EPS state machines), PLCValve for pneumatic valves and FSOTR for 3-position fluorescence screens. Those classes 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.

Higher-level procedures involving several Tango devices are implemented using macros from PANIC Alarm System[14][15] or Sardana SCADA. Those macros allow the scientists to program automated actions on elements controlled by the EPS (valves, shutters) during experiments.

CONCLUSION

PyPLC Tango Device provides a common interface to all PLC's at ALBA using the Modbus protocol. This developer-friendly interface allowed dynamic and effort-less integration of new PLC signals into our Archiving, Alarm System and Beamlines SCADA (Sardana). This work-flow enabled by PyPLC reduced the time needed to upgrade or modify PLC systems in Beamlines.

ACKNOWLEDGEMENTS

Many former ALBA engineers have collaborated in the PLC-related projects in the last 6 years: D.Fernández, A.Rubio, R.Ranz, R.Montañó, R.Suñé, M.Niegowski, M.Broseta and J.Villanueva. The collaboration of Tango core developer, Emmanuel Taurel, was fundamental in the development of Dynamic Attributes templates and debugging of PyPLC performance.

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] R.Ranz et al., "ALBA, The PLC based Protection Systems.", ICALEPCS'09. Kobe, Japan (2009)
- [5] D.Fernández-Carreiras et al., "Personnel protection, equipment protection and fast interlock systems", ICALEPCS'11, Grenoble, France (2011)
- [6] T.Coutinho et al., "Sardana, The Software for Building SCADAS in Scientific Environments", ICALEPCS'11. Grenoble, France (2011)
- [7] SARDANA website: <http://www.sardana-controls.org>
- [8] R.Sune, E.Taurel and S.Rubio, "Adding Dynamic Attributes to a C++ Device Server", available at www.tango-controls.org (2008)
- [9] D.Fernández et al., "Alba, a Tango based Control System in Python", ICALEPCS'09, Kobe, Japan (2009)
- [10] S.Rubio et al., "Dynamic Attributes and other functional flexibilities of PyTango", ICALEPCS'09, Kobe, Japan (2009)
- [11] Fandango website: <http://www.tango-controls.org/Documents/tools/fandango/fandango>
- [12] PyPLC website: <http://www.tango-controls.org/device-servers/alba/pyplc-device-server/>
- [13] D. Beltran et al., "ALBA Control And Cabling Database", ICALEPCS'09, Kobe, Japan (2009)
- [14] S.Rubio et al., "Extending Alarm Handling in Tango", ICALEPCS'11, Grenoble, France (2011)
- [15] S.Rubio et al., "PANIC, a Suite for Visualization, Logging and Notification of Incidents", PCaPAC'14, Karlsruhe, Germany (2014)

A REAL-TIME DATA LOGGER FOR THE MICE SUPERCONDUCTING MAGNETS

J.T.G. Wilson, STFC Daresbury Laboratory, Warrington, UK

Abstract

The Muon Ionisation Cooling Experiment (MICE) being constructed at STFC's Rutherford Appleton Laboratory will allow scientists to gain working experience of the design, construction and operation of a muon cooling channel. Among the key components are a number of superconducting solenoid and focus coil magnets specially designed for the MICE project and built by industrial partners.

During testing it became apparent that fast, real-time logging of magnet performance before, during and after a quench was required to diagnose unexpected magnet behaviour. To this end a National Instruments CompactRIO (cRIO) data logger system was created, so that it was possible to see how the quench propagates through the magnet. The software was written in Real-Time LabVIEW and makes full use of the cRIO built-in FPGA to obtain synchronised, multi-channel data logging at rates of up to 10 kHz.

This paper will explain the design and capabilities of the created system, how it has helped to better understand the internal behaviour of the magnets during a quench and additional development to allow simultaneous logging of multiple magnets and integration into the existing EPICS control system.

MICE

The Muon Ionisation Cooling Experiment (MICE) is an international collaboration of particle and accelerator physicists from Europe, the US and Japan. It seeks to design, build and operate a muon ionisation cooling channel, which given the consequence of the short muon lifetime that makes traditional cooling techniques inappropriate, is an essential technology for the design of a muon collider or neutrino factory[1].

The MICE cooling channel is of the same design as the cells proposed for the neutrino factory and consists of 3 absorber coil modules with low density absorbers inside a focusing magnetic field and 2 RF-coupling coil modules. It is being built on a dedicated muon beam from the ISIS accelerator at Rutherford Appleton Laboratory.

MAGNET QUENCHES

Upon testing of the first Focus Coil magnet a series of unexpected magnet quenches were occurring meaning that the magnets were not able to reach the power levels specified by the design requirements.

There was already a quench detection system installed, however, this was only designed for machine protection,

shutting down the magnets in the event of a quench and was thus not designed to monitor all of the individual coil power levels. This also led to doubts as to whether there were actually magnet quenches occurring or if the quench detection system was not functioning correctly. To be able to properly address the unexpected quenches and prove the integrity of the quench detection system it was decided that a further diagnostic tool was required to monitor the power levels on each of the coils of the magnet so that the starting point of a quench could be determined and it's propagation through the rest of the magnet analysed.

The proposed solution to this was to create a standalone logging system that could capture magnet performance data before, during and after a quench. Because of the unexpected and unpredictable nature of the magnet quenches (testing could be running for hours before experiencing a quench) a system that simply logged the values from the magnets as soon as they were turned on would create far too much unnecessary data. Similarly, a system that started logging only once it had received the signal from the quench detector would miss vital information because data showing the quench starting to build up on the coils was needed for diagnosis of the fault. The solution to this was to have a system with a 'rolling capture window'. This 'window', or buffer, would temporarily save the data (i.e. in RAM) and once the buffer was full would start to overwrite the oldest data in the buffer with the newest. This would allow for the system to have already captured and be temporarily holding the data showing a voltage differential building up to a quench which, after receiving a signal from the quench detection system, it could amend with the data during and after a quench.

THE DATA LOGGER SYSTEM

Hardware

CompactRIO is a reconfigurable embedded control and acquisition system. The CompactRIO system's rugged hardware architecture includes I/O modules, a reconfigurable FPGA, and an embedded controller [2].

For this application it was decided that NI 9222 4 channel C-series modules [3] were needed, despite the greater cost than other similar analog cards, to achieve the necessary sample rates. These cards can provide up to 500 kS/s per channel at a 16-bit resolution. These were coupled with the NI 9103 chassis [4] which provides 4 C-series module slots and the NI 9012 controller [5].

Additional external electronics were needed to reduce the input voltages down to the +/- 10 Volts that the NI

9222 cards can handle. All of this was then packaged up into a 4U rack mountable crate.

Software

The cRIO is designed to be programmed using LabVIEW. LabVIEW is a graphical programming language created by National Instruments and is designed to make programming and configuration of their hardware quicker and simpler for the user.

Specific modules of NI LabVIEW were needed for this project, the LabVIEW real-time and LabVIEW FPGA modules which allow for configuring the on-board FPGA chassis using the same tools and methods available in standard LabVIEW as well as programming the controller for real-time processing by providing a real-time operating system.

The overall design of the software follows the State Machine architecture [6]. This is one of the fundamental architectures commonly used by LabVIEW programmers to implement complex decision-making algorithms that can be expressed as state diagrams or flowcharts. Each step of the program, e.g. Initialisation or buffering data, can be defined as a state of the system and depending on either user input or internal calculations will lead onto the next state. This allows for a very modular programming design where the processing is not fixed to a straight processing path but can quickly jump to different states when needed. This is achieved in LabVIEW by use of a case structure contained inside a while loop where each state of the program is a different case of the case structure as shown in Fig. 1. An enumerated value can then be used with a shift register to dictate which case or state will be processed next as a shift register will pass the data input to it to the next iteration of the while loop. Shift registers are also used to pass other data between the different cases of the case structure.

The cases defined for this system are 'Ready' where the data logger is waiting for the user to input the desired values for each of the operational parameters; sample rate, pre-trigger sample time (the length of time the user wants to be able to see a quench building up), post-trigger sample time (the length of time the user wants to see after the quench detection signal has been received) and the name to be used for the next log file. The GUI (called Front Panel in LabVIEW) is accessed using the NI Remote Panel Server [7] which allows the user to access the front panel through a web browser. As nearly all PC operating systems come with a web browser as standard this removes the need for each computer wishing to access the data logger having a LabVIEW program or executable that can connect to the variables and allow configuration.

Once the user has finished selecting these values and pressed the start button the program moves into the 'Initialise' case in which the rolling buffer is created based on the sample rate, the pre-trigger sample time and the post-trigger sample time. It is also the where the log file (of TDMS data type [8]) is created and the FPGA configured.

Next the data logger starts acquiring data and storing it to the buffer. The acquisition and some basic scaling of the data is done at the FPGA level as this provides the highly synchronised data acquisition and the high throughput needed to deal with 16 channels of data at up to 10kHz per channel. The output from the FPGA is stored in an internal FIFO buffer that can be accessed by the controller. Due to the limited space on this FPGA FIFO buffer the controller needs to read off the data in blocks of 300 samples and then store this in its own RAM buffer. This RAM buffer can provide for up to 450,000 samples per channel before needing to overwrite old data. This case in the program is also when the data logger is waiting to receive a trigger from the quench detection

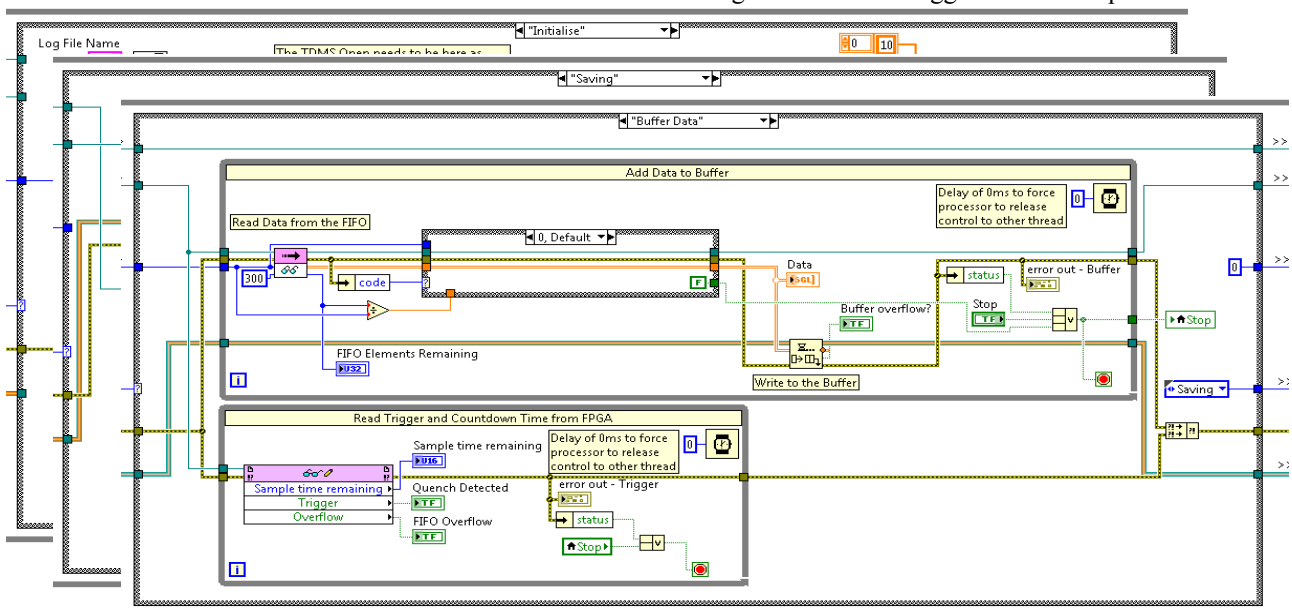


Figure 1: Three of the cases from the Data Logger case structure to illustrate the State Machine architecture.

system. This trigger comes into channel 16 of the data logger and is captured in the same way as the coil data as this allows for it to be saved in the log file synchronised to the coil data. To detect the trigger from the quench detection system a separate process runs in parallel to the data buffering, checking channel 16 for the voltage change that indicates a trigger. Once the trigger is detected the data buffering process continues for the post-trigger sample time specified by the user and then the program proceeds to the 'Saving' case.

The 'Saving' case simply takes data off the buffer and saves it to the TDMS log file created in the Initialise case.

Once all the data has been saved the program moves to the final case where it releases all of the resources it was using and deletes the DRAM buffer ready for the user to specify new values for the configurable parameters and start the logging process again. The users can then FTP into the non-volatile memory of the controller to access the log file.

RESULTS

The data logger was successfully used to prove the integrity of the quench detection system and to show that the magnets consistently, when pushed to the desired operational levels, suffer from a quench. When delivered the second focus coil magnet was tested under the same conditions, using the data logger as a diagnostic tool, and it did not suffer any unexpected issues proving that there was a fault in the first focus coil magnet. The first focus coil is now undergoing repairs to attempt to fix the issue.

FUTURE WORK

Due to the success of the data logger at helping to diagnose the fault in the focus coil it has been decided to integrate it into the final system as a permanent diagnostic tool. Other systems of the same design are being created to monitor other magnets on the machine as well.

Because of the limited size of the RAM on the NI 9012 controller care has to be taken to ensure that the desired buffer size does not exceed what the controller is capable of. Improvements are being made to limit the user to sample rates and sample times that the controller will be able to manage without error. Also, considerations are being made as to the added value of investing in a higher specification controller with more RAM available.

Usage of the data logger also pointed out another weakness in its design that is being addressed. As the data logger is designed to be a standalone system it is up to the user to extract the desired log files from it when they want them. However, this revealed a potential flaw as if the user was not interested in the log file from the last run, and thus did not extract it from the data logger and free up its non-volatile disk space then after a couple of runs the data logger has no more disk space to save new log files too. This causes it to enter an error state and lose the data. There was also a problem of the users starting the data logger running but then remembering to free up the disk space for the log file. When doing this after the data

logger had already started the user could accidentally delete the newly created log file from the controller causing it to enter an error state when it tried to save the data. Both of these issues are being addressed, firstly with a function in the Initialise case that checks to see how many log files are currently being stored on the data logger and deletes all but the newest ones, ensuring that there is enough space for new data to be saved. More thorough error handling is also being added in to recover from the error caused by the user accidentally deleting the current log file by recreating a new file to dump the data to.

It is also desirable to have the data logger integrated with the existing EPICS control system [9] being used on MICE. Due to the limited RAM and processor time on the data logger's NI 9012 controller adding on the EPICS Server support software module that is available would have meant losing some vital buffering capacity. Because of this it was decided to make use of the on-board RS232 port. A simple serial protocol is being created using the LabVIEW VISA library [10] to allow EPICS IOCs to send values for the configurable parameters and start/stop signals. The data logger will in turn be able to send back information on what case it is currently running and any errors that have occurred.

CONCLUSION

This Data Logger system was successful in its purpose. The high specification analog acquisition hardware coupled with the FPGA chassis was able to give the high levels of synchronisation needed for this task. The limitations and difficulties of the system are being addressed within the scope of the future work to create a more robust tool that will be copied several times and is planned to be used throughout the MICE experiment for fault diagnosis.

REFERENCES

- [1] MICE – Muon Ionization Cooling Experiment; <http://www.stfc.ac.uk/208.aspx>
- [2] cRIO – Compact Reconfigurable Input Output; <http://www.ni.com/compactrio/whatis/>
- [3] NI 9222 – 4 Channel Simultaneous analog inputs; <http://sine.ni.com/nips/cds/view/p/lang/en/nid/209142>
- [4] NI 9103 – 4 slot 3M gate chassis; <http://sine.ni.com/nips/cds/view/p/lang/en/nid/14158>
- [5] NI 9012 – Real-Time cRIO Controller; <http://sine.ni.com/nips/cds/view/p/lang/en/nid/14158>
- [6] Application Design Patterns: State Machines; <http://www.ni.com/white-paper/3024/en/>
- [7] Remote Panels in LabVIEW; <http://www.ni.com/white-paper/4791/en/>
- [8] The NI TDMS File Type; <http://www.ni.com/white-paper/3727/en/>
- [9] EPICS – Experimental Physics and Industrial Control System; <http://www.aps.anl.gov/epics>
- [10] NI VISA – National Instruments VISA; <http://www.ni.com/visa/>

BEAM DATA LOGGING SYSTEM BASE ON NoSQL DATABASE AT SSRF*

Y.B. Yan, Y.B. Leng[#], L.W. Lai, Z.C. Chen, H.L. Geng Shanghai
Synchrotron Radiation Facility (SSRF) Shanghai Institute of Applied
Physics, Chinese Academy of Sciences
Shanghai 201204, P.R. China

Abstract

To improve the accelerator reliability and stability, a beam data logging system was built at SSRF, which was base on NoSQL database Couchbase. The Couchbase is an open source software, and can be used both as document database and pure key-value database. The logging system stores beam parameters under predefined conditions. It is mainly used for the fault diagnosis, beam parameters tracking or automatic report generation. The details of the data logging system will be reported in this paper.

OVERVIEW

Shanghai Synchrotron Radiation Facility (SSRF) is a low emittance third generation light source located at Shanghai, China. It includes a 150MeV LINAC, 150MeV to 3.5GeV Booster, LINAC to Booster transfer line, Booster to storage ring transfer line and 3.5GeV storage ring. To improve the accelerator reliability and stability, a beam data logging system was built, mainly based on the beam instrumentation 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 every second. With proper storage and analyze tool-kits, these data could be invaluable. Otherwise the potential of various new electronics will be wasted.

On the other hand, various hardware and software failures have been recorded during 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 entire machine. There were no effective tools to analyze the reason due to lack of adequate raw data. The regular sampling rate of achieved data is about 1Hz. History of broadband data such as turn-by-turn (several hundreds kHz) 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, which stores the raw data under some predetermined conditions, is urgently needed.

* Work supported by the Knowledge Innovation Program of Chinese Academy of Sciences and National Natural Science Foundation of China (No. 11105211, 11305253)
lengyongbin@sinap.ac.cn

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.

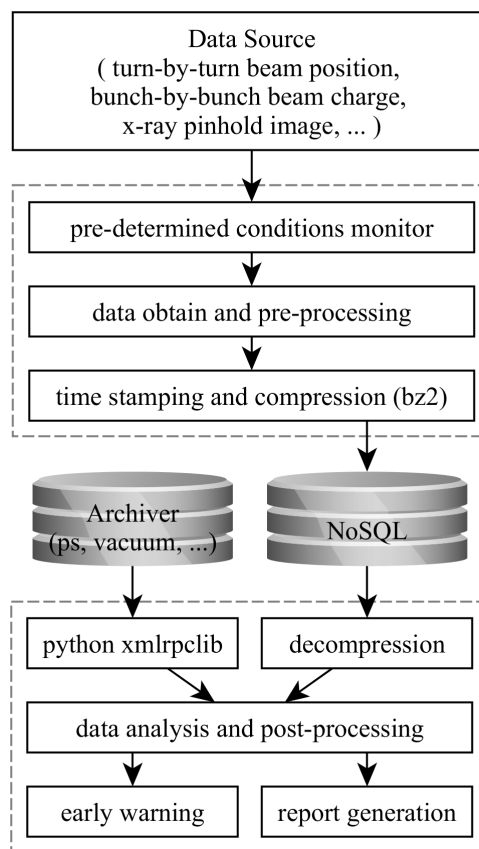


Figure 1: The architecture of data logging system.

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.

Data Source

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 vacuuty,

magnet current, undulator gap or shift, etc.) and scalar, primary beam parameters (such as beam current, close orbit, etc) have been achieved in the regular archiver system. 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 ccd image, etc) and processed data (such as beam spectrum data, beta function measured, etc).

Data Storage

All the data are obtained under some pre-determined conditions, such as the global orbit disturbance, beam injecton, etc. The dedicated routines decide whether the conditions have been satisfied.

Before storing, some of data are pre-processed using some algorithms, then all data under the pre-determined condition are packaged and stored in the database as a 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.

Data Processing

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 raw data. Beside the standard library, Python supports a large number of 3rd party libraries. It makes these algorithms can be implemented easily.

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 output results [4].

The principal component analysis (PCA) 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 an singular value decomposition (SVD) 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 [5, 6].

The post-processing is mainly used for fault diagnosis, beam parameters tracking, which will be discussed in the later part.

Early Warning

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.

Report Generation

For the regular data analysis, some general algorithms and scripts have been used, just called with different arguments. Actually more functions can be also added into the scripts, such as data retrieving from the database (Archiver or Couchbase), data processing and graphing, files generation, etc. According to the requirement, the files will be automatically generated daily, or at some specific time. As the instance, a multi-page document about the beam position distortion during the undulator gap adjustment is given in Figure 2.

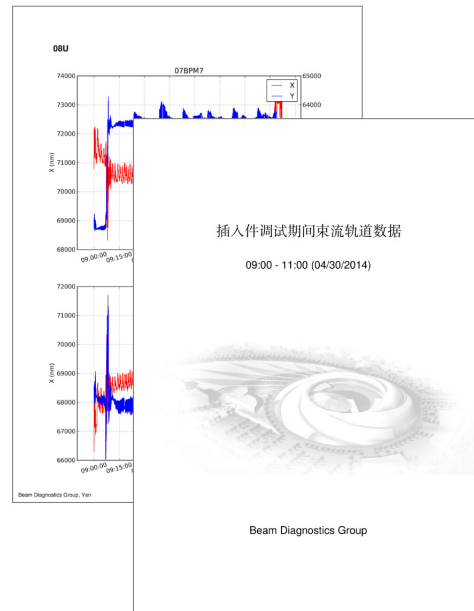


Figure 2: The auto-generated documentation.

The matplotlib and reportlab library are adopted in Python. The matplotlib is a plotting library and can save plot to image file instead of displaying it. The reportlab library allows rapid creation of portable document format (PDF) documents. Its open-source version is available under the BSD license [7]. The final reports include text (such as title, comments, calculation results, etc) and the previous generated images.

FAULT DIAGNOSIS

The particle accelerators are complicated system, with a large number of various components. The fault detection and diagnosis are a long and difficult task, and the data logging system will be helpful.

There is a typical example, which happened after a summer shutdown. The beam position monitor (BPM) 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 3. Finally the cause was found out, a cable connection error. The neighbouring cables (Channel C and D) were cross-connected by mistake.

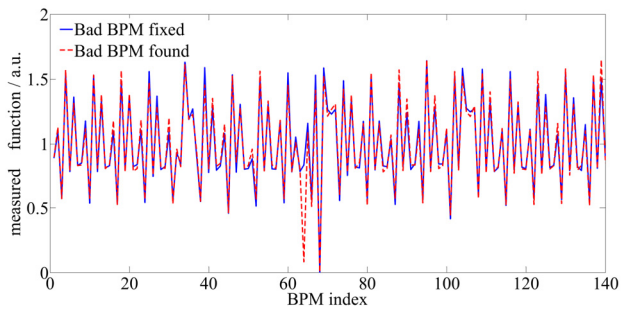


Figure 3: The beta function measured.

PARAMETERS TRACKING

The beam parameter tracking is very important for the accelerator operation or machine studies, such as the tune. Since December 2012, SSRF has 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 excited turn-by-turn orbit data, which is the amplitude of resonance peak of betatron oscillation. The pre-determined condition is the gate signal of top-up injection and one entry is stored. Figure 4 shows the horizontal tune during one day. The rms can be less than $2e-4$ in twelve hours. The jump may be derived from the adjustment of reference orbit.

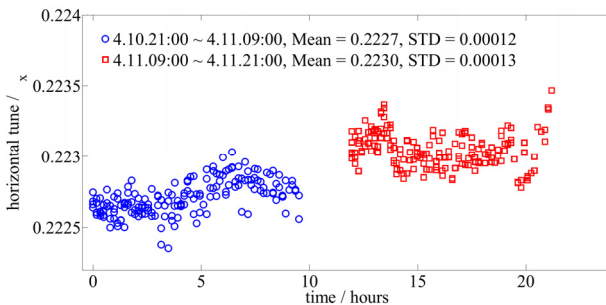


Figure 4: The horizontal tune during 24 hours.

The tune drift over one hundred days are shown in Figure 5. The blue points represent the horizontal tune, while the red represent vertical.

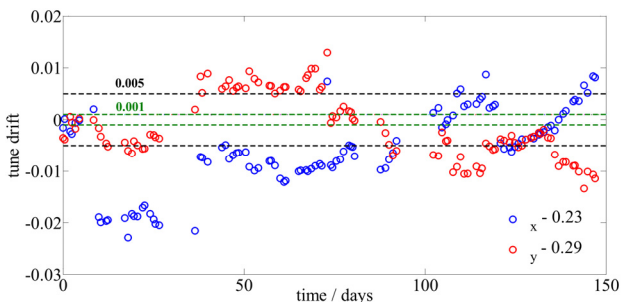


Figure 5: The tune drift during 150 days.

CONCLUSION

The beam data logging system has been implemented base on NoSQL database. More functionality will be added in the future. The preliminary applications show great potential for the fault diagnosis and parameters tracking. 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 analytics will also be 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), <http://www.jacow.org>
- [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), <http://www.jacow.org>
- [3] Couchbase website: <http://www.couchbase.com>
- [4] Z.C. Chen, Y. B. Leng, "Study of Correlation Analysis of Global Data from a Storage Ring", Chinese Academy of Sciences, 2014
- [5] 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
- [6] 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
- [7] ReportLab website: <http://www.reportlab.com>

NEW DATA ARCHIVE SYSTEM FOR SPES PROJECT BASED ON EPICS RDB ARCHIVER WITH PostgreSQL BACKEND

M. Montis, M. Giacchini, S. Fantinel, INFN/LNL, Legnaro, Italy
M. Bellato, INFN sez. Padova, Padova, Italy

Abstract

SPES project[1] is a ISOL facility under construction at INFN, Laboratori Nazionali di Legnaro, which requires the integration between the accelerator systems actually used and the new line composed by the primary beam and the ISOL target. EPICS[2] has been chosen as main Control System framework for the project; as consequence, a migration from the actual control system to the new one is mandatory in order to reuse the actual system for the new facility. One of the first implementation realized for this purpose is the Archiver System, an important service required for experiments. Comparing information and experiences provided by other Laboratories, an EPICS Archive System based on PostgreSQL is implemented to provide this service. Preliminary tests are done with a dedicated hardware and following the project requirements. After these tests, the system is going to be moved in production, where it will be integrated with the first subsystem upgraded to EPICS. Dedicated customizations are made to the application for providing a simple user experience in managing and interact with the archiver system.

INTRODUCTION

In a complex and extended installation like an accelerator, where different sub-systems work simultaneously and share information each other, the requirement of having available all the data under control, both online and offline, is mandatory: it results useful for production (such as post-analysis processes) and maintenance. The Channel Archiver is an archiving application realized for EPICS based control systems where a dedicated machine properly configured can archive any kind of process variable available in the control system network through the transparent communication protocol based on TCP/IP standard provided by EPICS, the EPICS Channel Access.

In the principal laboratory where EPICS is used as main control system framework, the original Channel Access Archiver, designed in 2006, is largely used. However in these few years different new solution based on Database backend are growing and starting to be a new standard for data archive service.

In this scenario, the EPICS RDB (Relational DataBase) Archiver with PostgreSQL Database has been chosen as main archive service for the SPES Project.

THE EPICS RDB ARCHIVER

During initial development, the test bench realized for the archiver system was composed by a single server equipped with all the hardware and the software required for a stand-alone test bench. In a next step, the hardware involved in the study case was upgraded and extended, in order to provide a full machine ready for production. The last set of tests has done using a part of the real control system environment under upgrade at LNL[3][4].

Hardware

The machine used to tests and deploy the archive service is a server equipped with 2 esa-core processor working in Hyper Treading, 32GB RAM and 2TB disk space, in order to have the maximum resources available both for development and production steps. It also has redundant power supply as required in this kind of environment.

Preliminary tests were performed on this host, using it as EPICS server (running an IOC) and EPICS client (the archiver application) due to analyse the correct configuration of the client tool. Later, extending tests to a real environment where different EPICS server provides different Process Variables (PVs), the hardware used for this represented in Figure 1.

Software

The focus followed in the software definition was having a machine with the minimum amount of applications and services required to perform this task, due to minimize the maintenance.

Following the guidelines adopted by LNL related to production hosts, RHEL compilant Linux was chosen as Operative System for the Archive machine. Over this OS, EPICS base was installed for providing the environment required to execute tests. At the same time, this machine was used to compile and develop the Archive tools (Archive Config Tool and Archive Engine) used to realize the final service; as consequence, a minimal graphic interface was installed for working with the Eclipse IDE and the EPICS CSS source code.

For the Archiver system, source code related to SNS CSS version 3.1.5 has been used. As suggested Ruizhe

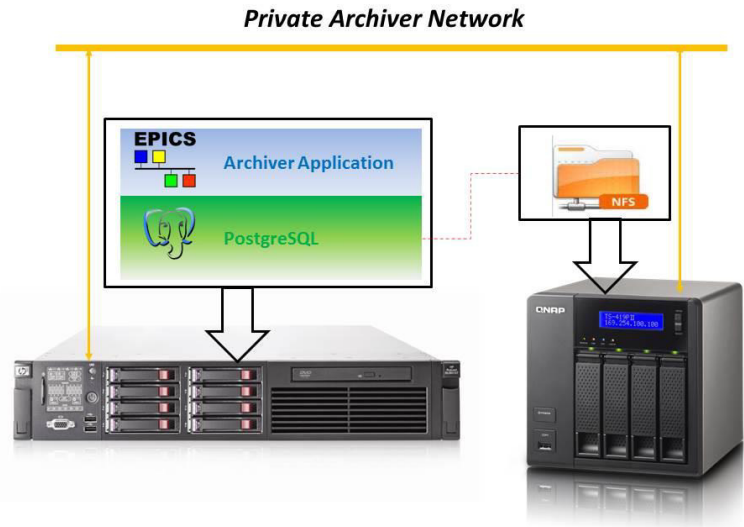


Figure 1: EPICS RDB Archiver schema: hardware and software involved to realize archive service.

Ma in his document[5], also source code stored in CSS Sourceforge repository[6] is required to compile properly part of the Archiver application. All the applications required have been compiled for Linux targets.

ARCHIVER APPLICATION AND CONFIGURATIONS

The EPICS RDB Archiver System results composed by different pieces (tools and applications) required to manage the communication between the PostgreSQL Database and the EPICS environment:

- *ArchiveConfigTool* let developer to automatize the Database configuration with all the information required to define a proper Archive Engine. For the SPES Project, different configuration XML files, used to describe Engines structure, are defined. Not particular configurations are required for this tool up to now.
- *ArchiveEngine* is the main application which defined the mechanisms of data storing and retrieving. For this application different Engine's profiles have been defined in order to customize and optimize host's resources. In this moment, all the experience acquired during the test phase was used to define the performance required for the ALPI-PIAVE diagnostics system. At the same time, because this system is still used as complex test bench, these configurations are not completed and frozen.

Because of the ArchiveEngine is a command line application which runs in foreground by default, a dedicated script was developed to realize a standalone service. This solution, merged with a dedicated network monitoring service, let system administrators supervise easily the status of the archiver system, minimizing the maintenance. In addition, for administrators and end

users, a dedicated main page, available through Apache webserver running on the archiver host, provides all the minimal information and links related to the Archiver service.

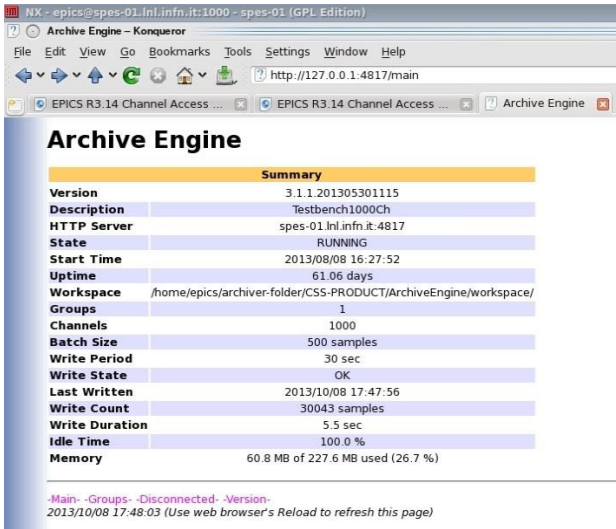


Figure 2: Archiver Engine used during tests: in this case a soft IOC running 1000 records with SCAN 1 second.

To start and stop archiving operations, a dedicated softIOC is prepared to provide the minimum set of records needed to manage the data storage (Figure 3): one binary record placed into the Control Panel related to a dedicated sub-system let user to enable or disable data acquisition, in order to save disk space when experiments are not executed. This binary input record controls several records placed in each Engine Group, starting and stopping them. This mechanism with double record (one main enable + one group enable) gives to users

(developers and maintainers) a high degree of freedom in customizing the archiver environment for particular operations.

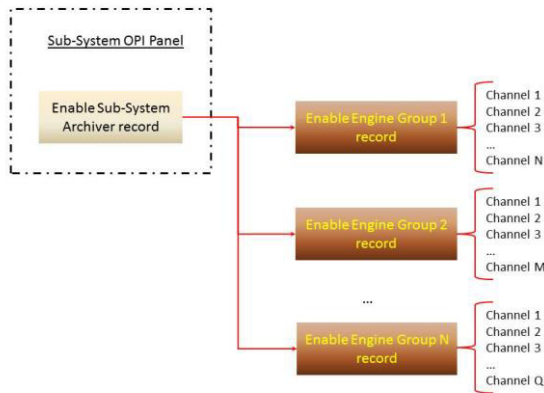


Figure 3: Database schema used to enable and disable data archiving without stopping the archive service.

TESTS PERFORMED

To prove the efficiency of the Archiver system, several tests have been performed using different configuration.

Table 1: Test Performed

Number of channels	Number of IOC	Parameters observed	Test period [months]
1000	1	<ul style="list-style-type: none"> DB partition Write/Read Time Data loss 	3
2000	1	<ul style="list-style-type: none"> Write Buffer Write Time Data loss 	1
310	9	<ul style="list-style-type: none"> Write Buffer Write Time Network config Time Synchronization 	2 (still under test)

Analysing the results obtained with the first test and comparing them with the information provided by M. Konrad[7], a database partitioning based on weekly sub-tables is coherent with end user's request and doesn't compromise its complexity. At the same time, the high volume of data managed by the service doesn't affect it, providing a reasonable write time period: during the test with 2000 channels having SCAN=1s, the archiver set with:

- Write Period: 30 s
- Batch size: 60000 samples

writes the entire buffer in 11.4s, without data loss. Moving the tests to a real scenario, PostgreSQL Database's network interface was properly set to communicate only with a well-defined set of machines available in the control system network, in order to avoid unauthorized intrusion. The engine's configuration parameters found during first tests are still under test in this second part but, up to now, feedbacks confirms the behaviour expected. More detailed results will be available in the next period.

CONCLUSION

The archiver system is a mandatory service required for the SPES project and a brand new feature for the actual facility. A big amount of time was spent to study and analyse its performances and behaviour in order to find a good configuration (hardware and software) for the production phase.

First tests performed with the ALPI-PIAVE diagnostics system confirm the good approach adopted during the first stage and let developer to define a starting point for integrate all the remaining sub-systems composing the ALPI-PIAVE facility. At the same time, custom configurations will be improved for providing a better experience for system-administrators and end-users.

ACKNOWLEDGMENTS

This works leveraged of years of experience on EPICS use from good engineers of other laboratories around the world: great acknowledgments to them.

REFERENCES

- [1] SPES Project website: web.infn.it/spes/
- [2] EPICS website: <http://www.aps.anl.gov/epics/>
- [3] B. Liu et al., "Upgrade of Beam Diagnostics System of ALPI-PIAVE Accelerator's Complex at LNL", WPO018, PCAPAC'14, Karlsruhe, Germany (2014)
- [4] M. Giacchini et al., "Magnet Power Supply Control Mockup for the SPES Project", WPO016, PCAPAC'14, Karlsruhe, Germany (2014)
- [5] Implementation and testing of RDB archiver with MySQL: http://www.illinoisacceleratorinstitute.org/2011%20Program/student_papers/Ruizhe_Ma.pdf
- [6] CSS Sourceforge website: <http://sourceforge.net/projects/cs-studio/>
- [7] M. Konrad et al., "Control System Studio Archiver with PostgreSQL Back-End: Optimizing Performance and Reliability for a Production Environment", WEPD03, PCAPAC'12, Kolkata, India

DEVICE CONTROL DATABASE TOOL (DCDB)

Pavel Maslov, Matej Komel, Klemen Žagar, Cosylab, Ljubljana, Slovenia

Abstract

In a physics facility containing numerous instruments, it is advantageous to reduce the amount of effort and repetitive work needed for changing the control system (CS) configuration: adding new devices, moving instruments from beamline to beamline, etc. We have developed a CS 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, etc. 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, which automatically loads and links device support libraries. The DCDB tool is compliant with CODAC (used at ITER and ESS), but can also be used in any other EPICS environment.

DCDB ARCHITECTURE

The DCDB-tool uses MySQL relational database together with the BLED [1] schema (a set of tables representing the whole facility).

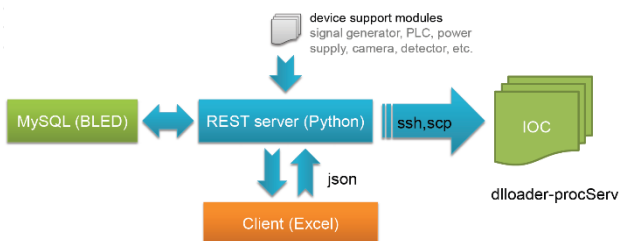


Figure 1: DCDB architecture.

The backend is a typical web-server, 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 [2]. The client-server communication is based on the exchange of JSON objects (strings).

DEVICE SUPPORT MODULES

Device support modules are created using standard commands that come with the Maven ITER plugin and Java API for managing CODAC development unit life-

cycle (packages m-iter-plugin, m-codac-unit-api), supplemented by dlloader support (Fig.2).

```
bled@bled:~$ mvn newunit -Dunit=m-BeamPositionMonitor
bled@bled:~$ cd m-BeamPositionMonitor
bled@bled:~$ mvn newdlloader
bled@bled:~$ mvn clean compile test package
```

Figure 2: The procedure of creating EPICS device support libraries.

Device support modules should contain a library (lib/*.so), an EPICS database definition file (dbd/*.dbd files), an EPICS database template/substitution file (db/*.db) and three scripts: init.cmd (contains the string *require <module>, <version>*), init-pre.cmd (a set of configuration fields or macros to be extracted into the BLED database, plus epics shell commands to be executed before IOC initialization) and init-post.cmd (epics shell commands to be executed after IOC initialization).

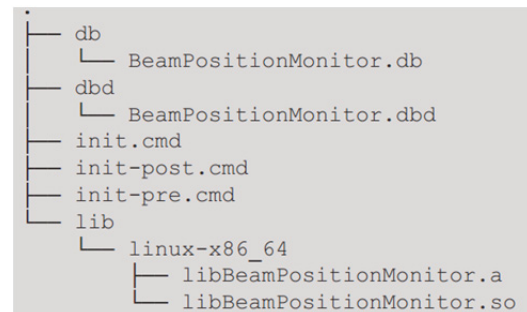


Figure 3: Files to deploy.

In order to load the information about the device support module into the BLED database, use the *bled* helper script that you should invoke from the module's \$(TOP) directory. It will extract the information about the module stored in the pom.xml and init-pre.cmd files and send it to the REST server over the network.

DYNAMIC LIBRARY LOADER

Dynamic library loader (or dlloader)¹ is an EPICS-based tool that allows you to load EPICS device support libraries by just adding its' definitions in the startup script (st.cmd). Hence, the integrator is freed from the necessity of compiling (recompiling) their EPICS applications/IOCs.

The DCDB-tool uses dlloader to dynamically load device support modules (described in the previous chapter). As it was already mentioned there is no need to recompile the source code, since all we have to provide is a st.cmd script, which in turn is generated by the REST server.

1. The concept of dynamically loadable device support modules, including the require function is developed by Dirk Zimoch (PSI).

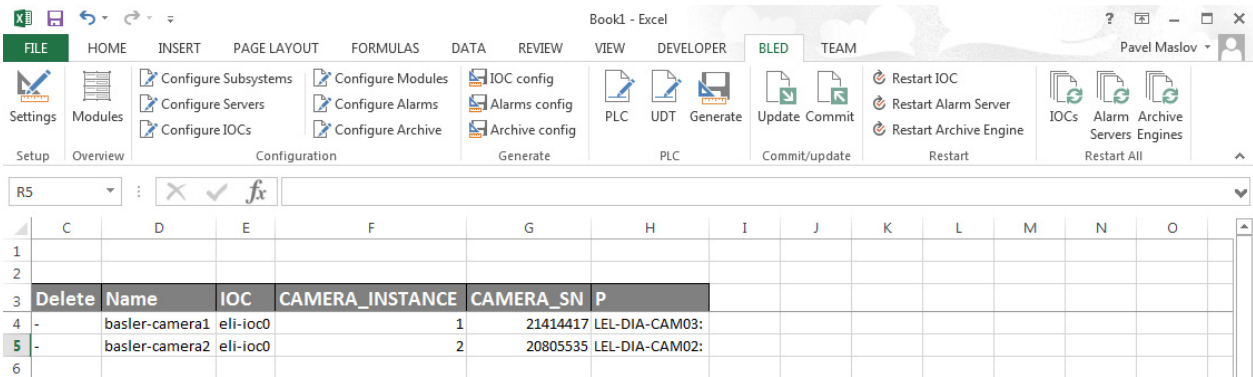


Figure 4: Front-end application (adding device instances).

Module	Version	IOC	Instances
TimingReceiver	test	bled	1
DataAcquisition	test	bled	4
BeamPositionMonitor	test	bled	1
fug	0.0a1	iter-fc2	1
lambda	0.0a1	iter-fc2	1
iocmon	0.0a1	bled	1
iocmon	0.0a1	eli-ioc0	1
basler	0.0a1	eli-ioc0	2
sysmon	0.0a1	bled	1
sysmon	0.0a1	eli-ioc0	1

Figure 5: Front-end application (showing all modules registered in the database).

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, using which you can easily edit your support modules' configuration (Fig. 4), configure IOCs, alarm servers and archive engines; start/stop/restart IOCs, and more.

Figure 4 shows an Excel configuration sheet for the Basler camera device support module with two camera instances running on the *eli-ioc0* IOC.

Another interesting feature of the DCDB-tool is when you click on the Modules button (Fig.5, top left corner), you can see the information about all the device support modules registered in the BLED database: which versions and which IOCs they are activated on, plus the number of instances. This can be handy when you want to see an overview of what is deployed on which IOC and in which quantities.

Along with the hardware modules, you can also have "soft modules" deployed with *dlloader* support (notice *iocmonitor* and *sysmonitor* in Fig.5).

The front-end application was tested with Microsoft Excel 2007 and 2013.

SUMMARY

The DCDB tool is a powerful control system configuration tool that saves a lot of integration effort (and thus, time). It is developed with best practices from the EPICS community, compliant with CODAC core system (used at ITER, ESS, ELI-NP), but can also be used in any other EPICS environment.

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] BLED – system for central management of the accelerator data (ESS, Cosylab d.d.)
- [2] procServ – Process Server (written by Ralph Lange); <http://sourceforge.net/projects/procserv/>

STATUS OF OPERATION DATA ARCHIVING SYSTEM USING Hadoop/HBase FOR J-PARC

N. Kikuzawa[#], H. Ikeda, Y. Kato, N. Ouchi 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 much equipment. In the Linac and the 3 GeV rapid cycling synchrotron ring (RCS) in J-PARC, data of about 64,000 EPICS records have been collected for control of these equipment. The data volume is about 2 TB every year, and the total data volume stored has reached about 10 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.

We are planning to replace PostgreSQL with Apache Hadoop and Apache HBase to accumulate enormous operation data produced from the Linac and the RCS in J-PARC. HBase is so-call NoSQL, which has scalability to data size at the cost of the high broad utility of SQL. HBase is constructed on a distributed file system provided by Hadoop, a cluster with advantages including automatically covering its cluster nodes' breakdowns and easily adding new nodes to expand its capacity. The new database system satisfies high availability, high performance, and high flexibility of storage expansion.

The purpose of this paper is to report the present status of this archive system.

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 using Apache Hadoop [2], a distributed processing framework, and Apache HBase [3], a distributed database [4].

Hadoop is a widely used open-source cloud framework for large scale data processing. The HBase is a distributed, scalable big data store on a cluster built with commodity hardware. One of the cores of Hadoop is a file system called HDFS (Hadoop Distributed File System), and HBase runs on it. Hadoop and HBase are scale-out architecture, and we can expand storage volume dynamically by adding new nodes. Moreover, they are designed based on the assumption of frequent breakdown

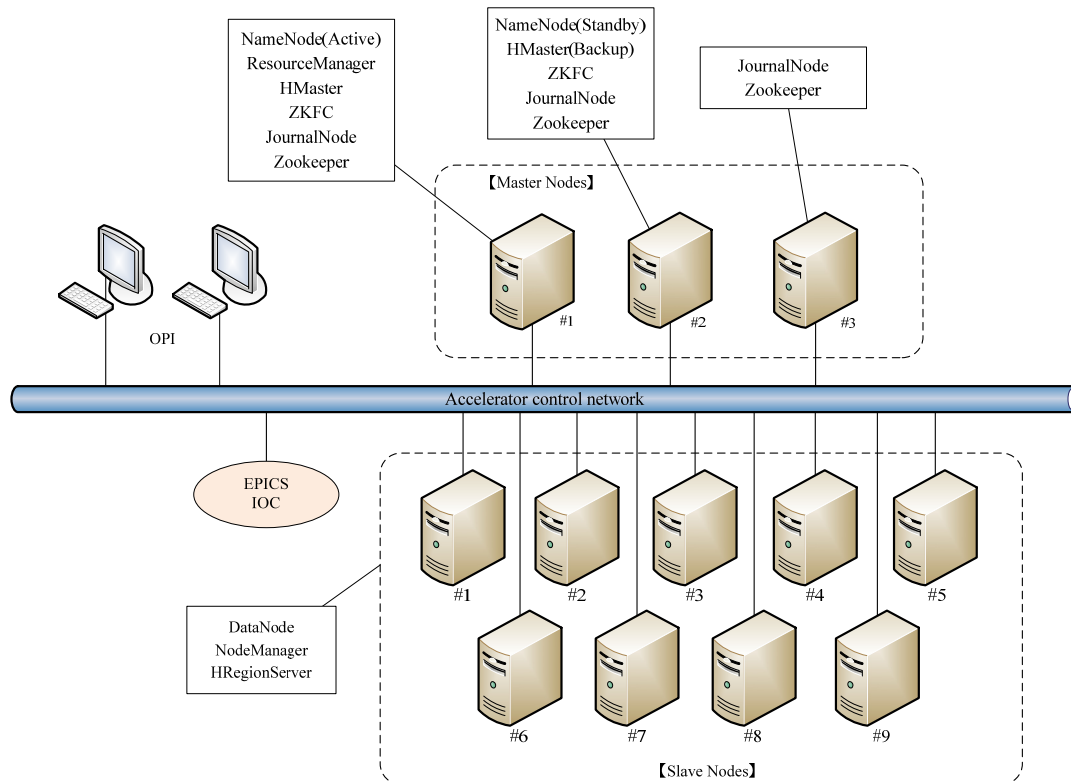


Figure 1: System configuration of the new archiving system.

[#]kikuzawa.nobuhiro@jaea.go.jp

of cluster nodes in large scale clusters, and they have a tolerance for the breakdowns and are easy to recover.

HBase is a type of "NoSQL" database and its scalability is established under restrictions on such functions as transaction, table combination, etc. that an ordinary RDB has. That means HBase doesn't take the place of RDBs and we should take account of trade-offs between them.

Having considered that HBase is the best for the time series data archiving system, we built and tested a small testing system, and we have updated the system with new versions of Hadoop/HBase that provide indispensable features. We will describe the present status of this new archiving system.

DATABASE SYSTEM CONFIGURATION

Hardware Configuration

Hadoop and HBase are designed to have master-slave architecture, and we use our machines as master nodes and slave nodes, with appropriately deploying their software components. We have built a cluster composed with nine slave nodes, each of which is a commodity server with four 2 TB local hard disk storages configured in RAID 5, and is interconnected using Gigabit Ethernet (GbE). About 50 TB of effective capacity contributes to the HDFS. Figure 1 shows the system configuration.

Update of Hadoop and HBase

With early versions of Hadoop/HBase we made a testing system by using Heartbeat [5], Pacemaker [6], and DRBD (Distributed Replicated Block Device) to add redundancy to NameNode. NameNode is one of the components deployed in a master node and manages metadata of the distributed file system, and prior to Hadoop 2.0.0 it was a single point of failure (SPOF), which is not acceptable for practical purposes.

One of the problems is, this is just cold standby and NameNode might take a quite long time to complete starting up, preventing you from accessing the cluster. It took about 5 minutes for failover and failback in our system, since it was necessary to perform starting of the HBase system service after the initialization processing of the Hadoop system service start-up is completed [7]. It might take an hour, depending on the size of the cluster. That is far from high availability.

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 and JournalNode. ZooKeeper [8] 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 machines, and it is meaningful to deploy them on an odd number of machines. Table 1 shows the spec of our system. For now we don't prepare sufficient machines for the 2nd and 3rd

Table 1: Spec of the New Data Archiving System

Master node #1	DELL PowerEdge R610 CPU: Intel Xeon E5620 (4Core 2.4GHz) MEM: 24GB HDD: 600GB SAS 10 x 4 (RAID10)
Master node #2	DELL PowerEdge R200 CPU: Intel Xeon E3210 (4Core 2.13GHz) MEM: 8GB HDD: 160GB x 1
Master node #3	DELL PowerEdge 860 CPU: Intel Xeon X3210 (2Core 2.4GHz) MEM: 4GB HDD: 250GB x 1
Slave Nodes	DELL PowerEdge R410 CPU: Intel Xeon E5620 (4Core 2.4GHz) MEM: 24GB HDD: 2TB x 4 (RAID5)

Table 2: Components Deployment (Master Nodes)

Master Node	#1	#2	#3
NameNode	o	o	
Journal Node	o	o	o
ZKFC	o	o	
Resource Manager	o		
History Server	o		
ZooKeeper	o	o	o
HBase Master	o	o	

master nodes, and we are planning to enhance them in preparation for the practical operation phase. Table 2 shows components deployment of the master nodes.

The slave nodes have been built on RAID 5, because we have focused on making clear the procedures to restore nodes rather than evaluating the performance of the system. However, any RAID makes the disks to cooperate and prevents simultaneously accesses, and has possibility to give significant impacts to the performance of Hadoop/HBase. We are planning to unbind RAID in slave nodes and reevaluate.

As for the version of HBase, we developed our system with HBase 0.94.x, but even the documents of HBase don't make clear whether the HBase 0.94.x is compatible with Hadoop 2.2.0. For this reason, we have adopted HBase 0.96.1.1, which targets Hadoop 2.x from the beginning.

Table Structure

The present system uses PostgreSQL to store much data. PostgreSQL had various restrictions about data size, and the system was designed to divide the large amount of the data into multiple tables, with dynamically creating tables, according to a group the data belongs to and its monitoring time. That drops many advantages of the

RDBMS, and introduces complexity and restrictions into logic to store and retrieve data to/from the system.

HBase is a type of column-oriented database, and a simple structure of key-value is suitable. It is possible to have huge size tables as compared with the conventional database system. That means a schema design in the column-oriented database is very different from one in an ordinary RDB.

In HBase each record is identified with a binary sequence referred to as a row (a primary key in terms of RDBs) and is stored in ascending order of rows. Basically rows are the only index and the design of the rows is directly related to the performance to retrieve records. The design is also related to load distribution to store records as follows; Records are divided by automatically or manually selected rows into regions, which are distributed among nodes in a cluster. If two rows start with the same value, the records tend to be located in the same region, and consequently in the same node. For an instance, if you design the row which starts with the same value followed by a monotonically increasing value like a timestamp, when you are going to store multiple records they tends to be written in the same server [9], which results in slow writing speed.

We have carried out examinations about table structure [10]. In order to avoid the above problems, we have designed the row that starts the binary expression of the EPICS record name followed by the binary expression of its measured time. The time is represented by progress milliseconds of the 1970 UTC epoch, and in order to search the latest data first, its binary representation is decided to subtract the time from the maximum of a signed 64-bit integer.

DATA BROWSER

An application has been developed which acquires data from the Linac/RCS and stores in HBase. The acquisition of data is performed via the EPICS channel access. The storing in HBase may be blocked temporarily, and the application has a buffer.

Another application has been also developed which retrieves the data stored in HBase [11]. The application is in the form of a plug-in into Control System Studio (CSS) [12].

This plug-in extends the class ArchiveReader in the plug-in org.csstudio.archive.reader provided by CSS, and you can refer data in HBase via Data Browser, one of the convenient GUI components in CSS. Figure 2 shows an example of the Data Browser plot.

The performance of these tools is being checked after setting up the cluster correctly.

SUMMARY

We have proposed the next-generation archive system using Apache Hadoop, a distributed processing framework and Apache HBase, a distributed database. With new versions of Hadoop and HBase, it has turned out that the system needs to be revised. Data archiving and retrieval tools have been developed. The performance of the tools is being checked after revising the system.

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. [in Japanese]
- [2] <http://hadoop.apache.org/>
- [3] <http://hbase.apache.org/>
- [4] N. Kikuzawa et al., "Development of J-PARC Time-Series Data Archiver using Distributed Database System", Proceedings of ICALEPCS2013.
- [5] <http://www.linux-ha.org/wiki/Heartbeat>
- [6] <http://www.linux-ha.org/wiki/Pacemaker>
- [7] A. Yoshii et al., "Status of J-PARC operation data archiving using Hadoop and HBase" Proceedings of the 10th Annual Meeting of Particle Accelerator Society of Japan. [in Japanese]
- [8] <http://zookeeper.apache.org/>
- [9] Apache HBase Reference Guide <http://hbase.apache.org/book.html>
- [10] 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. [in Japanese]
- [11] N. Kikuzawa et al., "Development of tools for the J-PARC operation data archiving using HBase/Hadoop", Proceedings of the 11th Annual Meeting of Particle Accelerator Society of Japan. [in Japanese]
- [12] <http://controlsystemstudio.org/>

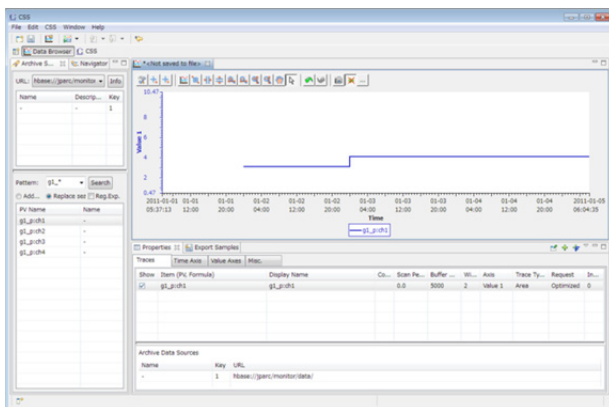


Figure 2: Example of data browser plot.

MANAGING MULTIPLE FUNCTION GENERATORS FOR FAIR

S. Rauch, M. Thieme, R.C. Bär, GSI, Darmstadt, Germany

Abstract

In the FAIR control system, equipment which needs to be controlled with ramped nominal values (e.g. power converters) is controlled by a standard front-end controller called scalable control unit (SCU). An SCU combines a ComExpressBoard with Intel CPU and an FPGA baseboard and acts as bus-master on the SCU host-bus. Up to 12 function generators can be implemented in slave-board FPGAs and can be controlled from one SCU.

The real-time data supply for the generators demands a special software/hardware approach. Direct control of the generators with a FESA (front-end control software architecture) class, running on an Intel Atom CPU with Linux, does not meet the timing requirements. So an extra layer with an LM32 soft-core CPU is added to the FPGA. Communication between Linux and the LM32 is done via shared memory and a circular buffer data structure. The LM32 supplies the function generators with new parameter sets when it is triggered by interrupts. This two-step approach decouples the Linux CPU from the hard real-time requirements. For synchronous start and coherent clocking of all function generators, special pins on the SCU backplane are being used to avoid bus latencies.

DESCRIPTION OF SCU AND FG

The quadratic function generator (FG) which is described in this paper, is a VDHL macro that runs in SCU bus slave cards. At the moment, there are three slave cards with this feature: DIOB (1 FG), ADDAC1 (2 FGs) and ADDAC2 (2 FGs). The DIOB card has an digital output with 32 Bit for the FG output value. The two ADDAC cards offer an analog output with 16 Bit resolution for the FGs. The slave cards are controlled via the SCU bus from the Scalable Control Unit (SCU). The SCU is a FPGA based controller equipped with a ComExpress Board which runs linux. The communication between FPGA and ComExpress Board is done via PCIe. Inside the FPGA is a System-on-Chip (SoC) on basis of a wishbone [1] crossbar with a PCIe-to-wishbone bridge (wishbone master). The SCU bus is connected with a bridge too, that acts as a wishbone slave. Part of the SoC is LM32 cluster with a configurable number of softcore processors and shared memory (see Fig. 1). A separate crossbar is used for message signaled interrupts (MSI). A interrupt master, e.g. SCU bus bridge, sends MSIs to an interrupt slave. That can be a interrupt queue of an LM32 or the PCIe bridge. With the use of MSIs, the interrupt system is quiet flexible, because every slave can address every interrupt target.

The SCU bus is a parallel bus with 12 slave slots. The data and address lines are each 16 Bit wide. Each slave has a separate IRQ line. Inside the SCU bus bridge the IRQs are translated into MSIs. The system should be used as an arbitrary function generator with 12 independent channels. Each

channel will control equipment that needs ramped nominal values. That means for example power converters and Direct Digital Synthesis (DDS) systems. The FG is configured with a set of data and interpolates then a predefined number of output values. After the interpolation is started, the FG waits for the next set of data that is provided by the linux FESA class. A brief hardware description of the FG can be found here [2]. In contrast to the older paper, a few things had to be changed for the implementation. The data path is now 64 Bit wide and both parameters, the linear and the quadratic one, can now be shifted in a 64 Bit range.

FG Inside SCU

Other then for ramped power converters, the SCU will be used to control DDS systems. This will be done with FIB cards, which are supplied from the radio frequency group. These FIB [3] cards are used as SCU slaves. But in there current revision the are not able to run a FG macro in the slaves. So a different solution had to be found. The same FG macro as used in the slaves is put behind a wishbone interface and connected to the crossbar of the SCU. The output of that FG is connected to a special wishbone master, that splits the 32 Bit output value from the FG into two 16 Bit accesses to the SCU bus. So the interpolation of the FG is done inside the SCU, instead of inside the SCU slaves. Because this modus uses a lot more bandwidth than the slave approach, the SCU bus should only be used for sending FG values. The FG macro with wishbone interface acts exactly like the FG in the slaves, only the interface is different. For the software layer they look identical.

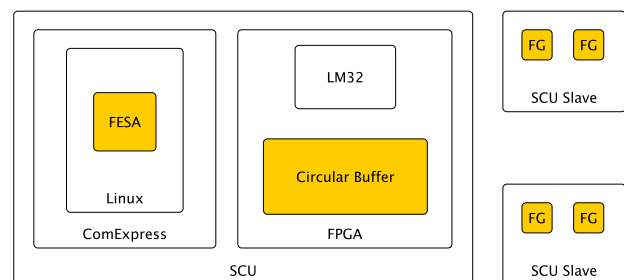


Figure 1: SCU with two slaves.

DATA SUPPLY WITH REAL TIME BOUNDARIES

The FG can be configured to interpolate in steps from 250 up to 32000. The sample frequency is configurable from 16 kHz up to 1 MHz. If the FG should now sample with 1 MHz for 250 steps that means the linux program has to provide a new data set every 250 μ s. This data rate is to high for linux to be serviced reliably for 12 channels.

IMPLEMENTATION WITH LM32 AND MSI

As direct control of the FG macro from linux is not possible, another real time layer is inserted. One of these LM32s is now used for configuring the FGs and supplying new data when triggered by the FG. Because each of the LM32 soft-cores has a dual port memory, that is accessible for the other LM32s and the linux system via the PCIe bridge, communication is done via circular buffers in shared memory. The software running on the LM32 is written in C and needs no operating system. So the real time behavior is easy to predict. The real time processing is done in a IRQ handler, that supplies the FGs with data. The round trip time for each interrupt from the source (FG macro) to the IRQ handler and then back again is around 5 μ s. One SCU bus access is 300ns long. With six SCU bus accesses needed for every parameter set, that makes 18 μ s for the handling of each FG. With a step width of 250 theoretically 13 FGs could be serviced. In reality the smallest step width used, will be 1000. That gives enough time for handling the 12 FGs needed.

For controlling the LM32 software, another IRQ handler is used. Here the linux software generates MSIs by writing to LM32 interrupt queues. This works as a software interrupt scheme.

FG OPERATION

After reset the LM32 software scans the SCU bus and enumerates the FG macros found in the slave cards. In addition Up to 12 FGs are supported. These virtual FG devices are then presented to the FESA class. A virtual device number (0-11) is stored in each FG to make it easier to address the right FG, when an interrupt occurs. Each virtual device has its own circular buffer, that is filled by the FESA class and emptied by the IRQ handler. When the buffer is empty, the sampling of the FG halts.

The software sends the first parameter set with the start value to each FG. After that, the FGs can be started with a broadcast write to the SCU bus or with a signal from a

timing event. Directly after the start, the FG send a data request for the next parameter set. The signaling of the data request is done with the IRQ feature of the SCU bus and the MSI system of the SCU. Encoded in the MSI message is the number of the slave card, that has triggered a data request IRQ. Because there can be two FG macros in one slave card and the slaves use a shared IRQ scheme, the handler has to ask the slave, which macro has triggered the data request. It then reads the virtual FG number, to select the right buffer for the FG. Then the handler sends the next parameter set from the buffer and acknowledges the IRQ.

PROJECT STATUS AND FUTURE WORK

The hardware implementation of the FG macro in both modes (SCU slave/Wishbone slave) is done. A test program exist under linux, that is able to start the FGs with arbitrary ramp data supplied by a text file. The integration into FESA is in progress.

A Data Acquisition (DAQ) system for the slaves is planned, that can make use of the same infrastructure as the FG. Only the direction of the data flow would be the other way around.

At the moment the data transfer between linux and the LM32 runs in polling mode. The linux program constantly tries to fill the circular buffer. This wastes lots of bandwidth. It would be more effective, if the LM32 signals to the linux program, that the buffer needs to be refilled. That can be done with an interrupt to the PCIe bridge.

REFERENCES

- [1] Wishbone B4: http://cdn.opencores.org/downloads/wbspec_b4.pdf
- [2] S. Rauch et al., "Improved Function Generator for Device Control for the GSI Control System", TUP006, <http://jacow.org/pc08/papers/tup006.pdf>, PCaPAC08, Ljubljana, Slovenia (2008).
- [3] M. Kumm et al., "Realtime Communication Based on Optical Fibers for the Control of Digital RF Components", GSI Scientific Report (2007).

SETUP AND DIAGNOSTICS OF MOTION CONTROL AT ANKA BEAMLINES

Karlheinz Cerff, David Haas, Denis Jakel, KIT-ANKA, Markus Schmitt KIT-IBG, Karlsruhe, Germany

Abstract

The precise motion control in high resolution [1] is one of the necessary conditions for making high quality measurements at beam line experiments.

At a common ANKA beam line, up to one hundred actuator axes are working together to align and shape beam, to select beam Energy and to position probes. Some Experiments need additional motion axes supported by transportable controllers plugged temporary to a local beam line control system.

In terms of process control all the analogue and digital signals from different sources have to be verified, levelled and interfaced to the motion controller. They have to be matched and calibrated in the control systems configuration file to physical quantities which give the input for further data processing. A set of hard- and software tools and methods developed at ANKA over the years is presented in this paper.

INTRODUCTION

Building a new beam line, the motion control setup is accompanied by ANKA-IT from an early stage of the design phase, the factory acceptance tests of components and systems at manufacturer site and the final tests at ANKA site.

Therefore ANKA-IT proposes to the beam line designer a catalogue of up to date ANKA-proven specifications for hardware components and control software environment.

The suggestion defines objects of hardware with a range of preferred attributes which describe the components manufacturer independently. This gives a flexible response to upgrade obsolete components over years of operation or setup new systems equipped with not 'ANKA-standard' compatible components.

On the software site versatility in intermediate layer, concepts enables the embedding of OEM controllers in the ANKA control system without giving preference to the operating system they were primarily designed for.

A design guideline was created by ANKA-IT to minimize the effort to select communication software and define interface specifications. The preferences for key elements are described in the chapters below

ANKA DESIGN GUIDELINES

- The manufacturer/supplier should clearly state the proposed scope of supply for the control system and associated electronics.

- The beam line control software concept will be provided by ANKA-IT.
- Network Interfaces are preferred for beam lines but alternatively an ANKA-standard hardware Interface, s. Figure 3 is accepted.
- The motion beam line components, use VME-bus as well as various bus protocols working over Ethernet TCP/IP.
- For the system realisation, a Tango-interface is preferred, in case of not availability spec*[2] can be used. The libraries and component software drivers for setting up motion control should be documented and supplied to ANKA-IT.

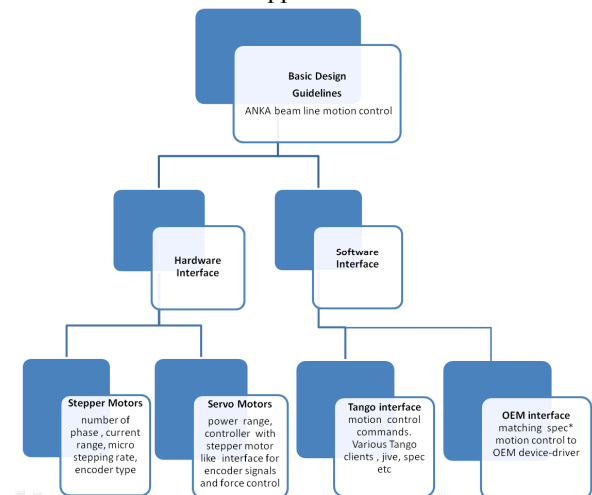


Figure 1: Overview hard- and software design guidelines for ANKA-motion control.

Since ANKA is migrating to Tango [3] for beam line control the communication between spec* and WinCCOA*[4] is fully Tango-based and all new implemented devices are controlled via Tango device servers.

DESIGN REPORT

A Tango interface and a set of geometry parameters for setup of the motor configuration should be supplied by the manufacturer in his design report. During the factory acceptance test these attributes and parameters are verified by test. The Tango interface is defined by the five device attributes: position, velocity, offset, limit switch negative direction, limit switch positive direction. The motor device is addressed by a set of specific commands given in Table 1.

*Trademark

Table 1: ANKA Tango device control commands

<i>Argum. Input</i>	<i>Argum. Output</i>	<i>Tango Name</i>	<i>Description</i>
DEV_VOID	DEV_VOID	Init	initialize motor
DEV_VOID	DEV_VOID	Forward	move motor + to limit switch
DEV_VOID	DEV_VOID	Backward	move motor - to limit switch
DEV_VOID	DEV_VOID	MotorON	init motor driver
DEV_VOID	DEV_VOID	MotorOFF	switch off motor driver
DEV_DOUBL	DEV_VOID	DefinePosition	new position in encoder units(dial)
DEV_DOUBL	DEV_VOID	ComputerNew-Offset	'is' position user, user-dial deviation. is calculated
DEV_DOUBL	DEV_VOID	MoveRelPosition	move position in user coordinates
DEV_VOID	DEV_VOID	Stop	stop motor
DEV_VOID	DEV_VOID	InitializeReferencePosition	starts homing
DEV_VOID	CONST_DEV_STRING	status string: ON,AL,FLT,MOV, OFF,STBY,UNK	states string
DEV_VOID	DEV_STATE	status integer: 0,11,8,6,1,7,13	states integer

MOTION CONTROL TEST SYSTEM

Having collected all parameters from manufacturers design report a test system is configured. The test system includes an OMS-MAXnet* [5] controller with TCP/IP interface for PC-controlled motion.

This allows different options for communication with the beam line components under test (Spec* native, Tango client or user defined clients, communicating with Tango) using the basic control software concept defined in the ANKA design guidelines described.

The System is completed by a test stepper motor/limit-reference-switch/encoder combination with ANKA-standard pin out. The test motor is used for

- The verification of a component test software configuration prepared for Factory Acceptance Test (FAT).

- At ANKA it is used complementary for the verification of the installed motion control pinout. The motor then replaces a beam line hardware component.
- The correct setting of the coordinate system for the component motion.
- The choice of geometrical parameters like gear ratios, encoder ratios, acceleration/deceleration intervals, motor backlash and homing procedure.



Figure 2: Newest version of ANKA-compact motor test system with display and stand-alone operator pulse control, which is used for direct verification of component compatibility to ANKA-Interface pinout, s. Figure 3. It is completed by power supplies for limit switches, encoder, two booster stages and a test stepper motor with limit/reference stop buttons and rotary encoder.

At FAT, the test system is used for fine tuning the kinematics of not visible components which are later enclosed in a vacuum container. This is of special interest due to define anti-collision stop and position recovery procedures for sensible and expensive motion components like a Bragg-crystals of a monochromator.

HARDWARE FOR MOTION CONTROL

Some of the standard ANKA beam line control hardware was designed by ANKA-IT. So a four channel I/O Interface for beam line motion component sensors, which is used universally at ANKA beam lines.

- Signal inputs levels are guarded by opto-couplers.
- Power supply outputs are generated for different kinds of limit- and reference switches (mechanical or electronic open collector).
- LED state indicators for limit switches, reference switch/encoder input signals. The power supply can be adapted for ohmic loss and routed to the Beamline component motion sensors.

In addition a signal conditioning terminal box (SCTB) for beam line components was developed. This SCTB (one per motion axis) is supplied to the component manufacturer with a detailed terminal diagram of the ANKA-standard pinout and preinstalled at his site before FAT.

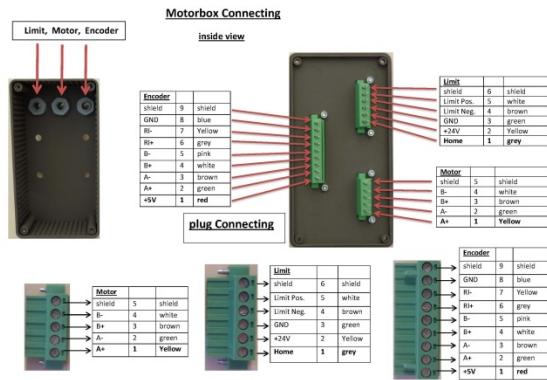


Figure 3: The SCTB box pin-out interface between the motion control signals generated by the beam line component and ANKA control system standard I/O. The signal levels can be reversed on the SCTB board due to positive or negative logic, referenced to Gnd/Vcc.

The advantage of ANKA motion control is based on the open loop concept. The actuators are moved in high speed mode without overshoot or oscillations to a measurement setpoint of interest.

The disadvantage will be that there is no direct automatic feedback to control absolute actuator positioning as with the ‘closed loop’ approach. When there is a controller power loss, the position of the motor steps may disagree with the encoder readings and a ‘homing’ procedure may be necessary. In case of a predefined deviation between the calculated, relative position (by motor pulse counts) and the monitored encoder value, a warning message is generated.

A choice is offered, the user can now agree to the spec* calculated position or when knowing the reason for discrepancy use the indicated encoder position or in case of doubt, test with the ‘check’ macros and initiate a homing procedure to regain absolute position.

SOFTWARE FOR MOTION CONTROL

At home a first setup of the motion parameters is given by the spec* motor configuration established at the Factory Acceptance Test.

The fine tuning of the motion behaviour is done using ANKA- software diagnostic tools

- A spec* macro set to test the functionality of the limit and reference switches of a component.
- A set to test absolute and relative positioning of the component under investigation
- A test for repetition accuracy doing various runs to gain statistics of motor-pulse calculated position and component final position indicated by encoder.
- Test of correct component motion in two directions and function of its limit switches.

- Test of encoder-gear ratio parameter set and motion repetition accuracy.
- Homing procedure with reference switch.
- Encoder homing with encoder reference mark.

CONCLUSION

At the beginning of 2014 more than ninety percent of ANKA motion control is based on PCs and low cost proven standard industry components:

PCI-VME*-bus, OMS-MAXv* controllers for response in the μ sec range and the OMS-MAXnet*-TCP/IP controller, making VME bus obsolete for applications with response time in the msec range.

Seven types of pin-compatible boosters with different electric current characteristic are driving a variety of stepper motor types in high resolution micro stepping mode and a fast open loop configuration.

With the introduction of OMS-MAXnet TCP/IP capable controllers, VME-based hardware communication for motion control will be on the decline at ANKA beam lines.

The last ten percent of ANKA motion control are closed loop controlled OEM systems with power in the kW range. This are mostly servo motors for heavy load applications like undulator gap mechanics or double crystal monochromators.

To avoid nested controllers, servo motors at ANKA are used in connection with special matching controllers [6] which accept the same feedback sensor input as used for stepper motor based motion control.

REFERENCES

- [1] B.Borovic *et al* 2005 *J. Micromech.-Microeng.* “Open loop versus closed-loop control of MEMS devices: choices and issues” in Journal of Micro-mechanics and Micro-engineering Volume 15 Number 10
- [2] spec, certified scientific software, <http://www.certif.com/>
- [3] Tango, Source forge, <http://www.tango-controls.org/>
- [3] Tango, Source forge, <http://www.tango-controls.org/>
- [4] WinCCOA, <http://w3.siemens.com/mcmsg/human-machine-interface/en/visualization-software/si-matic-wincc-openarchitecture/Pages/Default.aspx>
- [5] MAXnet manual, <http://www.omsmotion.com/>
- [6] Danaher motion, “S700 Digital Servo Amplifier S701...S724” Product Manual Translation of the original manual edition 12/2008. Valid for Hardware Revision

FPGA UTILIZATION IN THE ACCELERATOR INTERLOCK SYSTEM (ABOUT THE MPS DEVELOPMENT IN THE LIPAc)

K. Nishiyama*, J. Knaster, A. Marqueta, Y. Okumura,
IFMIF/EVEDA Project Team, JAEA, Aomori, Japan
T. Narita, H. Sakaki, H. Takahashi, JAEA, Aomori, Japan
T. Kojima, Nihon Advanced Technology, Aomori, Japan
R. Gobin, CEA, Gif-sur-Yvette, France

Abstract

IFMIF (International Fusion Material Irradiation Facility) will generate 14 MeV neutron flux for qualification and characterization of suitable structural materials of plasma exposed equipment of fusion power plants. IFMIF is an indispensable facility in the fusion roadmaps since provide neutrons with the similar characteristics as those generated in the DT fusion reactions of next steps after ITER. IFMIF is presently in its EVEDA (Engineering Validation and Engineering Design Activities) phase.

As part of IFMIF Validation Activities, LIPAc (Linear IFMIF Prototype Accelerator), designed and constructed mainly in European labs (CIEMAT, CEA, INFN and SCK CEN) with participation of JAEA, is currently under installation at Rokkasho (Japan). LIPAc will accelerate a 125mA CW and 9MeV deuteron beam for a total beam average power of 1.125MW. The Machine Protection System (MPS) of LIPAc provides the essential interlock function of stopping the beam in case of anomalous beam loss or other hazardous situations, particularly critical for investment protection reasons in high power accelerators.

High speed processing is indispensable to adequately achieve the MPS main goal. This high speed processing of the signals, distributed alongside the accelerator facility, is based on FPGA technology. This paper describes the basis of FPGA use in the accelerator interlock system through the development of LIPAc's MPS.

INTRODUCTION

LIPAc is a prototype accelerator that will reach a beam average power of 1.125 MW with deuterons in CW at 125 mA and 9 MeV. It will validate the accelerators of IFMIF [1] (125 mA in CW at 40 MeV) by demonstrating that the space charge issues can be overcome at its lowest 1st energy superconducting accelerator stage (the 40 MeV will be achieved in three additional SC stages at 14.5, 26 and 40). The involved high power and beam nature entails investment protection arguments and radiation safety aspects. The control system for LIPAc is responsible of its safety functions, the control and monitoring functions to realize these tasks efficiently target the minimization of activation induced by beam losses (driven by 'hands-on' maintenance principles) and potential hazard to the investment. In light of these requirements, the control system of LIPAc is broken down as follows: 1) Central Control System (CCS), 2) Local Area Network (LAN), 3) Personnel Protection System (PPS) to avoid unnecessary exposure to radiation, 4) Machine Protection System (MPS) for the accelerator subsystems and the facility,

5) Timing System (TS), 6) Local Control System (LCS) for the accelerator subsystems.

The remote operation for LIPAc is performed by CCS, LAN and the different LCS. The high level data (using EPICS) is transmitted by using LAN (Ethernet). All subsystems with synchronization for pulse operation are realized by using TS signals. The radiological safety for the personnel is established by PPS. And the beam inhibit (fast and slow) is realized by MPS in order to protect the accelerator and its components. This paper describes the outline and development status of MPS [2].

OUTLINE OF MPS

LIPAc produces a powerful CW deuteron beam [3] with high average beam current of 125 mA. The 9 MeV deuteron beam with its MW range beam power will be absorbed on a Cu beam dump water cooled. If an excessive beam loss event happens in an undesired manner on an accelerator component, the power would potentially cause a fatal damage. Additionally, the high inelastic cross sections of deuteron would lead, in case of excessive beam losses, to an increase of neutron and gamma induced dose rates in the radiation controlled area, with a potential dramatic impact of the targeted 'hands-on maintenance' approach. The Machine Protection System (MPS) is defined as the safety system against the accelerator troubles for the protection of the investment. The MPS has interfaces with other LIPAc subsystems, including the PPS (Personnel Protection System) Accelerator subsystems. It comprises the beam stop interlock signal from each accelerator subsystems, each of which presents interfaces with the MPS, PPS, EPICS, and emergency stop logic of subsystem. MPS realizes the beam rapid stop to minimize the effects on the beam pipe by beam loss. The target time of MPS signal transfer, which is the time taken from "MPS unit receives the interlock signal from accelerator subsystem" to "MPS sends the beam stop signal to the injector", is less than 10 μ s. To achieve this fast response time, FPGA technology has been chosen.

The backbone of MPS for LIPAc is the already consolidated successful MPS unit used at J-PARC Linac. In the case of MPS for J-PARC Linac, the "Beam rapid stop" is achieved within 5 μ s after "MPS unit receives the interlock signal from accelerator subsystem" with high reliability experienced [4].

Therefore, the use of J-PARC Linac's knowhow as the basis of LIPAc's MPS unit is suitable for the interface between MPS and accelerator subsystem; MPS itself will realize the logic for beam stop and beam restart.

CONFIGURATION OF MPS

The MPS consists of hardwired units (MPS unit, FPGA based for the fast logic processing) and PLCs (for the communications through EPICS to the CCS), as shown in Figure 1. The Interlock signal can be any relevant status signals coming from the subsystems (vacuum, temperature, cooling, RF, LVPS, valves, slits, BLOMs...).

The means of action are stopping the beam. Each MPS unit is connected by hardwire (metal or optical cable). An interlock signal received by an MPS unit is sent to the unit for Injector within the 10 μ s specified speed. Next, the unit for Injector sends the “beam stop signal” to Injector, and Injector will stop injecting the beam. The PLC connected with the MPS unit does not participate in stopping the beam process; the PLC is independent from MPS logic, so that it is only status monitors and reset the interlock latch of the MPS units. The MPS logic is separated from the PLC. The interlock logic function of MPS is realized using FPGA, because high speed signal processing is necessary. Since I/F with the external interlock signal of MPS correspond to transmission noise, it is isolated by the photo-coupler. Furthermore, the digital filter in FPGA removes noises.

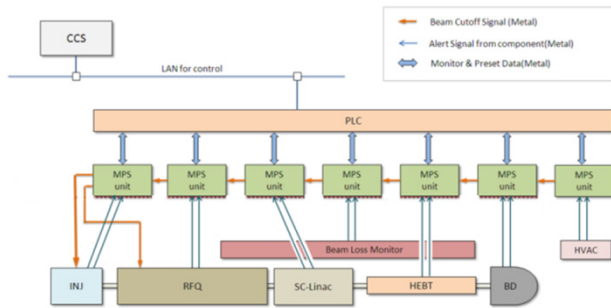


Figure 1: Principle of MPS.

MPS Unit and Interlock Module

The MPS unit is the basic structure, where modules and a power-supply module are united. Different modules are installed for each type of interlock signals. The unit can install 5 modules at maximum. There are interface terminals at the back panel. Figure 2 shows how the MPS modules are functionally linked together, arranged by type of signal. (slow/fast stop, etc) The modules are installed for each of the interlock signals. In turn, Figure 3 shows the functional configuration of the standard module of MPS. A standard module has the function to summarize and transfer the interlock signals. The standard module consists of a FPGA with some devices of interface to external signals. The communication between one MPS unit and another uses differential signal transmission protocol, to maximize the reliability of the link and increase the maximum available distance. The FPGA is rapid speed processing of summarized interlock signals. There are some interlock signal types from a subsystem; one is sent to the MPS unit directly after a fast response circuit, and the other is sent to the MPS unit after a slow response circuit. An interlock Beam Inhibition (BI) signal made by fast

response circuit is defined as “FBI signal” because the signal is sent to the MPS unit (and transmitted through it) in the tens of μ s order. In turn, a similar signal made by slow response circuit in the subsystem, is defined as “SBI signal” because the signal is sent to the MPS unit (and transmitted through it) in the hundreds μ s order. The standard module of MPS corresponds to these two types of interlock signals (FBI and SBI), and presents two kinds (fast, slow) of interlock I/F and mode change by jumper settings.

The interlock logic function of MPS is realized using FPGA, because high speed signal processing is necessary. Since I/F with the external interlock signal of MPS corresponds to noise transmission, it is isolated by the photo-coupler. Furthermore, the digital filter in FPGA removes noise.

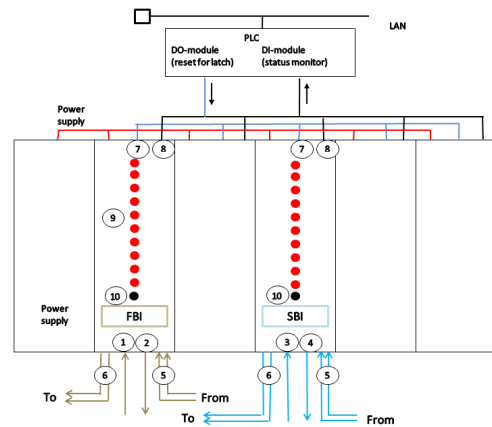


Figure 2: Functional arrangement of the MPS.

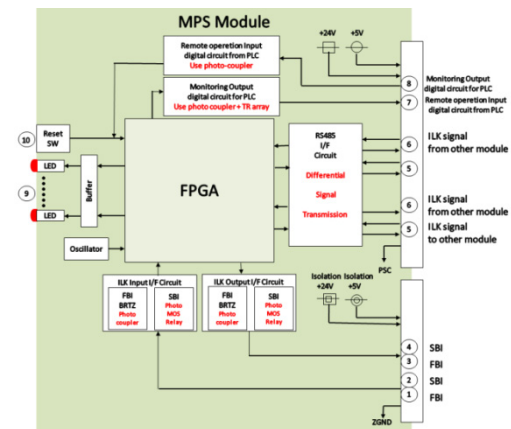


Figure 3: Functional Configuration of standard module of MPS.

DEVELOPMENT STATUS

Figure 4 shows the over view of the stopping beam methods on the MPS. The MPS stops the beam at its origin by the interception of RF (2.45 GHz) at the ECR ion source. The RF can be intercepted by three different approaches: 1) intercepting the timing gate, 2) intercepting the high voltage power supply, and 3) activating a crowbar. The MPS chooses the method depending on the

*Nishiyama.koichi@jaea.go.jp

interlock classification: 1) Beam Reset To Zero (BRTZ), 2) SBI, and 3) FBI.

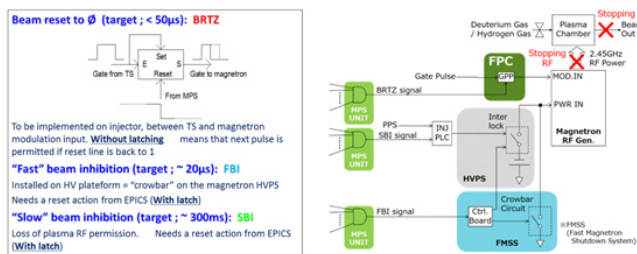


Figure 4: Over view of the stopping beam methods on the MPS.

The standard module of MPS is applied to FBI and SBI. For the BRTZ, it is necessary to add the beam stop logic on the standard module and its development is described next.

Beam Reset To Zero ($< 50 \mu s$)

This beam stopping method will be available only in pulsed operation mode and low duty cycle. The principle is that upon a MPS request, the beam pulse will be reset to zero as fast as possible by resetting the signal sent to the RF modulator (magnetron) of the ECR ion source, as shown in Figure 5. By this way, the duration of the selected beam pulse is shortened but the next pulses are again permitted as soon as the MPS signal line is back to normal state. The delay between the MPS request and the actual stop of the beam will be shorter than $50 \mu s$. The request for this beam stopping mode could come from the RF cavities control system, etc. Such a way of stopping the beam will become a key tool during the optimization phases especially when using automatic computer-assisted tuning or RF system conditioning procedures.

For an efficient BRTZ, it is required that the timing signal of the Injector is controlled by the summarized beam stop signal which is not latched. So it is unrealizable with the MPS unit. To overcome this difficulty, we have developed a specific beam stop logic (Fast Procedure Circuit (FPC) using FPGA as additional I/F) without changing the basic configuration of MPS, which already has been used successfully in J-PARC. The reason it twofold: 1) requirement of response time is very high speed with $50 \mu s$, and 2) additional function will be assumed. So FPGA was adapted such that high speed processing and change of logic is flexible.

Figure 5 shows the functional configuration of FPC and MPS and the functional chart.

Fast Procedure Circuit

The gate pulse for Magnetron Modulation is controlled by a BRTZ interlock signal from MPS. So we developed the FPC, as a functional addition to MPS. A gate pulse is controlled by a BRTZ interlock signal combining the standard module of MPS and FPC. FPC consists of two parts. There is an Interface (I/F) converter and a Gate Pulse Processor (GPP). An I/F converter cooperates with a MPS standard module, and performs signal processing

of the received interlock signals. A GPP controls a gate pulse signal by the BRTZ signal outputted from the I/F converter.

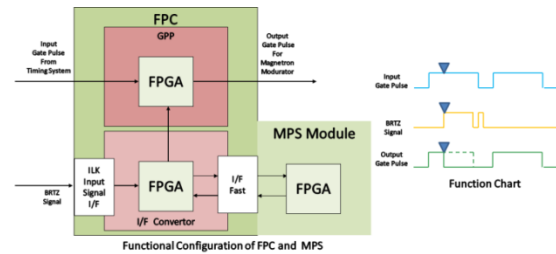


Figure 5: Functional configuration of FPC and MPS and Functional Chart.

Results of the Last Interface Tests of BRTZ

Tests between a MPS module and LIPAc injector were successfully carried out in CEA Saclay in November 2012. We measured the processing delay time, the response time of beam inhibit procedure for injector. The alert signal was measured with the help of an oscilloscope, and the response time was determined. The results obtained of these performance tests yielded for the response time of BRTZ an average of $40 \mu s$, faster than the target $50 \mu s$.

CONCLUSION

This paper describes the status of development of MPS using FPGA. The FPGA is an indispensable device of MPS that allows a rapid beam stop within $50 \mu s$ in compliance with the machine requirements for investment protection and 'hands-on' maintenance. The basic composition of MPS is not changed since the new interface which uses FPGA is added to MPS. It is not to be forgotten that we have already the beam stop time shorter than $30 \mu s$ by FBI, in compliance with the required specifications from beam dynamics. The quick restart after a beam stop has also been taken into consideration. There is still margin for future enhancements towards a faster process to beam restart using by FPGA.

REFERENCES

- [1] J. Knaster et al., "IFMIF, a fusion relevant neutron source for material irradiation current status", J. Nucl. Mat. 453 (2014) 115–119.
- [2] H. Takahashi et al., "Overview of the control system for the IFMIF/EVEDA Accelerator" Proceedings of the 6th Annual Meeting of Particle Accelerator Society of Japan, Tokai, Japan, August 2009.
- [3] J. Knaster et al., "Installation and Commissioning of the 1.1 MW deuteron prototype Linac of IFMIF", Proc. IPAC 2013, Shanghai, <http://jacow.org/>.
- [4] T. Suzuki et al., "Development of J-PARC LINAC/RCS MPS Sub System", Proceedings of the 5th Annual Meeting of Particle Accelerator Society of Japan, Hiroshima, Japan, August 2008.

NEW DEVELOPMENTS ON THE FAIR DATA MASTER

M. Kreider, R. Bär, D. Beck, W. Terpstra, GSI, Darmstadt, Germany
J. Davies, V. Grout, Glyndŵr University, Wrexham, United Kingdom

Abstract

During the last year, a small scale timing system has been built with a first version of the Data Master. In this paper, we will describe field test progress as well as new design concepts and implementation details of the new prototype to be tested with the CRYRING accelerator timing system. The message management layer has been introduced as a hardware acceleration module for the timely dispatch of control messages. It consists of a priority queue for outgoing messages, combined with a scheduler and network load balancing. This loosens the real-time constraints for the CPUs composing the control messages noticeably, making the control firmware very easy to construct and deterministic. It is further opening perspectives away from the current virtual machine-like implementation on to a specialized programming language for accelerator control. In addition, a streamlined and better fitting model for beam production chains and cycles has been devised for use in the data master firmware. The processing worst case execution time becomes completely calculable, enabling fixed time-slices for safe multiplexing of cycles in all of the CPUs.

OVERVIEW AND SYSTEM LAYOUT

As discussed in our previous papers [1, 2], the FAIR accelerator will be a highly complex system which needs a control system to match. This suggests a design that supports high performance, flexibility and deterministic command generation and distribution. While all machine commands for beam production will be calculated from physics data ahead of time, all final scheduling and deterministic delivery is the responsibility of the Data Master (DM). The DM itself is a hybrid of an industrial PC and a Field Programmable Gate Array (FPGA) based embedded real-time system with hardware acceleration modules. We will now discuss the system layout of the current implementation and the inner workings of the sub-modules in greater detail.

CONTROL DATA

Structure

The control data received by the DM broadly resembles a flowchart for beam production. It consists of $2..I$ alternative beam production scenarios, called plans. Each plan has $1..J$ event chains in it. Chains are the basic building blocks. They each contain $0..K$ command messages. They also come with the means for simple control structures. The input format is currently XML based and converted to a binary format for the embedded system.

Time

From the start of a plan, all times are relative offsets. Each chain has a given duration, and chain start times are calculated by adding up previous durations. The only exception to the rule is a conditional wait. Here, the start time of the next chain is set to time the condition was fulfilled, plus a fixed offset. An absolute execution time is calculated for each command message at the moment it is dispatched. This is done by adding the message offsets to its chain's start time.

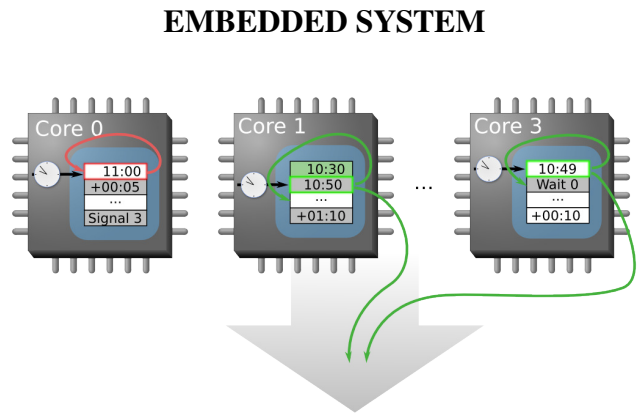


Figure 1: Scheduling commands in the Soft-CPU Cluster.

Layout and Firmware

The LM32 (Lattice Mico 32 Micro-controller) is a 32 Bit RISC processor for use in Field Programmable Gate Arrays (FPGAs) and Application Specified Integrated Circuits (ASICs) and a good choice for a control system [3]. Timer Interrupts were unsuitable for dispatch due messages, because saving and restoring all 32 registers takes considerable time and not all instructions have the same execution time. Furthermore, it would have meant the processors being idle most of the time. In order to build a deterministic system with a low reaction time, a polling approach with fixed time slices was used and multiple Soft CPUs were instantiated to deal with different parts of beam production in parallel, aided by hardware acceleration cores. Each of the LM32 runs very simple firmware with three responsibilities for each iteration. The first is synchronization, meaning checking conditions or signalling another process(or). Next comes processing the current chain. This means sending a due command message to the priority queue. The third is to check the command interface for external instructions from the control system. All worst case execution times are completely deterministic. The only exception would be dispatch, because the network interface is a shared resource for all cores. The current testbed only features one thread

per LM32 as yet. Figure 1 shows how the control data is evaluated in each iteration by the CPU cluster.

DEDICATED HARDWARE

This leaves the problem of arbitration and gathering these messages. They must also be ordered by urgency before adding them to network packets because different commands will have different requirements for control lag. Kicker Magnet control at FAIR will require a lag below 500 μ s while commands to the vacuum system can easily be implemented with a margin of seconds. Last but not least the payload/packet-size ratio should be high in order to make the most of the limited bandwidth. An additional core with a priority queue was introduced to deal with these requirements, making the whole system a heap on top of a calendar queue. The messages need to be wrapped in the EtherBone (EB) [4] network protocol when adding them to a network packet. In our earlier prototype, we used a full port of the x86 EB library. This is very impractical for various reasons and a wasteful solution in terms of RAM. Instead, an EB Master was implemented as a hardware core, providing a simple and fast bridge from the local Wishbone (WB) [5] bus over the network interface to the remote WB bus of the controlled endpoints.

Priority Queue

All command messages consist of device parameters and their execution timestamp. The priority queue uses the timestamp part to sort the messages in order of urgency, device parameters are payload and will reside in an extra RAM. There are several ways to implement priority queues, depending on the focus. In our case, we aim for minimal execution time of the get-Min and extract-Min operations, which returns the minimum key element.

In order to get a better performance than sequential re-ordering in software solutions can provide, multiple parallel accesses to the RAM are necessary. The underlying constraint for efficient sorting is the ability to read both child nodes at the same time and write the falling node to the former parent. FPGAs natively only provide dual port memory (DPRAM). An ASIC design can have more read ports to the same content without any problem. If this is to be emulated in an FPGA, memory and routing costs increase considerably.

RAM-based Heap At least two independent read ports are required on the RAM holding the sorting keys, so both left and right child node can be read at the same time. With the moving element in a register, it is possible to compare parent and both children in a single cycle. For reordering, an independent write port is required, so the minimum is a 3 port RAM (2 read, 1 write). Many manufacturers provide macro cores which emulate a multi-port RAM with 2 read and 2 write ports. However, the synthesizer can only achieve this by replication. So instead, it is more efficient to exploit

the fact that the read ports never access the same child node and keep left and right children in two different DPRAMs.

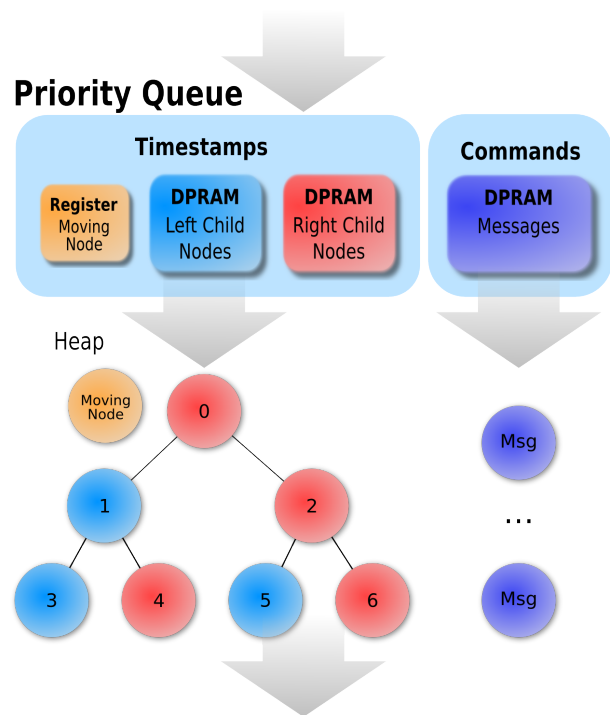


Figure 2: Heap-based priority queue in FPGA DPRAMs.

As Figure 2 shows, no redundant copy of the keys is necessary, and the arbitration logic is mostly Multiplexers and Write enable signals controlled by the least significant bit of the node index. The path taken through the heap is a sequence of indices. Placed in shift register, it can be used as addresses to order the message payload with constant delay to the key sorting. This simple design can implement *INSERT* and *REMOVE* Operations in $O(\log(n))$ time, with one comparison/move per clock cycle per heap level plus two cycles delay and minimal memory cost. But there is still much room for improvement.

Optimizing the Solution Firstly, *REMOVE* operations can only be pipelined if all heap levels are accessible in parallel [6]. Secondly, because *REMOVE* and *INSERT* normally work in different directions, pipelining of mixed operations is prevented. At the cost of fan-out and additional arbitration logic, one can reformulate the *INSERT* operation to work from top to bottom as well. In addition, two DPRAMs, one for left children, one for right, are necessary per level so each level is a pipeline stage that can move one element and all operations must work from top to bottom.

Both solutions combined are likely to be the most high-performance implementation of this design, managing one *INSERT* or *REMOVE* per two or three clock cycles, depending on desired design frequency. This is, however, slightly overpowered for our scenario for two reasons. The first lies in the layout of memory blocks in modern FPGAs, which are too big for the top heap levels and would result in wasted

memory [7]. The second reason is possible maximum speed. As we want a low system reaction time and elements in the heap are committed to be sent, it is not sensible to provide more than two EB packets (1500 Bytes - header) [4] of command messages. This makes the max. heap elements $n = \frac{2(1500B-8B)}{(8+2)4B} = 74.6$ and max. heap depth would be $\log_2(n) = 7$. For our design, this equals a min. dequeuing time of $T_{d\ min} = n + 2 = 9$ clock cycles. Taking a look at the timing constraints for queuing and dequeuing, we have relaxed conditions on the input side and leave a larger margin for packet processing. On the output side, we want the possibility to dequeue with line speed and catch up after delays. In our scenario, the 64 bit wide keys to be sorted come with a payload field of 192 bits, which equals 8 words at 32 bit. Absolute minimum output would be line speed of the Giga-bit Ethernet interface, which is $8\ bit/125\ MHz \rightarrow 256\ ns$ dequeue, bus speed is $32\ bit/62.5\ MHz \rightarrow 128\ ns$ dequeue $\rightarrow 8$ Clock cycles. So a dequeuing time of $8 \leq T_d < 16$ clock cycles already works well with a heap depth of 6 to 13, making the simpler implementation a valid approach for our timing requirements.

EtherBone Master

The EtherBone Master (EBM) core translates WB bus operations into the EB network protocol [4]. The main obstacle was a basic property of WB, which is it being a cycle based bus. This means that all operations have to be completed/acknowledged before a cycle can be finished. Over the network, this would lead to very high lag, stalling a local bus device, with the added risk of packet loss and therefore unacknowledged operations stalling the bus indefinitely. Our solution was to design a core which presents a simple interface to the user and fully conforms to the wishbone standard [5] and avoids the risk of freezing the local bus.

EBM Design It consists of two distinct WB slaves. The first is used for configuration, such as source and destination addresses for the network layer and additional information needed by EB [4]. The second slave acts as a write-only FIFO Buffer for WB operations to be sent. This way, the EBM can acknowledge all incoming data immediately, since it is not waiting for remote data. All replies go to the local EB Slave core. This now leads to several problems, first and foremost that directly using address and value of local WB operations is not possible. The reason is that the EB Master is itself a WB slave, occupying a certain address. The high address bits depend on its own position in the crossbar hierarchy and might be different from those of the remote target device. Another two bits are needed because control and data interface as well as read and write operations have to be distinguished. Our solution is therefore to replace all high address bits by the content of a control register which must be set according to the remote target address.

In the case of a WB write operation, address and value have the normal meaning except for the high address bits. A read operation is different. It is turned into an EB write on the destination platform, writing back the requested values

to the source. For this purpose, the address is the location to be read on the destination, but the value is the location on the source's bus where the return value is to be written. A flush command to the control register delivers the content to the network interface.

Fitness for DM Because command messages will always follow the same 8-word-write scheme, it is true that meta data is known in advance and a faster and much simpler core would have sufficed for the DM. The fixed format still makes encoding command messages deterministic. The EBM therefore turned out to be a good fit with added flexibility, obviating the need for a DM specific solution. It is also a powerful component, suitable for generic remote WB access.

The introduction of a generic representation for accelerator control in form of an XML string has already proven a great help for rapid development and generating test data. It is also invaluable for debugging. With the addition of hardware modules for sorting and sending command messages, most sources for non-determinism could be removed from the DM design. Both modules have been tested with the new data format on a quad-core embedded system in a lab environment and tests to generate timed pulses on a timing endpoint were satisfactory.

The DM has yet to be fully integrated with the FAIR control system and further to prove its ability to control the real CRYRING accelerator by early 2015.

REFERENCES

- [1] M. Kreider, R. Bär, D. Beck, J. Davies, and V. Grout. The fair timing master: A discussion of performance requirements and architectures for a high-precision timing system. In *Proc. of the International Conference on Internet Technology and Application ITA*, September 2013.
- [2] R. Bär, T. Fleck, M. Kreider, and S. Mauro. The timing master for the fair accelerator facility. In *Proc. of the International Conference on Accelerator and Large Experimental Physics Control Systems ICALEPCS*, pages 642–645, October 2011.
- [3] W. Terpstra. The case for soft-cpus in accelerator control systems. In *Proc. of the International Conference on Accelerator and Large Experimental Physics Control Systems ICALEPCS*, pages 642–645, October 2011.
- [4] M. Kreider, R. Bär, D. Beck, W. Terpstra, J. Davies, V. Grout, J. Lewis, J. Serrano, and T. Wlostowski. Open borders for system-on-a-chip buses: A wire format for connecting large physics controls. *Phys. Rev. ST Accel. Beams*, 15:082801, Aug 2012.
- [5] Wishbone b4 wishbone system-on-chip (soc) interconnection architecture for portable ip cores. Technical report, OpenCores, 2010.
- [6] Wojciech M. Zabołotny. Dual port memory based heapsort implementation for fpga. volume 8008, pages 80080E–80080E–9, 2011.
- [7] Embedded memory blocks in arria v devices. Technical report, Altera Corporation, 2014.

FIRST IDEA ON BUNCH TO BUCKET TRANSFER FOR FAIR

J. Bai^{1,2,*}, D. Beck¹, R. Bär¹, D. Ondreka¹, T. Ferrand^{1,3}, M. Kreider¹, C. Prados¹, S. Rauch¹,
W. Terpstra¹, M. Zweig¹

¹ GSI Helmholtzzentrum für Schwerionenforschung, Darmstadt, Germany

² IAP, Goethe University Frankfurt am Main, Frankfurt, Germany

³ Technical University Darmstadt, Darmstadt, Germany

Abstract

The FAIR facility makes use of the General Machine Timing (GMT) system and the Bunch phase Timing System (BuTiS) to realize the synchronization of two machines. In order to realize the bunch to bucket transfer, firstly, the source machine detunes slightly its RF frequency at its RF flattop. Secondly, the source and target machines exchange packets over the timing network shortly before the transfer and make use of the RF frequency-beat method to achieve the synchronization between two machines with accuracy better than 1° . The data of the packet includes RF frequency, timestamp of the zero-crossing point of the RF signal, harmonic number and bunch/bucket position. Finally, both machines have all information of each other and can calculate the coarse window and create announce signals for triggering kickers.

INTRODUCTION

The bunch to bucket transfer means that one bunch of particles, circulating inside the source machine, must be transferred in the center of a precise bucket and on the desired orbit of the target machine. It is realized by the General Machine Timing (GMT) system [1] and the Bunch phase Timing System (BuTiS) [2].

The main task of the GMT system is the time synchronization of more than 2000 Timing Receivers (TR) with nanosecond precision, distribution of timing events and subsequent generation of real-time actions by the TRs of the timing system located at the FAIR accelerator complex. The timing system is based on the White Rabbit (WR) network, which achieves the time synchronization by adjusting the clock phase (125 MHz carrier) and the time offset (Coordinated Universal Time – UTC) of all network TRs to that of a common grandmaster clock [3]. For the synchronization of radio-frequency (RF) components, the timing system is complemented and linked to the BuTiS. The BuTiS is a campus wide clock synchronization and distribution system. It generates an ident impulse clock at a rate of 10 μ s, a 10 MHz sinewave reference clock and a 200 MHz sinewave clock [4].

After a bunch of particles is accelerated to the top energy, the RF flattop, it must be extracted from the source machine to be injected in the centre of a bucket of the target machine without phase and energy error, e.g. Four batches of U^{28+} , each batch has two bunches ($h = 2$), at 200 MeV/u of SIS18 will be injected into eight out of ten buckets of SIS100 [5] (see Fig. 1). This paper explains the process of the bunch to

bucket transfer. The first step is the frequency detune and the second step is the synchronization of two machines by the frequency-beat method.

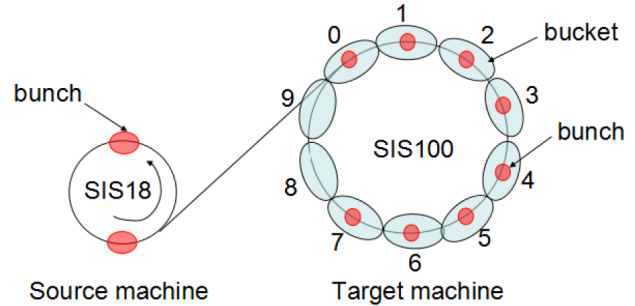


Figure 1: The bunch to bucket transfer of U^{28+} from SIS18 to SIS100.

BEAM-DYNAMICS VIEW OF THE FREQUENCY DETUNE

The first step for the bunch to bucket transfer is the RF frequency detune. In order to realize the frequency-beat between two machines, the RF frequency of the source machine has to be detuned. It means that the particles run at an average radius different by ΔR from the designed orbit R . To make the frequency detuning effective, the feedback loop (i.e. the radial loop [6]) must be turned off just before the frequency detuning begins. Accepting to decenter the orbit by 8 mm for SIS18 [7]:

$$\frac{\Delta R}{R} \approx 2.4 \times 10^{-4}, \quad (1)$$

the RF frequency detuning at the U^{28+} 200 MeV/u [7] extraction energy ($\gamma = 1.217$) is

$$\frac{\Delta f}{f} = -\frac{\gamma^2 - \gamma_t^2}{\gamma^2} \frac{\Delta R}{R} \approx 5 \times 10^{-3}, \quad (2)$$

where Δf is the frequency deviation for the frequency detuning, f is the RF frequency, $\gamma_t = 5.8$. The maximum RF frequency detuning is approximate to 7.5 kHz at 1.57 MHz for the U^{28+} .

The relative momentum shift is

$$\frac{\Delta p}{p} = \gamma_t^2 \times \frac{\Delta R}{R} \approx 8 \times 10^{-3}, \quad (3)$$

where p is the desired momentum of particle, Δp is the momentum shift caused by the frequency detune.

* J.Bai@gsi.de

The frequency detune process must be performed adiabatically. In order to perform this process adiabatically, the RF frequency detuning must be slowly enough for the longitudinal emittance to be preserved. The SIS18 synchrotron frequency Ω_s for the U^{28+} is:

$$\Omega_s^2 \approx 13.35 \times 10^{-6}. \quad (4)$$

The SIS18 synchrotron tune Q_s at the RF flattop is:

$$Q_s \approx 0.00075. \quad (5)$$

During the frequency detuning process, the SIS18 synchrotron frequency must satisfy the following relation [6]:

$$\frac{1}{\Omega_s^2(t)} \left| \frac{d\Omega_s(t)}{dt} \right| \ll 1. \quad (6)$$

However, the frequency detuning will cause the average radial excursion and relative momentum shift.

SYNCHRONIZATION OF TWO MACHINES

The second step for the bunch to bucket transfer is the synchronization of two machines using the frequency-beat method. For each machine, the TR of the timing system is coupled to its RF system. After receiving the timing event (e.g. "Synchronization begin") from the timing network, the TRs enable to timestamp the zero-crossing point of the RF signals locally with accuracy better than 1ns. Besides, the TR at the target machine measures the phase of the harmonic number first ($h=1$) of the RF signal, with which the source machine can create the announce signals locally. Then the TR of the target machine sends the packet to the source machine. The data of the packet includes the RF frequency, timestamp of the zero-crossing point, harmonic number and the phase of $h=1$. At the same time, the source machine sends the packet to the target machine, which includes the same information but the phase of $h=1$. Both machines have all information so that they are able to calculate the propagation of uncertainties of the zero-crossing point measurement, the coarse window. Within this window, the bunch of particles could be transferred to the target machine with a deviation less than 1° . The source machine makes use of the information of the phase of $h=1$ to produce a series of announce signals to choose its next RF rising edges, which coincides with $h=1$ of the target machine. With the help of the coarse window and the announce signals, both machines can trigger their kickers.

Frequency-beat Method

Here we assume that the source machine is SIS18 and the target machine is SIS100. The SIS18 RF frequency is f_{rf}^{SIS18} and the SIS100 RF frequency is f_{rf}^{SIS100} . Δf is the RF frequency detuning value of the SIS18.

The number of SIS100 revolution to realize the synchronization is n .

$$n = \frac{t_{100best} - t_{18best} - \frac{\Delta n}{f_{rf}^{SIS18} + \Delta f}}{\frac{1}{f_{rf}^{SIS18} + \Delta f} - \frac{1}{f_{rf}^{SIS100}}} \quad (7)$$

t_{syn} is the time cost for the synchronization.

$$t_{syn} = \frac{(f_{rf}^{SIS18} + \Delta f) \times t_{18best} - f_{rf}^{SIS100} \times t_{100best} + \Delta n}{(f_{rf}^{SIS18} + \Delta f) - f_{rf}^{SIS100}} \quad (8)$$

where t_{18best} and $t_{100best}$ are the timestamps of the zero-crossing point of the RF signals measured by TRs with uncertainty [8] of 1 ns ($\delta t=1ns$). Δn equals 1 when $t_{18best} < t_{100best}$ and equals 0 when $t_{18best} \geq t_{100best}$.

Coarse Window and Example

The coarse window is the result of the propagation of uncertainties of the zero-crossing point measurement.

$$\begin{aligned} A &= \frac{(f_{rf}^{SIS100})^2 + (f_{rf}^{SIS18} + \Delta f)^2}{\Delta f^2} \\ B &= \frac{2 \times [(f_{rf}^{SIS18} + \Delta f) \times (t_{18best} - t_{100best}) + \Delta n]^2}{\Delta f^4} \\ C &= \frac{2 \times (f_{rf}^{SIS18} + \Delta f) \times (t_{18best} - t_{100best})^2}{\Delta f^3} + \frac{2 \times \Delta n \times (t_{18best} - t_{100best})}{\Delta f^3} \\ D &= \frac{(t_{18best} - t_{100best})^2}{\Delta f^2} \end{aligned}$$

$$\delta t_{syn} = (A \times \delta t^2 + B \times \delta f^2 - C \times \delta f^2 + D \times \delta f^2)^{\frac{1}{2}} \quad (9)$$

where we assume that $f_{rf}^{SIS18}=f_{rf}^{SIS100}=1MHz$, $\Delta f=100Hz$, $\delta t=1ns$. Because the RF frequency has the long term stability, $\int \delta f dt=0$.

Based on these assumptions, the coarse window is $\pm 14.143 \mu s$ of the best estimation. The maximum time for the synchronization is 10 ms. So the accuracy within this coarse window is better than 1°

$$\frac{10 \text{ ms}}{360^\circ} \approx 27.7 \mu s \quad (10)$$

Test Setup

We use two MODEL DS345 Synthesized Function Generators [9] with the frequency accuracy of ± 5 ppm of the selected frequency to simulate RF signals from RF cavities of SIS18 and SIS100. Two FPGA-based cards are responsible for the time/phase measurement, information transmission and coarse window calculation (see Fig. 2).

Test Result

This setup theoretically simulates the synchronization of two machines, with accuracy better than 1° (see Fig. 3). It paves the way for the further FAIR bunch to bucket transfer.

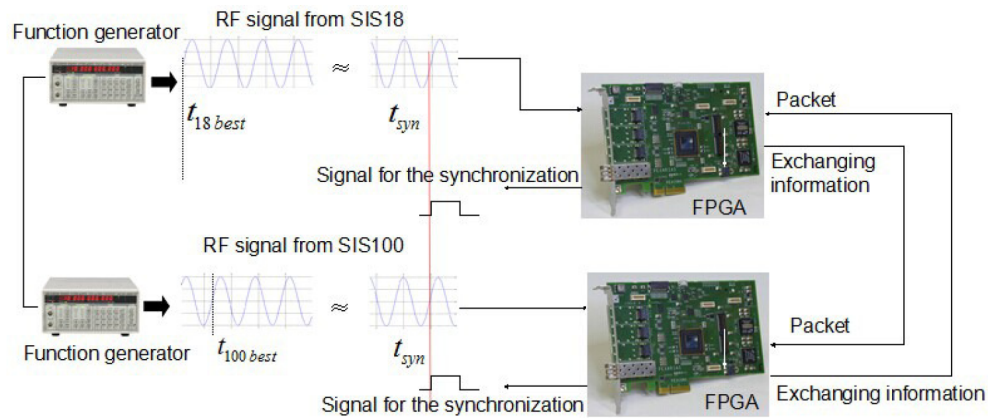


Figure 2: The test setup for the bunch to bucket transfer.

=====	
The timestamp of RF signal from SIS18 (accuracy to 1ns)	
Timestamp:	0x423c51cdc83
GMT:	1970-01-01 01:15:51.0.677369475
=====	
The timestamp of RF signal from SIS100 (accuracy to 1ns)	
Timestamp:	0x423c51cdc5
GMT:	1970-01-01 01:15:51.0.677370053
=====	
The number of the revolution of SIS18: 5780	
Synchronization time: 5.780000ms	
Best estimation timestamp for the synchronization: 0x423c5750ea3	
GMT for the synchronization: 1970-01-01 01:15:51.0.683149475	
=====	
Uncertainty of the coarse window: 14.143μs	
=====	
Period of synchronization is : 10000000ns = 10.000000ms	
Current timestamp: 0x423c79bf0d8 (1ns)	
GMT: 1970-01-01 01:15:51.0.719252184	
=====	

Figure 3: The test result of the bunch to bucket transfer.

SUMMARY AND IMPROVEMENT

This setup of the FAIR bunch to bucket transfer has been basically realized in the laboratory. It still has plenty of room for improvement. Improvements are:

- Compared with ± 1 ns uncertainty of the TR, the DSP system [10] from the department of Primary Beam RF at GSI reaches the uncertainty of $\pm 0.1^\circ$ of the measured phase, which corresponds to an accuracy of 50 ps of a RF frequency of 5.4 MHz. It will narrow the coarse window.
- The department of Primary Beam RF could provide RF signals of different harmonics of the target machine at the source machine instead of creating these signals locally, which is more precise and straightforward.
- The time consumption for the packet transfer over the current timing network is about 500 μs, which can be reduced to 200 μs without WR switches. We will connect TRs of two machines directly by the optical fiber.

ACKNOWLEDGEMENT

The author would like to thank Dietrich Beck, David Ondreka, Bär Ralph and other 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 Ferrand Thibault for his assistance in keeping contact with the Department Ring RF Systems. The grateful thanks are also extended to Professor Dr. Oliver Kester and Peter Moritz (†) for their support.

REFERENCES

- [1] D. Beck et al., *The New White Rabbit Based Timing System for the FAIR Facility*, FRIA01, PCaPAC'12, India, p. 242, (2012).
- [2] D. Beck, *Interface to BuTiS*, <https://www-acc.gsi.de/wiki/Timing/TimingSystemButisInterface>
- [3] D. Beck et al., *White Rabbit Technology as Basis for the FAIR Timing System* GSI Scientific Repor 2011, p. 333, (2011).
- [4] P. Moritz, *BuTiS - Development of a Bunchphase Timing System*, GSI Scientific Report 2006, p. 64, (2007).
- [5] D. Ondreka, *Settings Generation for the RF Systems in FAIR*, GSI/CERN Meeting (2014).
- [6] E. Ezura et al., *Beam-Dynamics View of RF Phase Adjustment for Synchronizing J-PARC RCS with MR or MLF*, KEK Internal 2008-1, June (2008).
- [7] H. Liebermann, D. Ondreka, *FAIR and GSI Reference Cycles for SIS18*, GSI, October 3, (2013).
- [8] John R. Taylor, *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements* University Science Books, Sausalito, CA, USA (1997).
- [9] Stanford Research Systems, *MODEL DS345 Synthesized Function Generator*.
- [10] H. Klingbeil, *Low-Level-RF (LLRF) Systems for FAIR Synchrotron and Storage Ring Cavities*, FAIR-Technikforum, November, (2010).

ADEI AND TANGO ARCHIVING SYSTEM – A CONVENIENT WAY TO ARCHIVE AND REPRESENT DATA

D. Haas, S. Chilingaryan, A. Kopmann, D. Ressmann, W. Mexner, KIT, Karlsruhe, Germany

Abstract

Tango offers an efficient and powerful archiving mechanism of Tango attributes in a MySQL database. The tool Mambo allows an easy configuration of all to be archived data. This approved archiving concept was successfully introduced to ANKA (Angströmquelle Karlsruhe). To provide an efficient and intuitive web-based interface instead of complex database queries, the TANGO Archiving System was integrated into the “Advanced Data Extraction Infrastructure ADEI”. ADEI is intended to manage data of distributed heterogeneous devices in large-scale physics experiments. ADEI contains internal preprocessing, data quality checks and an intuitive web interface, that guarantees fast access and visualization of huge data sets stored in the attached data sources like SQL databases or data files. ADEI and the Tango archiving system have been successfully tested at ANKA's imaging beamlines. It is intended to deploy the whole system at all ANKA beamlines.

INTRODUCTION

ANKA is a third generation synchrotron light source operated by the Karlsruhe Institute for Technology (KIT). ANKA is operating sixteen beamlines and three more are under construction.

The control system of the ANKA beamlines is based on Tango [1], which has been a convenient and reliable control system. Tango offers an archiving system [2] which allows logging all Tango-attributes. In 2014 this archiving system was evaluated at ANKA to log the data of an experiment.

All logged data of a beamline and the experiment should be presented and retrieved in a modern, state of the art web interface. This offers the user a convenient way to access the data. ADEI a web based interface for database query, developed by the Institute for Data Processing and Electronics (IPE) of KIT fulfils exactly these requirements. Using ADEI as a viewer respectively analysis tool and connecting it to the databases of the Tango archiving system creates a platform to track and to monitor the status of a beamline.

For testing the environment, the system was developed and implemented at two beamlines Topo-Tomo [3,4] and Image at ANKA.

ADEI

The “Advanced Data Extraction Infrastructure (ADEI)” has been developed to provide ad-hoc data exploration capabilities to a broad range of long-running physical experiments dealing with time series data [5]. Such experiments have varying characteristics and 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 1 is modular. New data sources can easily be included. The backend 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.

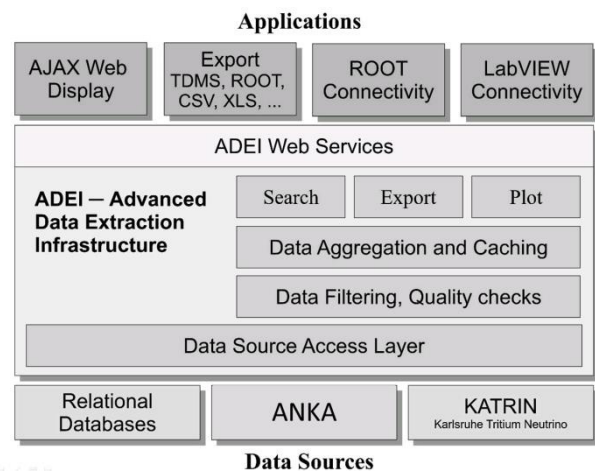


Figure 1: Architecture of Advanced Data Extraction Infrastructure ADEI. Data Source Access Layer unifies access to the time series stored in different formats. After data filtering and quality checks the data is aggregated and stored in intermediate caching databases. Access to the data is provided by the ADEI library and web services are used to communicate with client applications.

The backend 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 released 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 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.

ADEI provides several interfaces to access the managed data. The Export module provides direct access to the data in variety of formats implemented as plugins. The Search module allows to quickly find 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. This interface is used by ADEI front-end but also by a number of 3rd party applications.

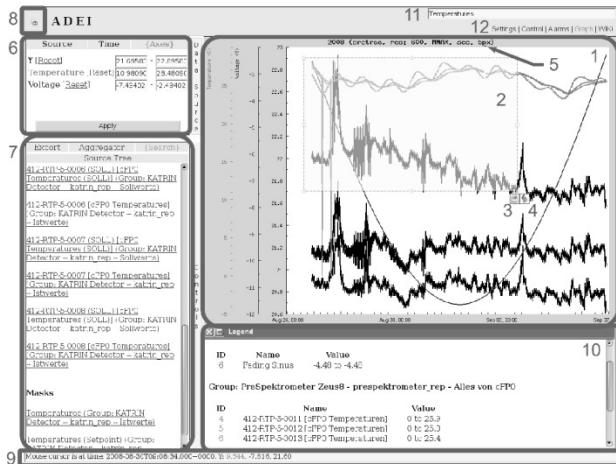


Figure 2: Screenshot of ADEI Web Front-end. The slow control data is rendered in the plot. The data outage is indicated using a small line on top of the plot (see 5). Legend contains description of displayed graphics. The selected part of plot may be zoomed or exported using buttons 3 and 4. Axes controls and results of search are located in the left sidebar.

The ADEI front-end facilitates fast and intuitive (Google maps-style) navigation as shown in Figure 2. 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 on the overview charts. User, then, can easily navigate to the interval of interest and zoom-in using 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 Instruments LabVIEW environment and Cern's ROOT data analysis suite.

SETTING UP THE WHOLE TANGO ARCHIVING SYSTEM AT ANKA

The ANKA archiving system is based on the approved Tango archiving and ADEI (see Fig. 3).

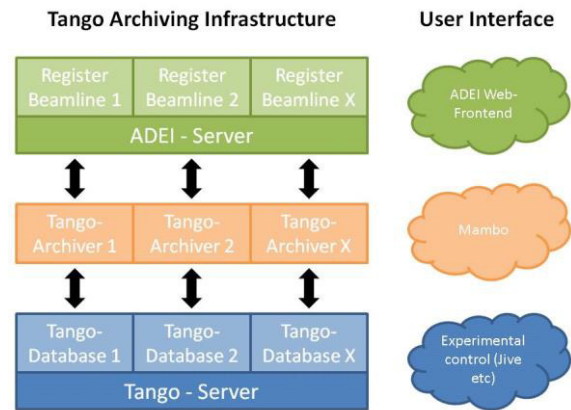


Figure 3: Tango Archiving Infrastructure and User Interface.

Every beamline at ANKA has its own Tango database which can be archived by one server, where the Tango archiving system is running. To configure the values and the attributes which have to be logged, Mambo [2], is used. Mambo is a user friendly graphical user interface (GUI), which offers an absolute well-structured workflow to set up the archiving by the beamline scientist.

To read out and view the logged data by the archiving system, ANKA was looking for a powerful tool representing it by a state of the art web interface.

To integrate Tango data, we have developed a new Tango reader which implements ADEI data access interface and exposes the data from the Tango database using it. Additionally, to make data navigation more convenient, we implemented the Tango device tree in the ADEI channel selection dialog. Now, the data from Tango archives can be interactively visualized using ADEI web interface, exported in multitude of formats supported by ADEI, and transparently accessed using ADEI web services. Other existing non-Tango data sources can be visualized side-by-side with the Tango data (see Fig. 4).

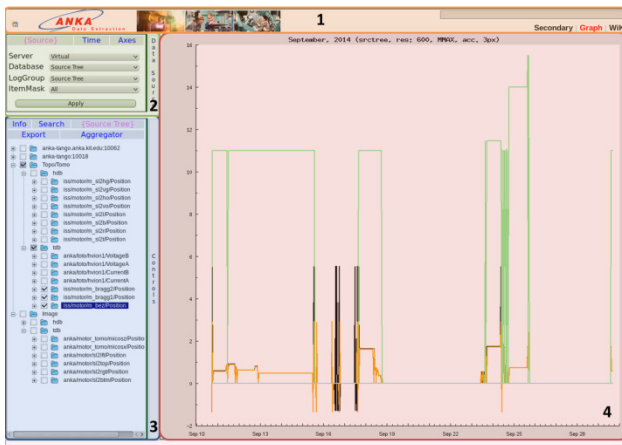


Figure 4: Screenshot of ADEI Web Front-end for the ANKA Tango Archiving System. The data from ANKA slow control system (Bragg axes at the Topo-Tomo beamline) is rendered in the plot. (1) main menu, (2) dropdown-menu for selecting server respectively beamline, (3) data selection, (4) plot of the selected data.

CONCLUSION & OUTLOOK

The whole system can be setup in fast and easy way with only a small amount of work. The concept has been proved to be absolute stable and reliable. The big advantage of the whole system is that it's connecting the Tango archiving system with a total user-friendly web-based user interface to read out the logged data. Due to the ADEI architecture the visualization of huge datasets is extremely fast.

After several months of testing phase it has been proved that the beamline and experiment status can be represent in a convenient and transparent way.

As it has been previously emphasized, the Tango archiving system connected to the web front-end ADEI is implemented at the Topo-Tomo as well as at the Image beamline.

The remaining challenge is now to implement it to all ANKA beamlines. This is scheduled for the beginning of 2015.

WHERE TO GET ADEI?

ADEI and the install instructions can be found under <http://adei.info>

REFERENCES

- [1] 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.
- [2] Tango archiving system, <http://www.tango-controls.org/tools/archiving-system-2/archivingsystem>

- [3] A. Rack, T. Weitkamp, S. Bauer Trabelsi, P. Modregger, A. Cecilia, T. dos Santos Rolo, T. Rack, D. Haas, R. Simon, R. Heldele, et al., "The micro-imaging station of the topotomo beamline at the ANKA synchrotron light source", *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 267(2011): p. 1978 - 2009.
- [4] 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*.
- [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

WEB BASED MACHINE STATUS DISPLAY FOR SIAM PHOTON SOURCE

N. Suradet, C. Thamtong, C. Preecha, S. Klinkhieo, P. Klysubun
SLRI, 111 University Avenue, Muang District, Nakhon Ratchasima, 30000, Thailand

Abstract

A new machine operation status broadcasting system has been developed for Siam Photon Source (SPS), a 1.2 GeV synchrotron light source in Thailand. The system is implemented using web-based interface, and broadcasts the information over the SPS website, mobile application, as well as local TV network within the SPS facility, allowing users as well as technical personnel to easily access a variety of information related to the machine via web browsers and other mediums. The new system also provides supporting message services for alarm, event notification, and other operational necessities. In this report, the design of web and mobile applications, which are based on HTML5, CSS3, and adopts PHP, AJAX, Bootstrap framework (for responsive design), jQuery, High charts JS, Twitter widget, and others, will be described. The details of the hardware and software configurations, users requirements and satisfactions, as well as suggestions on further improvements, will be presented.

This web-based system has two main disadvantages. First, it consumes quite a bit of the network bandwidth because the whole web page had to be constantly updated, and the size of the chart image was quite large. Secondly, the system cannot display the data in real-time. We found it necessary to develop a new system that is more robust, more responsive, and more accessible. The new system has to meet the following requirements:

- The core system is based on web technology.
- The web layout is able to present the contents clearly and accurately across multiple types of devices (PC, mobile phones, tablets, etc.) with diverse display resolutions.
- The data is constantly updated every 5 seconds, but the network traffic must be kept low.
- The beam current and beam lifetime chart is generated by the browser on the client side. The displayed data can be exported to a CSV file.
- The system is capable of broadcasting notification messages.

INTRODUCTION

The Siam Photon Source (SPS) is a synchrotron light source operated by Synchrotron Light Research Institute (SLRI), and is located in Nakhon Ratchasima, Thailand. The first light was achieved back in December 2001. At present, the machine is operating at 1.2 GeV in decay mode with a maximum electron beam current of 150 mA. Three insertion devices, a permanent magnet planar undulator, a hybrid multipole wiggler, and a superconducting magnet wavelength shifter, are currently in operation, providing synchrotron radiation from infrared to hard x-rays to synchrotron light users.

The original machine operation status broadcasting system was developed back in 2000, providing the operation status of the machine, for e.g. beam current, beam lifetime, beam energy, to users, who can access the provided information through the internal cable TV system within the facility. Each display channel receives the machine status data from a LabVIEW program located on a computer server. Since this system was available only for on-site users, another system was developed in 2006 to provide the machine status information via the internet. The fundamental language used to create this web-based system was basic static HTML. The displayed beam current and lifetime chart was captured from a NI LabVIEW window.

SOFTWARE ARCHITECTURE

The machine status data originates from a variety of sources. These sources/hardwares are interconnected via an assortment of interface standards (OPC, GPIB, RS-232, etc.). A data logging program written with LabVIEW and installed on an acquisition server is employed to continuously gather all the machine data and log them into a database. The logging interval is 5 seconds. Open source database MySQL [1] was chosen for our purpose. LabVIEW MySQL connector toolkit [2] allows LabVIEW to communicate with MySQL (version 4.1 or later) via the TCP/IP protocol. It is a part of the LAMP (Linux-Apache-MySQL-PHP) platform that has to be installed on the web server.

When the user opens the SPS machine status web page, the browser on the client side will make a request for the PHP webpage to the web server. The web server responds by sending HTML, JavaScript, and CSS scripts to the client for processing, so that the execution is performed by the client browser. We use AJAX (Asynchronous JavaScript And XML) [3,4] to help refresh the web page for updating the data. AJAX runs a background operation which extracts the data from the database in the XML/JSON data format every 5 seconds. It updates the data field of the web page without reloading the whole page, thereby substantially reducing the traffic demand on the network. Fig. 1 shows the architecture of the machine status broadcasting system.

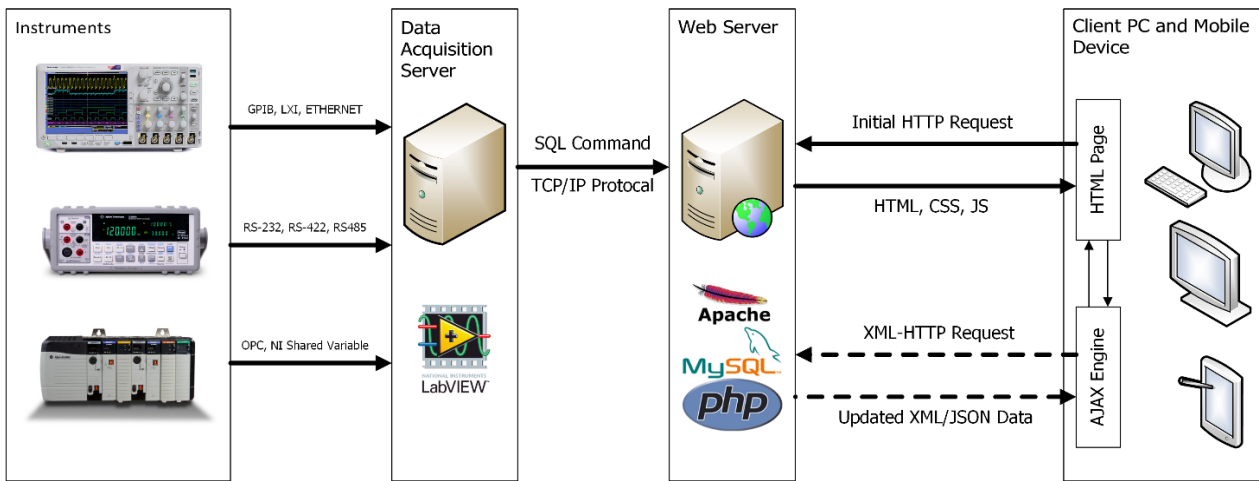


Figure 1: Machine status broadcasting system architecture.

DESIGN AND CODING

We employ various web-based technologies and frameworks to create the machine status webpage. These tools, which help make the code simpler and more efficient, are as follows.

Frontend Framework with Bootstrap

Regarding our aforementioned requirements that the new system has to be able to display the machine information correctly across all supported devices, and it must be relatively easy to develop, we employ the open source Bootstrap framework [5] to help manage the frontend. The Bootstrap package includes Scaffolding, Base CSS, Components, and JavaScript plug-ins. It handles display duty across all the devices with multiple resolutions, making the machine status webpage display-responsive, as shown in Fig. 2.

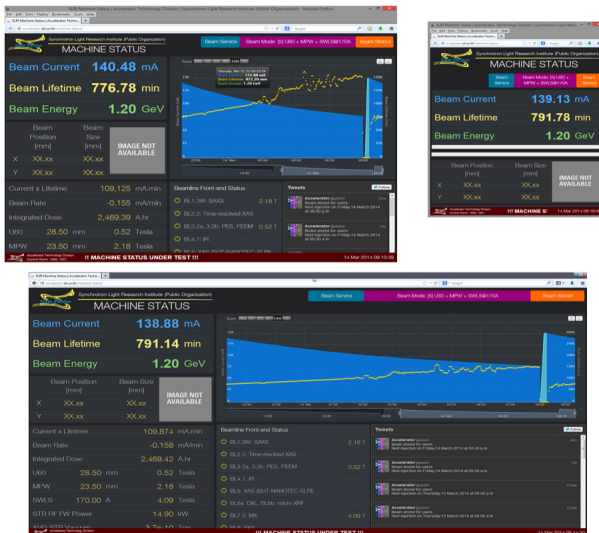


Figure 2: Machine status webpage at different screen resolutions. (Top left: 1280x1024, top right: 640x800, and bottom: 1920x1080)

Asynchronous Communication with AJAX

AJAX (Asynchronous JavaScript And XML) is a tool for creating fast and dynamic web pages. AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scene. This means that it is possible to update specific parts of a web page without reloading the whole page, significantly reducing traffic load placed on both the server and the network.

Real-time Graph with Highcharts JS

The SPS beam current and beam lifetime chart which shows a historical record over 24-hour period contains the recorded values of the beam current, beam lifetime, beam energy, and the associated timestamp. We use Highcharts JS [6], which is a JavaScript framework for creating charts. Its process is performed on the client side (browser). A data buffer is created and stored in the MySQL database along with the associated timestamp, which is installed on the web server.

The data is extracted from this database through a PHP query program. AJAX together with jQuery manage refreshing of the chart, which is a JSON data format, resulting in an auto-refresh real-time chart. The process is illustrated in Fig. 3.

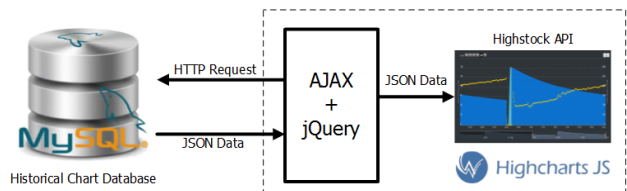


Figure 3: Software architecture of the auto-refresh real-time beam current and lifetime graphs.

Notification Broadcasting with Twitter

Incorporating a notification broadcasting into the system was deemed necessary from the beginning since it would be the main channel facilitating the communication from the control room to the users. For this we chose

Twitter [7], a social networking platform that allows one to send short messages (less than 140 characters in length) to followers. Twitter has a widget for displaying a message on the website and also delivering push notifications to a mobile device. The tool enables machine operators to send crucial machine status updates out instantaneously.

MOBILE APPLICATION

An iOS web application was written for iOS-based devices. The web app can be installed from the Safari web browser, and can be accessed later from the iOS home page, as shown in Figure 4(a). The program simply retrieves the machine data from the machine status webpage. When the condition that the user agent is a mobile device is detected, the system will display a specialized frontend framework containing the Slidebars [8] and a navigator menu, as shown in Figure 4(b). The machine status display remains on Bootstrap as mentioned earlier; however, the information is separated into multiple pages to fit the screen resolution of the mobile device, as shown in Figure 4(c).

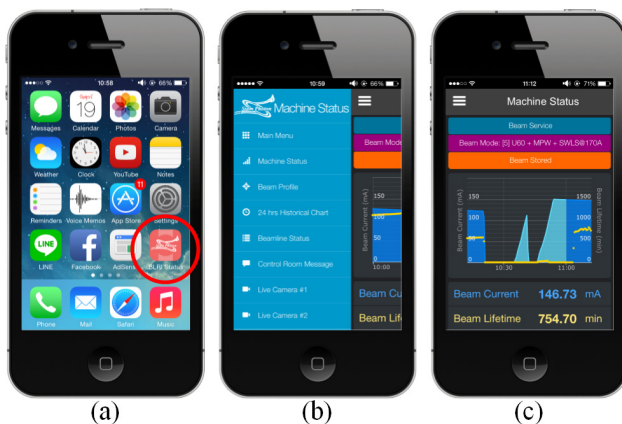


Figure 4: Mobile application: (a) SPS machine status app icon, (b) Slidebars navigator menu, and (c) SPS machine status on mobile device.

HARDWARE CONFIGURATION

Both the data acquisition server and the web server are based on two IBM System x3250 M4 servers having Intel Xeon E3-1270 3.4GHz (8MB L3cache), 8GB of system memory, 2xSATA drive of 250GB with RAID 1 (mirror) configuration, and gigabit ethernet. The operating systems are Windows 2008 Server and Ubuntu Linux, respectively.

CONCLUSION

The new SPS machine status broadcasting system has been in use for approximately 8 months since February 2014. The system is found to be robust, effective, and user-friendly. Future plan includes making it available for devices based on other platforms.

ACKNOWLEDGEMENT

The authors are thankful to all the members of Accelerator Technology Division and Technical Support teams for their support, constructive remarks, and feedbacks.

REFERENCES

- [1] MySQL website: <http://www.mysql.com>
- [2] LabVIEW TCP/IP Connector for MySQL Data base, <https://decibel.ni.com/content/docs/DOC-10453>
- [3] AJAX website: <http://www.w3schools.com/ajax>
- [4] Y. S. Cheng et al., "Upgrade the machine status broadcast system by PHP framework", THP113, Proceedings of ICALEPPCS2009, Kobe, Japan (2009), <http://jacow.org/>.
- [5] Bootstrap website: <http://getbootstrap.com>
- [6] High charts - Interactive JavaScript charts for your web pages, <http://www.highcharts.com>
- [7] Twitter website: <https://twitter.com>
- [8] Slidebars website: <http://plugins.adchsm.me/slidebars>

REDESIGN OF ALARM MONITORING SYSTEM APPLICATION BeamlineAlarminfoClient AT DESY

S.Aytac, DESY, Hamburg, Germany

Abstract

The alarm monitoring system ‘Beamline-AlarminfoClient’ is a very useful technical-service application at DESY, as it visually renders the locations of important alarms in some sections (e.g. fire or other emergencies). The aim of redesigning this application is to improve the software architecture and allow the easy integration of new observable areas including a new user interface design. This redesign also requires changes on server-side, where alarms are handled and the necessary alarm information is prepared for display. Currently, the client manages alarm data from 17 different servers. This number will increase dramatically in 2014 when new beam lines come into play. Thus creating templates to simplify the addition of new sections makes sense both for the server and client. The client and server are based on the Tine control system and make use of the Tine-Studio utilities, the Alarm Viewer and the Archive Viewer. This paper presents how the redesign is arranged in close collaboration with the customers.

INTRODUCTION

BeamlineAlarminfoClient (BAiC) is a visualization of emergency alarms in different areas. Currently it is used in the experimental halls PETRA III, FLASH and Photon-Science (PS) with 17 areas to be monitored.

The start of the first project was in 2004 and only used for monitoring FLASH area. Then PS alarms were added and at finally PETRA III. Since 2012 the FLASH extension project has been running and separated to FLASH-I and FLASH-II with independent FEL sources. In 2014 the PETRA III extension project began, with additional halls in the north of the storage ring and one in the east. The near future will give us ~30 areas to be monitored.

Aim of the Software Project

The functionality of the application is mostly defined by the customers and is regarded as an additional diagnostic for the technical-service personnel at DESY.

The followings alarms are monitored and displayed:

1. gas (concentration, magnetic- & exhaust valves)
2. fire
3. water
4. emergency call
5. emergency stop
6. common errors

The GUI is split into main and area views (outlined in Figure 1). If an area alarm is identified, a popup window of this area becomes visible, the active alarm is listed in

the area table, and the alarm location toggles on the area plan. When the alarm is cancelled the popup becomes invisible and the alarm is listed in the main view alarm table of the last 72 hours.

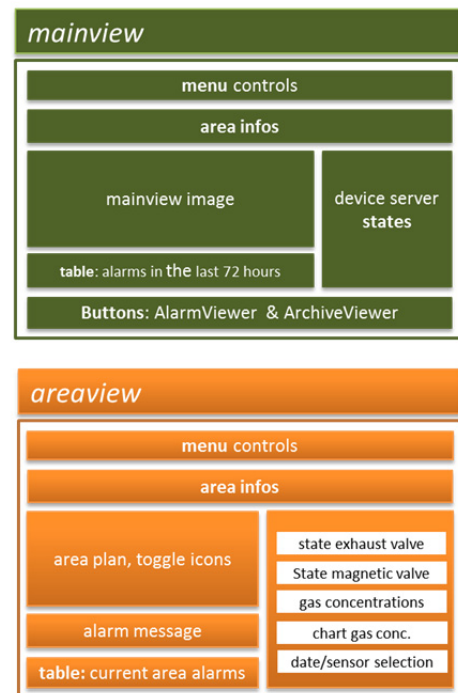


Figure 1: BeamlineAlarminfoClient GUI functionality of main and area view.

The Tine-Tool *Archive Viewer* is used for trending gas concentrations and *Alarm Viewer* for long time archiving of emergency alarms.

Motivation

Due to an unexpected increase in monitored areas, there is now a unique source code for every area server in the previous project. In 2013 a redesign of the project led to more maintainable software.

The main objective of this redesign is to reduce the project-code changes to a minimum of entries resulting from a new area as well as to reduce the communication links between server and client. In addition an upgrade of the graphical user interface is planned.

SYSTEM ARCHITECTURE

Every building displays one or more client application. The locations of these displays are at the entry of every building. Hence they are readily visible for technical-service personnel.

The client uses the time control system [1] to get data from the server and communicates with each of the n servers. Thus each server has m number of clients to provide with data (as outlined in Figure 2).

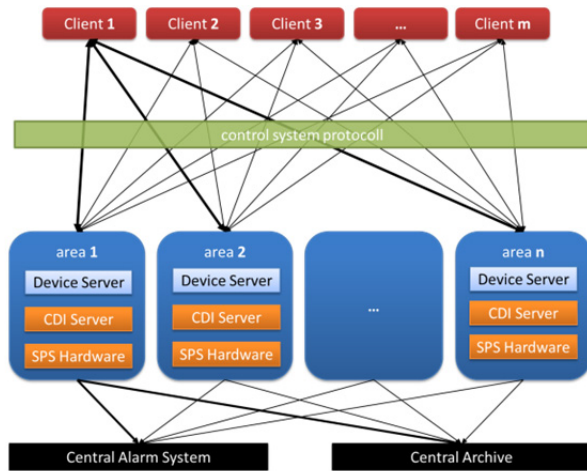


Figure 2: Communication between different server layers and the client.

Server

Before the client can display any data, this information has to be provided from a device server. Consequently the device server development depends on what alarm information the client needs to display. Due to the dependency of client and server development the main client requirements were discussed with the customers prior to any server development.

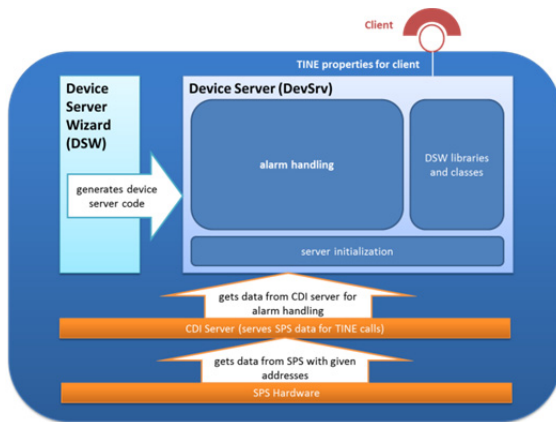


Figure 3: Device server architecture.

The server has priority because without data the client has nothing to display. In addition most logical operations concerning alarm handling are implemented at the server side. In Figure 3 the simplified server architecture is shown.

With the device wizard server [2] a basic source code for a device server is generated. Accordingly the programmer focus is on developing the handling of alarms.

During alarm handling (see Figure 4) SPS hardware data is checked for pre-defined emergency alarms. This data check occurs during the update() method of the device server. First the device server receives data from the CDI server, which maps the SPS hardware data. This data represents the value of emergency alarms.

Alarms are divided into alarm sensor states (digital alarm) and values of gas concentration (analog alarm). The server extracts the alarm information and checks digital alarms and, if needed, the analog alarms (if a gas alarm is set).

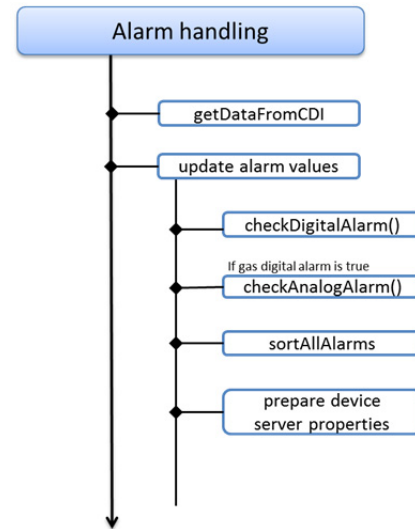


Figure 4: alarm handling.

All active alarms are saved into a list and sorted by their timestamp, and made available through property calls (active alarms). The cancelled alarms are listed separately and made available as an additional property.

Client

The client GUI is divided into jddd panels [3] and other java implementations. Jddd is useful for creating GUI components quickly, like displaying states of alarms with their icon location on the area plan. The main implementation is in rich-client java code.

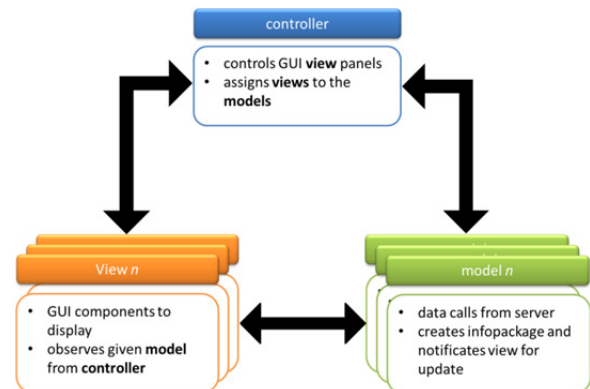


Figure 5: MVC pattern.

The GUI uses the model-view-control (mvc) architecture for receiving and displaying data from servers. Figure 5 shows the functions of the mvc components. The controller controls all views and models and for every area an instance of a model and view is created. An array list of area server is used in the controller class to initialize area views and models.

Communication

As previously mentioned some device server properties are utilized in simple jddd panels. However this is not discussed in this paper. Instead we focus on the time call links used in the main rich-client java code. Only one call back data link is defined to receive a property labelled *infobyte*. This property contains information about the CDI server state, existing active alarms, and whether or not an area alarm info text has been edited. Only when a flag is set in the received infobyte property, is the corresponding property called from the server. This process is shown in Figure 6.

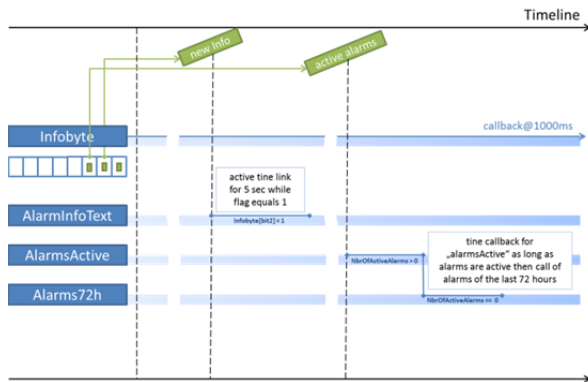


Figure 6: Client/Server communication, time calls.

STATUS / SUMMARY

At the end of this year the goal is to start with the alpha test of this project. First tests with some parts of server and client, primarily the correct extraction and representing of the alarm states, have already passed successful.

On upgrading the GUI the view became similar to the previous one, so that better recognition for the technical service is possible.

The server/client communication was reduced to one time call back link. Now only one server project exists with initializing files for a new area. Only the files have to be created or updated and only one project source code has to be edited. Parallel to this, new areas can be trivially added to clients.

As long as no new application functionalities are demanded on customer side the project maintenance has been dramatically simplified.

REFERENCES

- [1] TINE; <http://tine.desy.de>
- [2] Josef Wilgen, "First Experiences with a Device Server Generator for Server Applications for PETRA III", PCaPAC08, Ljubljana, Slovenia
- [3] jddd; <http://jddd.desy.de>

CONTROL SYSTEM SOFTWARE ENVIRONMENT AND INTEGRATION FOR THE TPS

Y. S. Cheng, Jenny Chen, C. Y. Liao, P. C. Chiu, C. H. Huang, C. Y. Wu, C. H. Kuo, K. T. Hsu
National Synchrotron Radiation Research Center, Hsinchu 30076, Taiwan

Abstract

The TPS (Taiwan Photon Source) is the latest generation 3 GeV synchrotron light source, and the commissioning starts from third quarter of 2014. The EPICS is adopted as control system framework for the TPS. The various EPICS IOCs have implemented for each subsystem. The control system software environment has been established and integrated specifically for the TPS commissioning. The various purposed operation interfaces have been created and mainly include the function of setting, reading, save, restore and etc. The database related applications have been built, and the applications include archive system, alarm system, logbook, Web and etc. The high level applications which are depended upon properties of each subsystem have been developed and are in test phase. The efforts will be summarized at this report.

INTRODUCTION

The TPS [1] is a latest generation of high brightness synchrotron light source which has been under construction at the National Synchrotron Radiation Research Center (NSRRC) in Taiwan since February 2010. The civil construction works are finished in later 2013. The TPS consists of a 150 MeV electron Linac, a 3 GeV booster synchrotron, and a 3 GeV storage ring, and the accelerator system installation and system integration had started from later 2013. The Linac and transport line (Linac-to-Booster) system had commissioned from half of 2014, and the booster ring is commissioning from the third quarter of 2014. The storage ring will schedule to commission from later 2014.

The EPICS (Experimental Physics and Industrial Control System) [2] framework was also selected as control system infrastructure for the TPS project. The EPICS platform has been gradually built and tested to control and monitor the subsystems of TPS. The various database records have been created for accessing the I/O data and setting parameters at the IOC (Input Output Controller) layer. Adopting the EPICS channel access mechanism with specific toolkits, the data can be accessed between the IOCs and the clients.

During the implementation process of the EPICS support for various subsystems, the operation interfaces of each subsystem are also developed according to the different operation methods. To simulate the operation process, the various operation interfaces are needed to integrate. The centralized management of the EPICS related files is also adopted, and the mechanism of save and restore will be continuously developed. The efforts will be summarized as following.

SOFTWARE ENVIRONMENT

The client consoles are adopted the Linux operation system. All of the EPICS base, modules, extensions and specific operation toolkits are installed at the Linux system. All EPICS related files at control consoles are mounted from the file server by using the NFS service to simplify software version management. Various directories are created and saved into various versions of related files for various hosts and purposes. Various directories provide a mount point for hosts mounted according to various purposes. The directories include EPICS base, modules, extensions, GUIs, saved data, temporary data and etc.

Several file servers are established to share the loading of NFS file service. The hosts mount specific file server according to its location and purpose. By loading testing, the NFS file service is divided into three parts. Two servers provide the NFS service for all IOCs and consoles; the other server is for engineer development.

SAVE AND RESTORE MECHANISM

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 [3]. The various files of grouped PVs (Process Variables) list are created for saving the respective parameter values of each subsystem. The file with PVs and saved parameters is also selectable for resume the settings. The graphical operation interface of save and restore mechanism is shown as the Fig. 1.

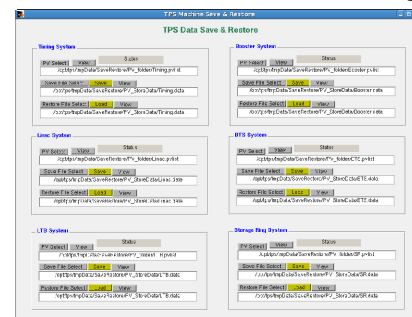


Figure 1: The OPI of save and restore mechanism.

SUBSYSTEMS CONTROL PAGES

At the commissioning phase, the main GUIs of TPS control system adopt the EDM (Extensible Display Manager) [4] toolkit to develop main graphical operation interface. The EDM OPIs of various subsystems are

created and saved into the NFS file server for client consoles operation.

The top-layer control page is built by the EDM toolkit shown as the Fig. 2. All control pages can be launched from this page. All control components are located at the foreground of the TPS accelerator illustration. For example, the LTB (Linac to Booster) dedicated control page is linked from the main control page for operation shown as the Fig. 3. The related control parameters or components are also located at the same page for tuning easily, such as the control page for the TPS timing as shown in the Fig. 4. These pages will be continuously refined and developed for the commissioning.

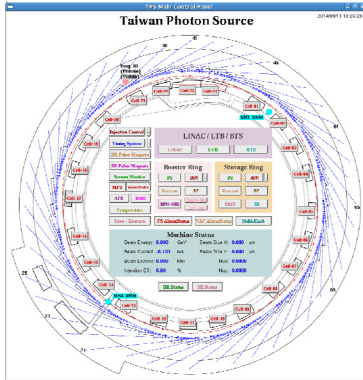


Figure 2: TPS main control page.

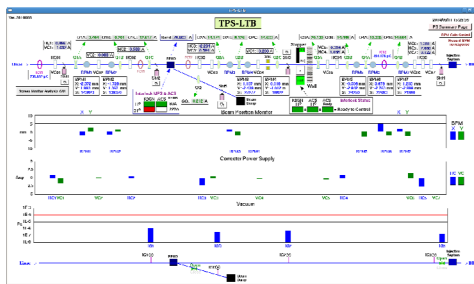


Figure 3: The LTB control page.

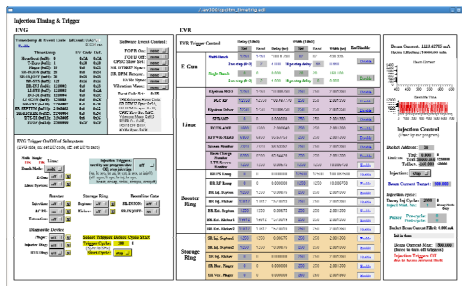


Figure 4: The control page for the TPS timing system.

The EDM toolkit is also adopted to develop the main control pages of power supplies. Each main page is included the same kind of power supplies. These main control pages are shown critical information for observing status easily, and the major operation functions are also executed from the main page. The sub-pages which included all detail parameters are also launched from this page. For example, the control page of booster ring power supplies is shown as Fig. 5.

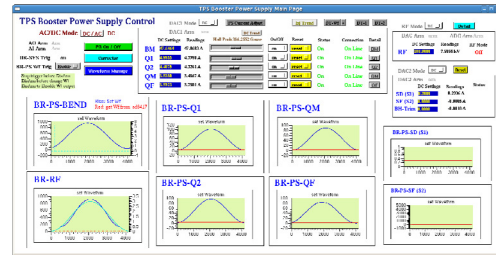


Figure 5: The control page of booster ring power supplies.

The status page of power supplies is created by the EDM toolkit for monitoring as shown in Fig. 6. These observed signals are usually the interlock signals. If one of power supply drops, this page will show one of red point to let operators know which problem is.

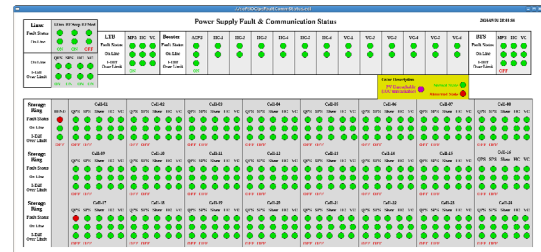


Figure 6: The status page for monitoring power supplies.

One of the benefits on EPICS waveform supports is that various LXI-based oscilloscopes can be controlled remotely and users can easily observe the acquired waveform data by using display toolkit on the control pages. The EPICS waveform supports of pulse magnets power supply CT, FCT, ICT, Linac RF power, WCM and etc. are built for observing and analyzing.

MATLAB-BASED APPLICATIONS

GigE Vision cameras are used to capture images. A dedicated EPICS IOC is used to acquire images and delivery waveform process variables (PVs) which represent the images. The image analysis IOC embedded MATLAB environment to perform image processing and analysis. The screen monitor analysis GUI as shown in Fig. 7 [5], has been developing and has various features including specify region-of-interest (ROI), optional background subtraction, 3D image viewing, and software multiple exposure. It also can create a simulated beam image for the purpose of evaluating the fitting correctness. All fitted parameters will be stored as EPICS PVs such that clients can easily access it for further usage.

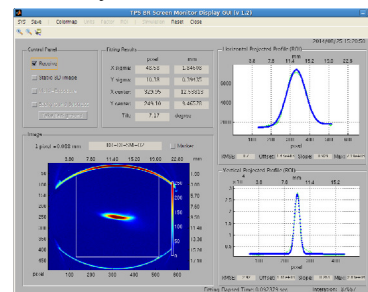


Figure 7: The Matlab analysis GUI for screen monitor.

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. Post-mortem analysis can help to find the weakest point and provide information to improve system reliability. The BPM GUI has created to analyze by using MATLAB. Fig. 8 shows main page for the TPS booster ring BPM first turn data.

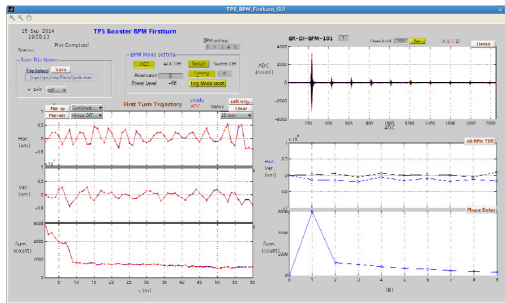


Figure 8: The OPI for TPS booster BPM first turn data.

DATABASE APPLICATIONS

The archive system of CS-Studio (Control System Studio) [6] which named “BEAUTY” (Best Ever Archive Toolset, yet) was built to be used as the TPS data archive system [7]. The PostgreSQL RDB was adopted as the data storage for the BEAUTY. The archived data can be retrieved in a form of graphical representation using the CS-Studio based data browser. Taking the performance and redundancy into considerations, the storage servers and RDB table structures are tuned relatively. This developed TPS archive system will be utilizing gradually during the installation and commissioning phases.

To observe the temperature variation for in-vacuum insertion device baking, the specific monitor GUI was created by the CS-Studio as shown in Fig. 9, and this page can be displayed the latest 30 minutes historic temperature data trend which retrieved from the database.

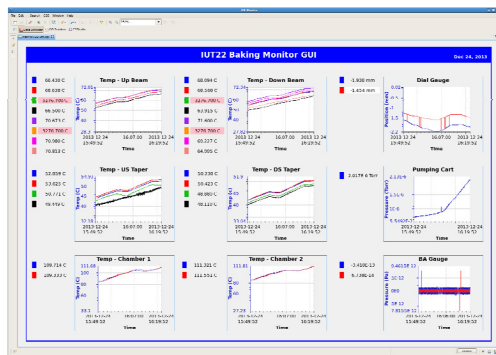


Figure 9: Archive data browsing GUI for ID baking.

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. 10. 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.

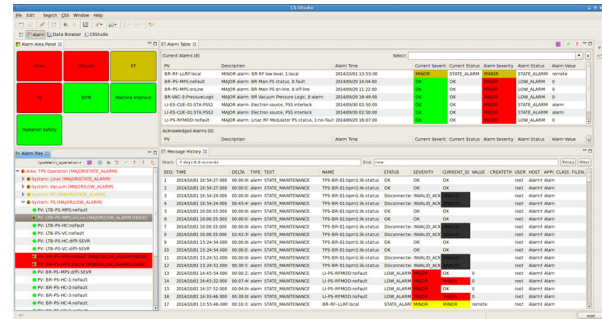


Figure 10: The “BEAST” of CS-Studio used for the TPS alarm system.

The “Olog” [8] 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.

SUMMARY

The TPS control system adopts the EPICS framework as the control infrastructure. The EPICS supports of various subsystems have been built, and the operation interfaces of various subsystems are also integrated in the meantime. The various GUIs are designed according to the different operation purposes. The management of the EPICS related files system has been established for maintaining files versions easily. The mechanism of save and restore are needed to apply parameters for operation process. Various operation interfaces are integrated and improved continuously during the TPS control system commissioning. Adopt newly developed supports accompany with the EPICS V4 will be included in deploy of the TPS control system includes directory service, high level applications, and etc.

REFERENCES

- [1] TPS, <http://www.nsrcc.org.tw/english/tps.aspx>
- [2] EPICS, <http://www.aps.anl.gov/epics/>
- [3] labCA, <http://www.slac.stanford.edu/~strauman/labca/>
- [4] EDM, <http://ics-web.sns.ornl.gov/edm/>
- [5] C. Y. Liao, et al., “TPS Screen Monitor User Control Interface”, FPO032, PCaPAC 2014, to be published.
- [6] CS-Studio, <http://ics-web.sns.ornl.gov/css/>
- [7] Y. S. Cheng, et al., “Implementation of the EPICS Data Archive System for the TPS Project,” Proceedings of IPAC2013, Shanghai, China, May 12-17, 2013.
- [8] Olog, <http://olog.sourceforge.net/olog/>

POWER SUPPLIES TRANSIENT RECORDERS FOR POST-MORTEM ANALYSIS OF BPM ORBIT DUMPS AT PETRA-III

G. K. Sahoo[#], P. Bartkiewicz, A. Kling, B. Pawlowski
Deutsches Elektronen Synchrotron, DESY, Hamburg, Germany.

Abstract

PETRA-III is a 3rd generation synchrotron light source dedicated to users at 14 beam lines with 30 instruments. The storage ring is presently modified to add 12 beam lines. PETRA III was operated with several filling modes such as 40, 60, 480 and 960 bunches with a total current of 100 mA at electron beam energy of 6 GeV. The horizontal beam emittance is 1 nmrad while a coupling of 1% amounts to a vertical emittance of 10 pmrad. During a user run the Machine Protection System (MPS) may trigger an unscheduled beam dump if transients in the current of magnet power supplies are detected which are above permissible limits. The trigger of MPS stops the ring buffers of the 226 BPM electronics where the last 16384 turns just before the dump are stored. These data and transient recorder data of Magnet Power Supply Controllers are available for a post-mortem analysis. Here we discuss in detail the functionality of a Java GUI used to investigate the transient behavior of the differences between set and readout values of different power supplies to find out the responsible power supply that might have led to emittance growth, fluctuations in orbits or beam dumps seen in a post-mortem analysis.

INTRODUCTION

PETRA-III [1] is a 3rd generation synchrotron light source commissioned with electron beam energy of 6 GeV and 100mA stored current at betatron tune values of 36.12 and 30.28. The horizontal beam emittance is 1 nmrad while a coupling of 1% amounts to a vertical emittance of 10 pmrad. The machine is dedicated to users for experiments from 14 beam lines with 30 end-stations. The storage ring is presently being modified to incorporate 12 new beam lines including a Superlumi beam line from dipole radiation. PETRA operates with several filling modes, such as 40, 60, 480 and 960 bunches with a beam current of 100 mA. During the normal user operation, there are unscheduled beam dumps triggered by the Machine Protection System (MPS) [2, 3]. These triggered dumps may occur before or some times after the loss of beam. The reasons for beam loss due to the MPS are of course understood. But the loss of beam prior to the beam dump by the MPS or a sudden fall of beam current, are unexpected. In these cases the reason remains unidentified or in some cases undetected. However, although the beam is lost, it leaves its signature in its post-mortem data. These post-mortem data are huge and contain a lot of information which can be extracted and analyzed in a Java Web Application MEOC [4]. Here we discuss how the Power Supply Controller (PSC)

Transient Recorders are used in the post-mortem analysis to pin point the source of disturbance in magnet power supplies.

TECHNICAL OVERVIEW

All PETRA III magnets are driven by power supplies designed and manufactured at DESY (Fig. 1), controlled by intelligent PSCs [5].



Figure 1: PETRA-III power supply modules.

The PSC design is based on generic controller mezzanine boards (Fig. 2), designed also at DESY and widely used for other control purposes as well. The board consists of Freescale Coldfire (MCF5282) microcontroller and Altera FPGA (Flex EPF10K50), offering not only control and communication capabilities, but also enough resources needed for real time output current monitoring and transient recording.

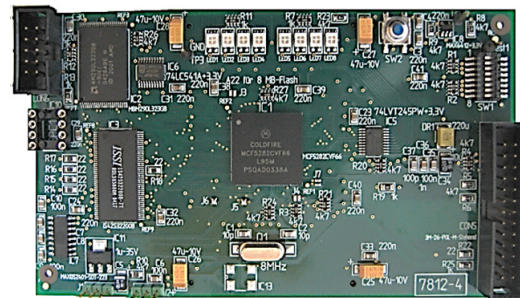


Figure 2: Generic Freescale Coldfire MCF 5282 based mezzanine card widely used for control purposes.

The 614 PSCs communicate over CAN buses (CANopen protocol [6]) with 20 front-end servers, running on PC104 Fanless Industrial Computers [7] with an embedded Linux operating system. The TINE [8, 9] network environment integrates the front-end servers with the PETRA-III control system and provides them access to central services, like data archiving, alarms and events recording systems (Fig. 3).

[#] Gajendra.Kumar.Sahoo@desy.de

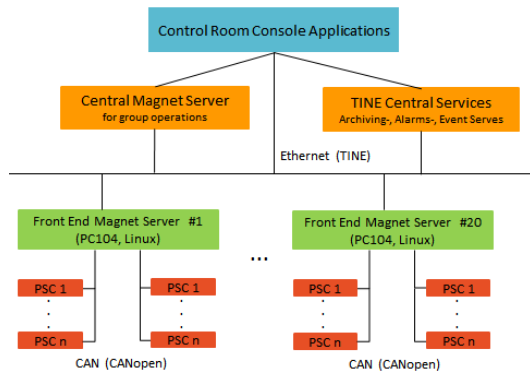


Figure 3: Schematic structural view of PETRA III magnet control system.

During a PETRA user run the circular buffer of length 20160, of which 16-bit words is filled every 500 microseconds with an averaged value of the difference between current set point and the measured current read back value. The number of samples used for averaging can be defined in range of 2 to 255. If the absolute value of the averaged difference is bigger than a configured threshold, the trigger is recorded. The local controller time is stored along with the trigger position, as well as the controller status bits. It is important that the local time of the PSC is synchronized with the front-end server time using the Time-stamp CANopen protocol, and the front-end server's time is synchronized with the global DESY control system time by TINE services. The trigger position within the circular buffer can be freely defined, making possible focusing on the recordings just before or just after the trigger. Since the transient recorder is designed to monitor transitions during a stable run and not during the machine energy ramp, the trigger is automatically disabled by the PSC while performing the change of power supply current. Additional parameters provide information about the trigger system activation, such as a time-delay following a ramp, in order to reject possible current fluctuations due to the magnet inductance.

At present all transient recorder parameters are set individually for each PSC using a dedicated client application. More sophisticated configuration tools, performing quick and convenient PSC group's configuration, are planned.

The front-end servers monitor periodically the status of all PSCs. For each triggered recorder the circular buffer content is transferred via the CAN bus to the server's memory and a Transient Recorder Event is sent to the TINE Event Server. In response to the event reception the Event Server fetches all copies of transient recorders buffers from the front-end servers and makes them available in the Archive System, where client applications can find them for further analysis.

GUI PSC TRANSIENT RECORDER

The Java GUI PSC Transient Recorder as shown in Fig. 4 is one of features in its main panels. The other panels are used to show the difference in set and read

back values, fast Fourier transforms, the Gaussian distribution, the Change of Events, as well as the Standard Out and Standard Error. There are provisions of choosing all PSCs, a particular type, or a particular PSC. The table shows the information associated with the PSC.

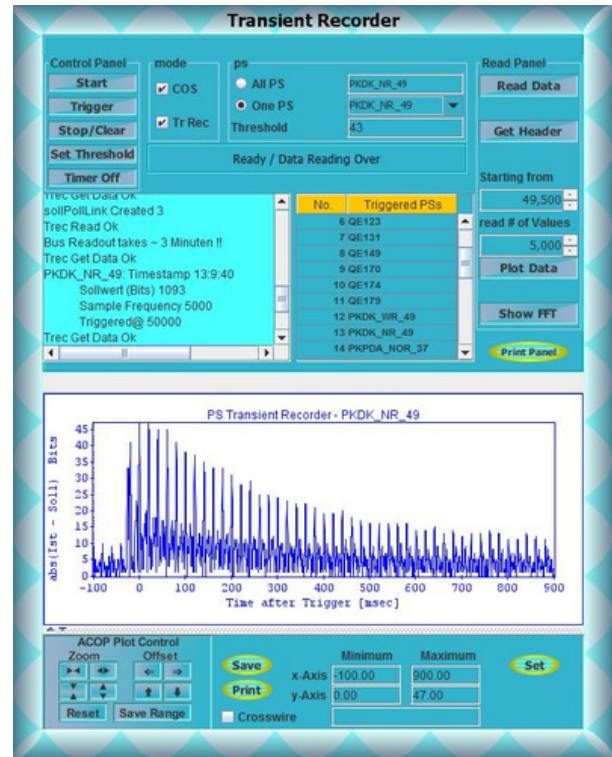


Figure 4: The graphics user interface of PSC Transient Recorder.

RESULTS AND DISCUSSIONS

Wrong Setting of Corrector Magnets

The orbit is corrected using slow orbit correction [10] based on the SVD algorithm employing 191 horizontal correctors, 187 vertical correctors and 226 BPMs. During normal user operation the golden orbit is maintained with additional 40 fast correctors on either plane using same algorithm. During the process of correction some correctors may set higher currents than desire leading to high orbit oscillations. This can also happen due to spikes in fast or slow corrector magnets. A small change in set current of any PS can be treated as the effect of an artificial corrector incorporated in it. So, in case of unknown beam dumps, the change in orbits in post-mortem data may be corrected choosing a few numbers of correctors. The MEOC is utilized to investigate the suitable corrector that might have perturbed the orbit beyond the interlock limits. For example, the event (Fri Mar 15 11:02:17 CET 2013) was due to the failure of the vertical corrector magnet PKVSU_SWL_46 which was receiving wrong set values due to spikes leading finally to a beam dump. You can see from Fig. 5 that the difference orbit was well corrected to zero using the same corrector.

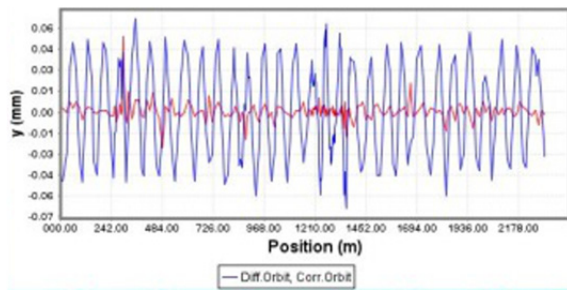


Figure 5: Vertical orbit correction for the Interlock Event on Friday 15 March 2013 at 11:02:17 which indicates that PKVSU_SWL_46 vertical corrector as the source of orbit perturbation.

Regulation Problems in PKVSX & PKVSU

There are 18 PKPDA type horizontal corrector magnets available as the back-leg windings in PDA type dipole magnets in the new DBA octant. These PSs are mostly trigger events in PSC Transient Recorders as the current controls in these are not good enough. Similarly, there are extra windings in sextupole magnets of S1, S3 and SDU to generate vertical correction coils. These coils are known as PKVSX (for S1, S3) and PKVSU (for SDU) type corrector magnet with a maximum of 55A, on 0.114mA/bit with a threshold of 240bits. But in operation many of these correctors are powered at less than 10A at which the controls do not work properly. Thus many of these PSs are frequently triggered in PSC Transient Recorders

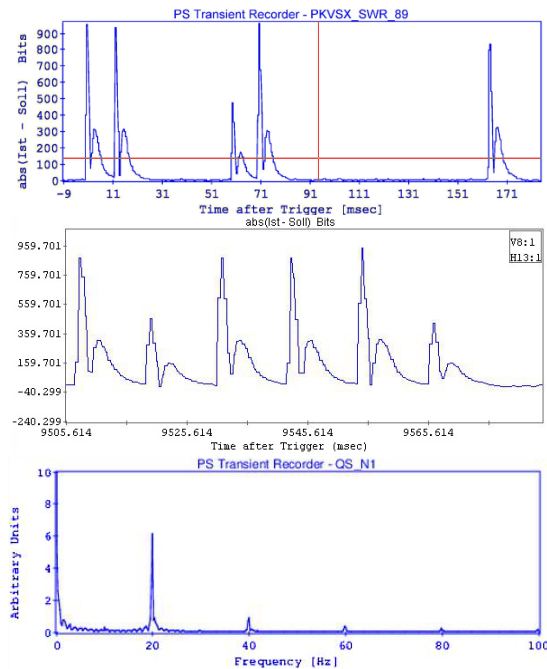


Figure 6: (a) The transients in the PS PKVSX_SWR_89, (b) The transients in the PS PKVSU_SWL_46 recorded in Transient recorders where the difference in between set & read back current is ~1000bits and (c) shows the frequency spectrum of the transients in PS QS_N1.

It is also noticed that the PSC associated with the quadrupole magnet Q4K_R is very often triggered. Some of the observations are shown in Figure 6 where the transients are much above the threshold limits of the respective power supplies.

CONCLUSION

The present PSC Transient Recorder is utilized to monitor the transients in 614 PSs of PETRA III electron storage ring. All the PSs are manually put in active mode which are triggered when the difference in read and set values are larger than the thresholds of respective PSCs. This is used with post-mortem analysis of BPM beam dumps to find the responsible PS that disturbs stability or causes loss of beam. We are currently developing auto triggers which would utilize circular buffers in the continuous monitoring of PSC transients in order to improve stability and reliability of PETRA III operation.

ACKNOWLEDEMENTS

The authors gratefully acknowledge and would like to thank for the initial contributions by Steve Herb; and Rainer Wanzenberg and Reinhard Bacher for their constant encouragements and guidance.

REFERENCES

- [1] K. Balewski, W. Brefeld, et al., "PETRA III: A new high brilliance synchrotron radiation source at DESY", EPAC-2004, Lucerne, 2004, pp.2302-2304.
- [2] T. Lensch, M. Werner, "Machine Protection System for PETRA III", Proceedings of DIPAC09, Basel, Switzerland, pp 351-353, 2009.
- [3] http://tftinfo.desy.de/petra/show.jsp?dir=/2011/17/29.04_a&pos=2011-04-29T15:26:23
- [4] G.K. Sahoo, K. Balewski, A. Kling, "Post-Mortem Analysis of BPM-Interlock Triggered Beam Dumps at PETRA-III", PCaPAC 2012, Kolkatta, pp 43-45.
- [5] B. Pawlowski, private communication.
- [6] <http://www.can-cia.org/index.php?id=canopen>
- [7] http://www.pc104.org/pc104_specs.php
- [8] P.K.Bartkiewicz, P. Duval "TINE as an accelerator control system at DESY", Measurement Science And Technology, 18 (2007) pp 2379-2386
- [9] <http://tine.desy.de>
- [10] G.K. Sahoo, K. Balewski, et al., "Closed orbit correction and orbit stabilization scheme for the 6 GeV synchrotron light source PETRA III", EPAC-2004, Lucerne, 2004, pp.2302-2304.

TPS SCREEN MONITOR USER CONTROL INTERFACE

C. Y. Liao[#], Y. S. Cheng, Demi Lee, C. Y. Wu, K. H. Hu, C. H. Kuo, K. T. Hsu
NSRRC, Hsinchu 30076, Taiwan

Abstract

The Taiwan Photon Source (TPS) is being constructed at the campus of the NSRRC (National Synchrotron Radiation Research Center) and in commissioning. For beam commissioning, the design and implementation of a screen monitor system for beam profile acquisition, analysis and display was done. A CCD camera with Gigabit Ethernet interface (GigE Vision) is a standard device for image acquisition, to be undertaken with an EPICS IOC via a PV channel; display beam profile and analysis properties are made with a Matlab tool. The further instructions for the design and functionality of the GUI were presented in this report.

INTRODUCTION

Taiwan Photon Source (TPS), a 3 GeV third generation synchrotron light facility, featuring ultra-high photon brightness with extremely low emittance which is being installation at National Synchrotron Radiation Research Center (NSRRC). For beam commissioning and optimize machine operation, the two-dimensional beam-related images were recorded by screen monitor in Linac, LTB, booster, BTS and storage ring, which are widely used in synchrotron light source facility. Due to the most of machine parameters in future TPS [1] will be accessible as EPICS (Experimental Physics and Industrial Control System [2]) process variables (PVs). Thus, an analysis tool use the PVs as inputs with ability to calculate and display results in complex ways is needed. The screen monitor user control interface design and its functionality are present in this report.

LAYOUT OF SCREEN MONITOR

For the TPS beam diagnostic application distributed in Linac, LTB, BTS, booster (BR), and storage ring (SR), the screen monitor is responsible for the beam profile acquisition from YAG:Ce screen and used to analysis to find the beam characteristic data. The location and quantity of the screen monitor is listed in Table 1. The beam profile image has extensive information on beam parameters, including beam center, sigma, tilt angle and etc. The optical system contains screen, lens, and lighting system. The screen monitor assembly consists of a hollow tube, a YAG:Ce screen with 25 mm in diameter and 0.5 mm in thickness. The YAG:Ce screen is mounted at 45° angle in one side to intercept the beam. A vacuum-sealed window is in the other end of the tube to extract the light. A CCD camera is mounted at a supporting tube with LEDs installed beside the CCD camera for illumination. A pneumatic device is used to move the whole assembly in or out. All of these devices are controlled remotely

including the CCD power control, screen in/out control and LED lighting system. The structure of the screen monitor assembly is shown in Fig. 1. The PoE CCD camera with Gigabit Ethernet interface (GigE Vision) will be a standard image acquisition device. The CCD timing trigger clock is locked with TPS injection system, which is produced from a local timing IOC (EVR).

Table 1: Location and Quantity of the Screen Monitor

Location	Quantity
Linac	5
LTB	5
Booster	6+1*
BTS	5
Storage ring	1+3*

* Plus additional screen monitor are temporarily installed during the commissioning.

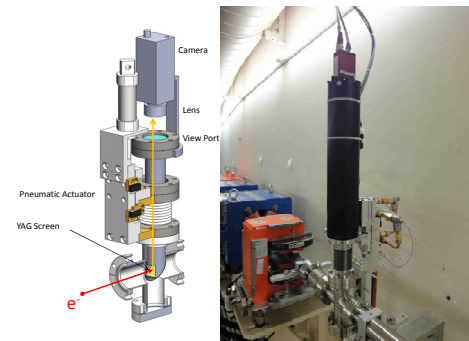


Figure 1: Screen monitor assembly in TPS booster.

USER INTERFACES DEVELOPMENT

Camera Parameters and Screen Position Control GUI

Based on the areaDetector module (R1-9-1) [3-4] which provides a general-purpose interface for area (2-D) detectors in EPICS, it is easy to construct a camera control panel by using the EDM tool. The TPS main control panel is shown in Fig. 2, the screen monitor launch page is marked. The screen monitor user interfaces for the TPS Linac, LTB, BR, BTS, and SR are shown in Fig. 3, which can switch in between in one GUI. The camera parameters of exposure time and gain can be configured in this panel, the camera location also shown below. The trigger mode selection include that the Free-Run for simply monitor the image and Sync-In for synchronization of linac injection (3 Hz). This EDM panel only offer lunch the screen, control the CCD parameters and simple monitoring features but do not perform any calculations. The screen position control also can perform in this GUI, the flow state as shown in Fig. 4.

[#]liao.cy@nsrrc.org.tw

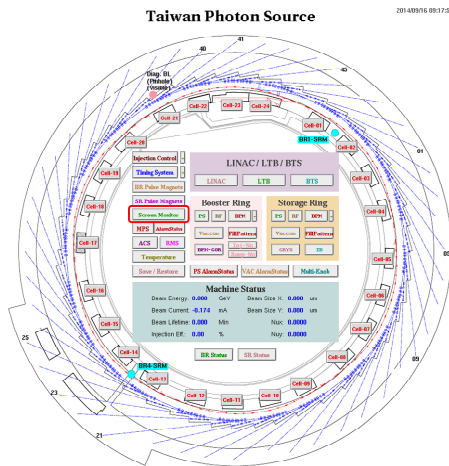


Figure 2: TPS control panel, screen monitor launch page is marked with red rectangular block.

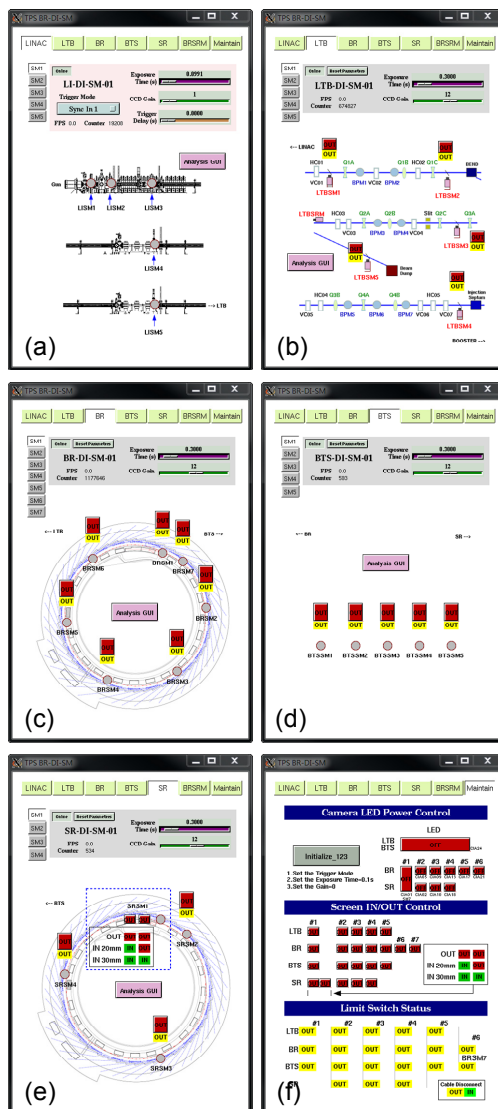


Figure 3: The EDM camera control user interfaces for the screen monitor of (a) Linac, (b) LTB, (c) BR, (d) BTS, (e) SR, and (f) maintain page. The top bar can switch in between the six pages, and the cameras location shows in below.

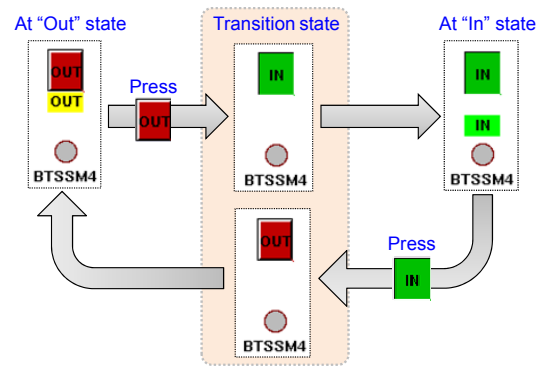


Figure 4: The screen position control flow state.

Matlab Display GUI

The display GUI was design to simply and quickly to use screen monitor, and it can be launched from the EDM page and display the image of beam profile, as shown in Fig. 5(a). When launch a screen, this action also specify which image will be analyzed by background analysis program. The axes of this figure in pixel unit (Left/Down) and mm unit (Right /Up) had already been calibrated. The status bar at the bottom of the figure is shown current capture date, system counter, and FPS information. The additional statuses, as shown in Fig. 5(b), such as screen out (screen position in out state), image no update (synchronous trigger signal loss) and camera pause will also show.

This display GUI contains right-click functions, the function name and description is shown in Table 2. It can be used to: control the camera update on/off, reset the GUI, save date to file, adjust the image contrast or colormap, image line profile analysis and distance measurement, and shown some information (PV name, resolution, size of image).

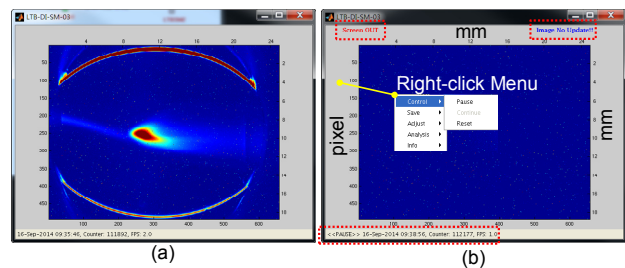


Figure 5: Screen monitor display GUI, (a) normal operation, (b) some information such as screen out, image no update, and camera pause will shown.

Table 2: Right-click Functions of Display GUI

Function	Sub-function Name	Description
Control	Pause	Pause capture image
	Continue	Continue capture image
	Reset	Reset this Program
Save	*.mat	Save as MAT file
	*.png	Save image as PNG file
Adjust	Contrast	Adjust image contrast
	Colormap - Gray	Change image to gray
	- Pseudo	Change image to pseudo (default)
Analysis	Line Profile	Analysis line profile on image
	Line Distance	Measure the line distance on image
Info	Image PV: LTB-DI-SM-03:image:ArrayData	Image array PV name
	1 pixel = 0.038 um	Image resolution
	Size: W656 x H492	Image size
	Ver.20140826	Current version

Matlab Analysis GUI

The image analysis work is mainly performed by using background Matlab program and display results via the Matlab GUI [5-6], the layout as shown in Fig. 6. The GUI can run in multiple clients simultaneously and read the analysis results from EPICS IOC and display them in the window. The GUI contains six parts: menu, toolbar, control panel, fitting results, projected profile, and raw image. The menu and toolbar provide save data, colormap change, ROI specify, simulation, reset and close program, and zoom in/out functions. In the control panel it contains active the program, 3D viewing, multi-exposure, and background subtract functions. The fitting results area contains sigma and center in the units of pixel and mm, and beam tilt angle. Two directions, horizontal and vertical, of beam projected profiles of raw data and fitting curve are predrilled in two axes. The camera raw image with the colorbar is integrated into the display GUI. The function of export the raw image data and analysis results can be done. It also can create a simulated beam image for the purpose of evaluating the fitting correctness.

The Matlab analysis program runs in IOC's PC as a background task which can do a complex analytical work to analyze the beam parameters, including the beam center, sigma, and tilt angle. It can specify the region-of-interest (ROI) for clipping the image of each individual camera, and do an optional background subtraction and software multiple exposures. All the analyzed data will store into the EPICS IOC as PVs.

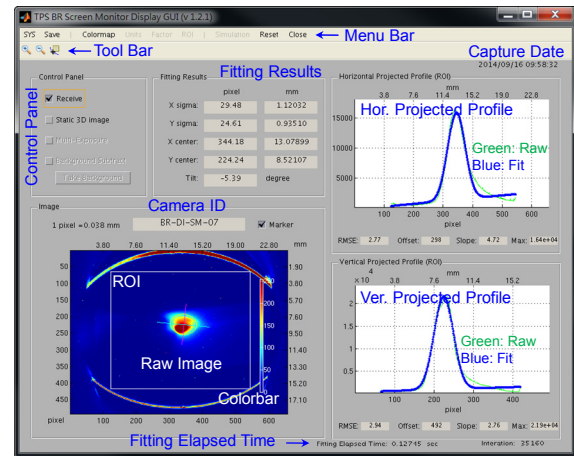


Figure 6: Layout of the Matlab analysis GUI for screen monitor.

SUMMARY

The Taiwan Photon Source (TPS) is currently in the commissioning phase, the screen monitor is one of the important diagnostic devices for beam commissioning. The user interface of screen monitor for display beam profile and analysis properties are made and online operation. Prolonged use and user feedback information will help the system reliability and integrity. More useful or desirable function will continue to grow up in this system.

REFERENCES

- [1] C. H. Kuo et al., "Conceptual Design of the TPS Control System", WPPA02, ICALEPCS07, <http://jacow.org/>.
- [2] EPICS (Experimental Physics and Industrial Control System), <http://www.aps.anl.gov/epics/>.
- [3] M. Rivers, areaDetector: EPICS software for area detectors, 2011, <http://cars9.uchicago.edu/software/epics/areaDetector.html/>.
- [4] M. Rivers, asynDriver: Asynchronous Driver Support, <http://www.aps.anl.gov/epics/modules/soft/asyn/>.
- [5] C. Y. Liao et al., "Beam Profiles Analysis for Beam Diagnostic Applications", TUPC146, IPAC2011, <http://jacow.org/>.
- [6] C. Y. Liao et al., "Diagnostics for the 150 MeV Linac and Test Transport Line of Taiwan Photon Source", TUPD04, DIPAC2011, <http://jacow.org/>.

BEAMLINE DATA MANAGEMENT AT THE SYNCHROTRON ANKA

A. Vondrous[#], T. Jejkal, D. Ressmann, W. Mexner, R. Stotzka
KIT, 76344 Eggenstein-Leopoldshafen, Germany

Abstract

We present a data management architecture consisting of beamline data management (BLDM) and data repository to enable data management at the synchrotron facility ANKA. Nearly each measurement device writes data with a different format, size and speed on storage devices that are distributed over the synchrotron facility. The operators perform some data management tasks manually and individually for each measurement method. In order to support the operators, users and data analysts to manage the datasets, it is necessary to collect the data, aggregate metadata and to perform ingests into the data repository. The data management layer between the measurement devices and the data repository is referred to beamline data management, which performs data collection, metadata aggregation and data ingest. Shared libraries contain functionalities like migration, ingest or metadata aggregation and form the basis of the BLDM. The workflows and the current state of execution are persisted to enable monitoring and error handling. After data ingest into the data repository, archiving, content preservation or bit preservation services are provided for the ingested data. The data repository is implemented with the KIT Data Manager. In summary, BLDM can connect the existing infrastructure with the data repository without major changes of routine processes to build a data repository for a synchrotron.

INTRODUCTION

Data management is essential for science in the information age. Preserving primary data produced in scientific investigations is not only important to establish scientific integrity [1, 2, 3]. The commodity for gaining knowledge and competitive advantages is data. Extracted relations from data stocks provide insights and help to understand phenomena. Beside good scientific practice and efficient data analysis methods, science also benefits from sharing data.

Astronomy is the textbook example for increasing the scientific outcome by sharing data. As an example, according to the Hubble space telescope bibliography [4] about 50% of the published papers related to measurements of the Hubble space telescope are based on publicly available data.

To create the basement for responsible data preservation, the European commission [5] introduces data management requirements for funding scientific investigations. Large facilities like synchrotrons produce a not negligible amount of valuable data, such that we try to find a convenient way to establish data management capabilities for synchrotrons.

The following two examples of biology and materials science illustrate the reasons for an elaborated data management at a synchrotron facility.

The value of measurement data produced in a synchrotron is not only determined by hardware, experience and operational methods, it is also determined by the application. As an example, measurements of biological specimens reveal insights into the biomechanical processes of insects as shown e.g. by van de Kamp et al. in [6]. The value to the biology community is higher than the actual costs because the insights answer questions, confirm argumentation chains and create new perspectives. Storing the raw and derived data is of interest to preserve the findings for further analysis.

Sometimes, the scientific application does not exist yet and the value of the measurement data cannot be properly estimated. Measurements in the field of materials science have the potential to be of use for science and industry in the future. The crystallography open database [7] is one example for preserving crystallographic data and providing access to structured datasets.

Storing data in a structured manner for long-term usage is a challenge in the synchrotron context. The broad spectral range of the produced light and the number of measurement huts (beamlines) enable the usage of diverse measurement methods simultaneously.

Manual data management by beamline operators with logbooks or with spreadsheets is common practice. Tasks as search, retrieval, analysis or conversion are becoming demanding considering large files, many files or distributed storage locations.

To preserve valuable data, to automate data management tasks and to support operators and users at the synchrotron ANKA, a data repository is going to be implemented. The data repository keeps track of all datasets and provides data services like search, preservation, analysis, publication, curation, processing or migration.

Those benefits of a data repository are not for free. It is necessary to agree on data structures, to define metadata schemes and to aggregate the metadata. In addition, the data management has to be aware of the synchrotron specific infrastructure properties. Finally, the data management should interfere as less as possible to enable smooth beamline operation.

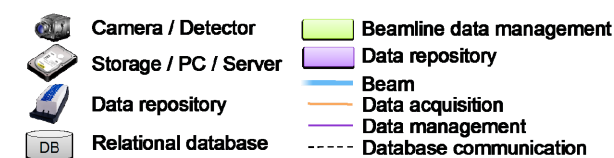
In this work we present data management from the source to the data repository considering synchrotron specific requirements.

METHOD

We divide the data flow from the data source (camera or detector) to the archive into two management areas. The first area is beamline data management (BLDM),

[#] Corresponding author: alexander.vondrous@kit.edu

The diagram illustrates a hybrid storage architecture. On the left, four server icons represent data sources. Each source is connected to a green box containing a hard disk icon, representing a data tier. These tiers are connected to a central 'DB' (Database) component. Below the DB is a 'Cache' component, which is connected to the DB and the data tiers. To the right of the Cache is an 'Archive' component, which is connected to the Cache and the DB. A 'Maintainer' component is shown at the top, connected to the DB, Cache, and Archive. A group of five user icons (labeled 'User') and one operator icon (labeled 'Operator') are shown on the right, connected to the Cache and Archive components.



Beamline Data Management

Because the datasets are valuable, we protect them against accidentally manipulation by users through changing the ownership after data acquisition to a service account referred to data repository user and provide read only access to the investigating user group.

Before the datasets are ingested, metadata is aggregated and checksums of the files are computed to detect data transfer errors. The final steps are to register the dataset in the data repository and to copy the dataset content to the cache location. The cache is managed by the data repository, which checks file integrity and copies the files to the archive storage. The metadata of the dataset are used for efficient search and data organisation. The following list represents an expected workflow at the beamline.

1. Create empty dataset
2. Measure data
3. Store dataset
4. Change owner
5. Change mode
6. Compute checksum
7. Aggregate metadata
8. Register dataset

9. Copy files to cache
10. Release for archiving workflow
11. Remove dataset from BLDM storage

To connect the data sources with the data repository for convenient data management, we provide an interface of data management tasks to the control system. The control system used at ANKA for user operation is TANGO [9]. With this interface, data management processes can be integrated into the daily measurement tasks of beamline operators. Figure 2 illustrates the structure of one task handler instance with some tasks.

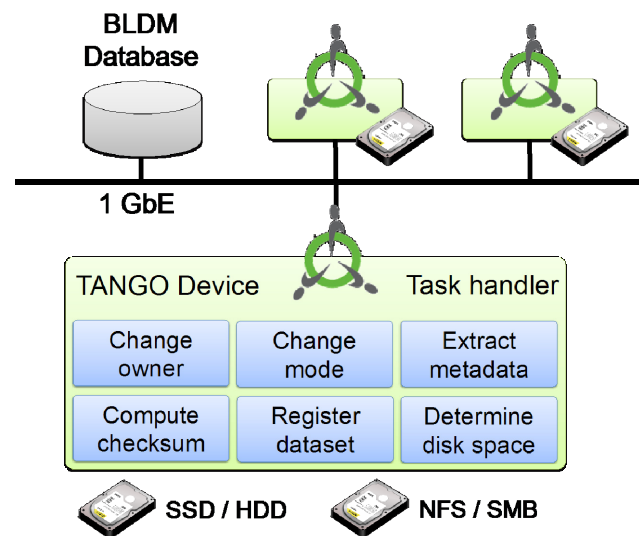


Figure 2: One task handler instance (green) using shared library tasks (blue) with a TANGO interface to integrate data management functionality into the synchrotron control system. Each task has access to the BLDM database.

The tasks and the TANGO device are implemented in C++ with the POCO [10] and Boost [11] libraries for logging and shared library access. It is necessary to provide operations on files after the data acquisition on the PC.

Data Repository

The data repository is implemented with the KIT Data Manager [12]. After the task handler registered and copied the dataset to the cache location, the data repository archiving workflow is triggered. The archiving process consists mainly of file integrity checks, file copy operations and metadata indexing. A representational state transfer (REST) interface can be used to build a user

interface or to access data with other programs or computers for further investigation.

Ingest Setup

We evaluated the transfer rate during the ingest process from a workstation to the data repository cache. The ingest process consists of 21 REST calls to register the dataset (8th workflow step) and the dataset upload via WebDAV (9th workflow step) over a one gigabit Ethernet connection between two Linux machines. Each dataset consists of one metadata file (382 byte) and one measurement data file with random content. The file size of the measurement data file is varied from one byte to 100 GB. Three ingests for each dataset are performed. The average value is used to estimate the total transfer rate.

RESULTS

The data repository and the BLDM are under construction but the key components are implemented:

Create empty dataset; Store dataset; Change owner; Change mode; Compute checksum; Register dataset; Copy files to cache

The evaluation results of the ingest processes are depicted in Table 1. The transfer rate is increasing with the file size. A file transfer of a 1 byte and a 1 KB measurement data file takes 7.28 and 7.26 seconds, such that we define the time to register the data to be 7.26 seconds. This registration time is used to estimate the transfer rate of the data upload. The estimated transfer rates for 1 MB and 10 MB without registration time do not reach a high transfer rate. Above 100 MB of file size the estimated transfer rates are greater 100 MB/s. The registration time of about 7 seconds is negligible for the large 100 GB dataset.

Table 1: Mean Ingest Runtime for Different File Sizes

File size	Mean time	Total transfer rate	Estimated data transfer rate*
1 byte	7.28 s	0.00 MB/s	-
1 KB	7.26 s	0.00 MB/s	-
1 MB	7.32 s	0.14 MB/s	16.67 MB/s
10 MB	7.58 s	1.32 MB/s	31.25 MB/s
100 MB	8.13 s	12.30 MB/s	114.94 MB/s
1 GB	16.75 s	61.13 MB/s	107.90 MB/s
10 GB	103.84 s	98.61 MB/s	106.03 MB/s
100 GB	955.85 s	107.13 MB/s	107.95 MB/s

The measurement results are performed with a simple test case consisting of one file. A more complex evalua-

* Establishing the connection and data registration are considered as registration time, such that 7.26 s are removed from the Mean time to estimate the data transfer time.

tion under production conditions with high performance hardware should be conducted.

CONCLUSION

The separation into two data management areas enables provides a convenient implementation of the interface with the synchrotron specific control system for seamless integration into beamline operation.

The presented architecture to execute data related tasks is one component for data management to hide complexity and serve modern data management requirements.

ACKNOWLEDGMENT

The authors thank the German Helmholtz Association for funding the project, the Large Scale Data Management and Analysis project, and the Large Scale Data Facility for providing storage resources. We also thank Halil Pasic for his contributions to the project.

REFERENCES

- [1] Deutsche Forschungsgemeinschaft, "Safeguarding Good Scientific Practice", WILEY-VCH 2013
- [2] Max-Planck-Gesellschaft, "Regeln zur Sicherung guter wissenschaftlicher Praxis", MPG 2009, <http://www.mpikg.mpg.de/23731> (visited Oct. 2014)
- [3] European Science Foundation, "Good scientific practice in research and scholarship", ESF 2000, http://www.esf.org/fileadmin/Public_documents/Publications/ESP10.pdf (visited Oct. 2014)
- [4] J. Lagerstrom, "Measuring the Impact of the Hubble Space Telescope: open data as a catalyst for science", World Library and Information Congress: 76th IFLA 2010 in Gothenburg, Sweden
- [5] European Parliament and Council, "establishing Horizon 2020 – the Framework Programme for Research and Innovation", Official Journal of the European Union 2013
- [6] T. van de Kamp et al., "A Biological Screw in a Beetle's Leg" Science 1 July 2011: 333 (6038), 52. [DOI:10.1126/science.1204245]
- [7] S. Gražulis et al. "Crystallography Open Database (COD): an open-access collection of crystal structures and platform for world-wide collaboration". Nucleic Acids Research 40, p. 420-427
- [8] E.S. Raymond, *The Art of UNIX Programming*, (Addison-Wesley Professional 2003)
- [9] TANGO Control System, <http://www.tango-controls.org/> (visited Oct. 2014)
- [10] POCO, <http://pocoproject.org/> (visited Oct. 2014)
- [11] BOOST, <http://www.boost.org/> (visited Oct. 2014)
- [12] C. Jung, A. Streit, Large-Scale Data Management and Analysis (LSDMA) - Big Data in Science, (Karlsruhe 2014) page 9

RENOVATING AND UPGRADING THE Web2cToolkit SUITE: A STATUS REPORT

R. Bacher, DESY, Hamburg, Germany

Abstract

The Web2cToolkit is a collection of Web services. It enables scientists, operators or service technicians to supervise and operate accelerators and beam lines through the World Wide Web. In addition, it provides users with a platform for communication and the logging of data and actions. Recently a novel service, especially designed for mobile devices, has been added. Besides the standard mouse-based interaction it provides a touch- and voice-based user interface. In addition, Web2cToolkit has undergone an extensive renovation and upgrading process. Real WYSIWYG-editors are now available to generate and configure synoptic and history displays, and an interface based on 3D-motion and gesture recognition has been implemented. Also the multi-language support and the security of the communication between Web client and server have been improved substantially. The paper reports the complete status of this work and outlines upcoming development.

INTRODUCTION

The Web2cToolkit [1] is a collection of Web services, i.e. servlet applications and the corresponding Web browser applications, including

1. *Web2cViewer*: Interactive synoptic live display to visualize and control accelerator or beam line equipment,
2. *Web2cViewerWizard*: Graphical WYSIWYG-editor to generate and configure synoptic displays,
3. *Web2cArchiveViewer*: Web form to request data from a control system archive storage and to display the retrieved data as a chart or table,
4. *Web2cArchiveViewerWizard*: Graphical WYSIWYG-editor to generate and configure archive viewer displays,
5. *Web2cGateway*: Application programmer interface (HTTP-gateway) to all implemented control system interfaces,
6. *Web2cMessenger*: Interface to E-Mail, SMS and Twitter,
7. *Web2cLogbook*: Electronic logbook with auto-reporting capability,
8. *Web2cManager*: Administrator's interface to configure and manage the toolkit, and
9. *Web2cToGo*: Interactive display especially designed for mobile devices [2] embedding instances of all kinds of Web2cToolkit services.

The Web2cToolkit provides a user-friendly look-and-feel and its usage does not require any specific programming skills. By design, the Web2cToolkit is platform independent. Its services are accessible through

the HTTP/HTTPS protocol from every valid network address if not otherwise restricted. A secure single-sign-on user authentication and authorization procedure with encrypted password transmission is provided.

The Web 2.0 paradigms and technologies used include a Web server, a Web browser, HTML5 (HyperText Markup Language), CSS (Cascading Style Sheets) and AJAX (Asynchronous JavaScript And XML). The interactive graphical user interface pages are running in the client's native Web browser or in a Web browser embedded in a mobile app or desktop application. The interface is compatible with almost all major browser implementations including mobile versions. The Web2cToolkit services are provided by Java servlets running in the Web server's Java container. The communication between client and server is asynchronous. All third-party libraries used by the Web2cToolkit are open-source.

The Web2cToolkit provides interfaces to major accelerator and beam line control systems including TINE [3], DOOCS [4], EPICS [5] and TANGO [6]. The toolkit is capable of receiving and processing video frames or a continuous series of single images.

In addition the toolkit provides an environment to support and test various Human-Machine-Interface (HMI) types including mouse, touch, speech and 3D-gestures depending on the capabilities of the underlying platform [7] and the proper design and handiness of multi-modal accelerator control system applications.

IMPROVED AND NOVEL FEATURES

Recently, the Web2cToolkit suite has been substantially renovated and upgraded. Besides smaller modifications such as supporting IP6-compliant client address encoding the objectives of this process include

- Improving the security of the communication between Web client and server
- Improving the multi-language support of the Web2c Viewer,
- Providing a specific user repository holding all user-defined configurations within the web server's directory structure,
- Supplying wizard applications to graphically design Web2cViewer synoptic displays and Web2cArchiveViewer history pages.
- Redesigning the interface for extending the Web2c Viewer servlet with code provided by the user,
- Redesigning the interface for connecting other accelerator and beam line control systems with the Web2c Viewer servlet, and

- Designing interfaces for connecting other accelerator archive systems and other video sources with the Web2c Viewer servlet.

Secure Communication

The Web2cToolkit can be configured to enforce secure data communication between client and server application by using the HTTPS protocol standard based on SSL/TSL.

Multi-Language Support

Web2cViewer synoptic live displays and the common login page of the Web2cToolkit can be configured to support more than one language at a time. All Unicode-compliant characters encoded as UTF-8 e.g. to be used for widget captions or tooltip descriptions can be displayed properly including complex characters such as Japanese characters.

User Repository

A repository for user-defined configurations has been added to the Web2cToolkit. It is located within a specific directory of the Web server which also holds the Web2cToolkit initialization files and does not belong to the directory tree of the Web2cToolkit Web server application which is overwritten during the deployment of the application. All user-defined files are restored from the repository at application start-up and saved into the repository when the application is terminating.

Wizards

To facilitate the configuration of synoptic live displays as well as generic or pre-defined 24h history displays two real WYSIWYG graphical editor applications have been implemented.

Fig. 1 shows a sample page of the Web2cViewerWizard application. It replaces the deprecated Web2cEditor. The wizard provides different views according to the task currently to be done. Common to each view is the large sketch board whereas the smaller panels on the left side may show a file explorer, a page outline and a list of available widget types or the list of attributes of the currently selected component, respectively.

The Web2cArchiveViewerWizard application is based on a similar lay-out. It hides the complex and error-prone configuration XML-structure of Web2cArchiveViewer pages which has been kept unchanged for compatibility reasons.

Besides creating new pages both wizards can download and parse configuration files which already exist in the user repository located at the Web server. The loaded configurations can be modified and uploaded to the user repository. A full version history of all uploaded configuration files is retained at the Web server.

Finally the page designer can preview newly created or modified Web2cViewer or Web2cArchiveViewer pages without leaving the designing environment.

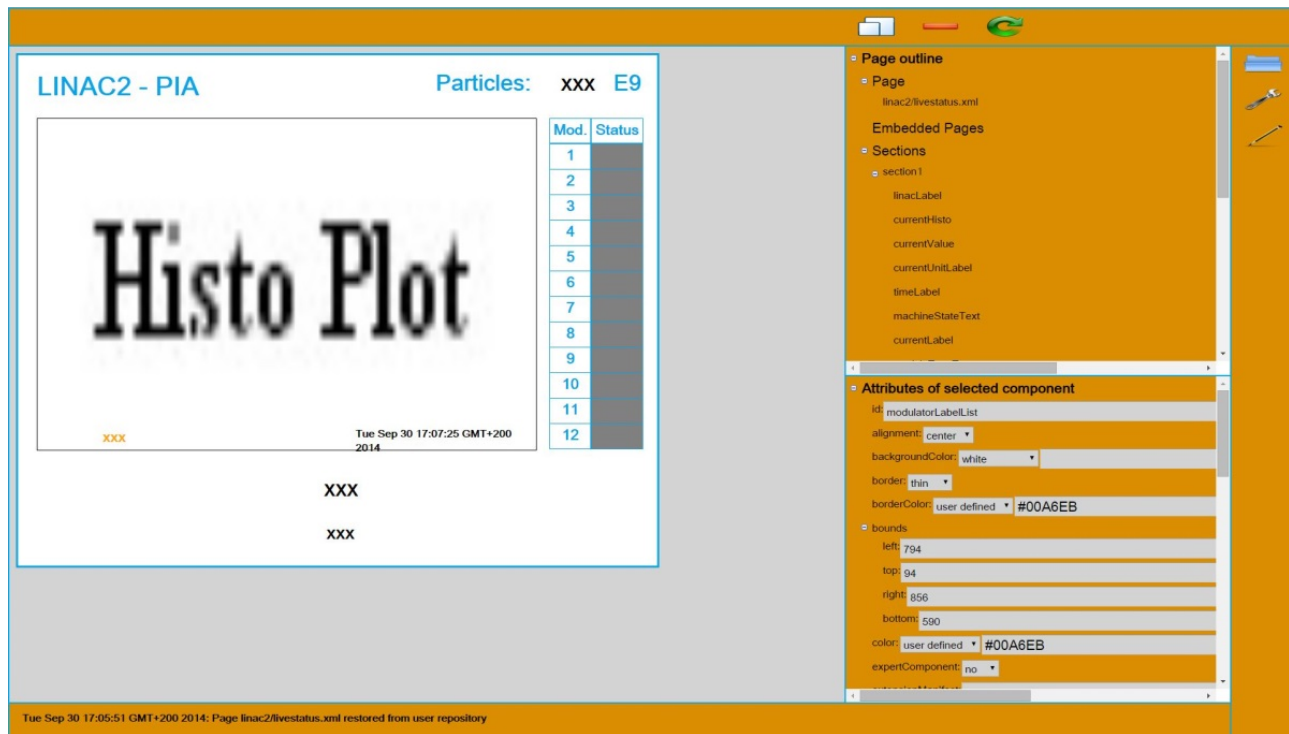


Figure 1: Web2cViewerWizard Web application showing an operations overview panel for the LINAC2 pre-accelerator at DESY. The panel is being graphically designed with the wizard. The corresponding configuration file is created by the wizard and uploaded to the user repository at the Web server.

User-Defined Widget Extensions and Plugs

The standard functionality of many Web2cViewer widgets can be extended by supplemental user-defined features. In particular it might be useful to apply a calibration to some raw data or perform a more complex data manipulation such as a Fourier transformation.

Besides the standard plugs to connect the Web2cToolkit with an external accelerator or beam line control system, archive system or video source interface plugs to other resources can be added. Recently, a plug for the STARS [8] control system used at the KEK Photon factory has been implemented.

Both user-defined extensions and interface plug-in classes have to be provided as Java archive files and added to the deployed Web2cToolkit Web application before application start-up.

ONGOING DEVELOPMENTS

A user can interact with a Web2cToGo application by mouse actions, finger touch gestures, spoken commands and hand or finger 3D-gestures [7]. With the exception of the mouse actions all interfaces are still a subject of research and current implementations and capabilities might change in the future. In particular the combination of all kinds of interfaces in a common multi-modal user application has still to be investigated and suitable use-cases have to be identified and tested.

Interaction by Speech Recognition

The response to spoken commands has been significantly improved. Repetitive client-side audio recording of short half-second audio sequences is performed using the most recent functionalities provided by HTML5. The communication to send a recently recorded audio sequence to the server for a phonetic analysis and to notify the client of the result is based on the bi-directional WebSocket protocol. Speech recognition is performed by the Web2cToolkit server application. The application concatenates the last uploaded audio sequences to a contiguous record which is a few seconds long. Whenever a new audio update is received by the server the leading audio sequence is clipped and the most recent audio sequence is appended to the trailing edge, allowing for almost continuous speech recognition. The application performs a search procedure using the most recent Sphinx-4 [9] speech recognition software. The overall response to a spoken command is almost immediate.

Interaction by Gesture Recognition

Interaction by finger touch gestures is already an established and common feature of modern mobile phones and tablet computers. Web2cToGo provides a set of intuitive application-specific single- and multi-finger gestures, although this set is still somewhat incomplete.

The recognition of real hand gestures has been popularized by game consoles. Web2cToGo uses the LEAP motion controller [10] to detect hand or finger 3D-gestures (Fig. 2). The corresponding sensor is a USB-device. The controller software runs on every PC (Windows, Linux, or MacOS) and provides a Web interface based on the WebSocket protocol. An extensive API facilitates the integration of the LEAP motion controller into a user application.



Figure 2: LEAP motion sensor device [10].

The Web2cToGo application displays a virtual cursor on top of the Web2cToGo desktop to guide the user. A preliminary and still incomplete set of finger- and hand-gestures has been implemented. The Web2cToGo application communicates with the local LEAP motion controller Web server and the 3D-gesture recognition does not involve the corresponding Web2Toolkit Web server application. Preliminary performance tests reveal a prompt response to 3D-gestures.

REFERENCES

- [1] Web2cToolkit; <http://web2ctoolkit.desy.de>
- [2] R. Bacher, "Web2cToGo: Bringing the Web2cToolkit to Mobile Devices", PCaPAC'12, Kolkata, India, December 2012, WEIC01, p. 4 (2012); <http://JACoW.org/>.
- [3] TINE; <http://tine.desy.de>
- [4] DOOCS; <http://doocs.desy.de>
- [5] EPICS; <http://www.aps.anl.gov/epics>
- [6] TANGO; <http://www.tango-controls.org>
- [7] R. Bacher, "Enhancing the Man-Machine-Interface of Accelerator Control Applications with Modern Consumer Market Technologies, ICALEPCS'13, San Francisco, USA, October 2013, THCOAAB02, p. 1044 (2013); <http://JACoW.org/>.
- [8] T. Kosuge, Y. Nagatani, "Current Development Status of STARS, PCaPAC'14, Karlsruhe, Germany, October 2014, WP0019, <http://JACoW.org/>.
- [9] Sphinx; <http://cmusphinx.sourceforge.net/sphinx>
- [10] LEAP; <https://www.leapmotion.com/>.

OpenGL-BASED DATA ANALYSIS IN VIRTUALIZED SELF-SERVICE ENVIRONMENTS

V. Mauch, M. Bonn, S. Chilingaryan, A. Kopmann, W. Mexner, D. Ressimann
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Abstract

Modern data analysis applications for 2D/3D data samples require complex visual output features which are often based on OpenGL, a multi-platform API for rendering vector graphics. They demand special computing workstations with a corresponding CPU and GPU power, enough main memory and fast network interconnects for a performant remote data access. For this reason, users depend heavily on available free workstations, both temporally and locally. The provision of virtual machines (VMs) accessible via a remote connection could avoid this inflexibility. However, the automatic deployment, operation and remote access of OpenGL-capable VMs with professional visualization applications is a non-trivial task.

In this paper, we discuss a concept for a flexible analysis infrastructure that will be part in the project ASTOR, which is the abbreviation for “Arthropod Structure revealed by ultra-fast Tomography and Online Reconstruction”. We present an Analysis-as-a-Service (AaaS) approach based on the on-demand allocation of VMs with dedicated GPU cores and a corresponding analysis environment to provide a cloud-like analysis service for scientific users.

INTRODUCTION

Particle accelerators like ANKA [1] provide synchrotron radiation for the investigation of solid material and biological samples. X-ray imaging techniques produce large data sets of up to several 100 Gigabytes. The subsequent processing of the data requires special analysis applications whose features and performance depend heavily on available CPU, GPU and memory resources. Typical analysis application like Amira [2] or VG Studio MAX [3] provide visual output based on OpenGL [4], a multi-platform API for rendering vector graphics. Up to now, several conditions concerning the user analysis process have complicated a flexible analysis workflow:

- transfer, access and storage of huge data sets
- required amount of computing and memory resources
- OpenGL/DirectX capable GPUs for visual output, eventually also CUDA [5] or OpenCL [6] are necessary
- expensive workstation licenses for commercial analysis software

All aspects mentioned above prevent the use of standard end-user devices for analysis and result in the operation of dedicated workstations for scientists. The required professional workstations need to provide a high-throughput network connection to the data set storage, licensed analysis software and hardware setup. This scenario implies further

disadvantages for users. They have to rent free time ranges for the workstation usage. Therefore they have to be present in special computing rooms within their institution shared with other users. Due to the operation of powerful hardware combined with an increased heat generation, inside the computing room one can often sense a high background noise. Furthermore, the workstations are typically configured with limited guest account privileges, which is why users have to consult an administrator to install additional software packages for a special purpose.

To improve this situation, one part of the project ASTOR deals with the automatic deployment of virtual machines (VMs) with remote connections supporting the rendering and display of OpenGL features. Scientists should be able to use a web portal with an overview of their available data sets and allocate virtual resources for their analysis on-demand. In this paper we discuss an Analysis-as-a-Service (AaaS) concept and present a first prototype implementation for the synchrotron community.

CONCEPT

The replacement of static localized stand-alone workstation towards a flexible analysis infrastructure concept for scientists is achievable with an Infrastructure-as-a-Service (IaaS) approach. Besides the provision of virtualized analysis environments, a suitable analysis workflow has to be defined which also considers the previous data detection and recording, data set access and the final result archiving. The complete analysis workflow could be considered as a novel Analysis-as-a-Service (AaaS) concept, defining a cloud-like service for individually customized analysis processes.

The main part of this concept is the intelligent integration of virtualized analysis environments. However, until a few years ago, the provision of remote connections to VMs supporting OpenGL or other complex visual APIs was impossible, as most virtualization solutions only came up with simplified graphic interfaces within a guest system. Furthermore the remote access protocol is a key aspect with regard to lossless data compression and smooth transfer of visual information. It must ensure a low-latency user interaction without any disturbing lag effects, as most analysis processes require several hours of work. Currently, there are just a few complete solution suites available offering the provision of virtualized workstations for professional visualization applications:

- Citrix XenDesktop [7]
- Microsoft RemoteFX [8]
- VMware Horizon View [9]

All of them are based on special GPU resources like the NVidia GRID Technology [10]. The allocation of the GPU resources is possible via a special software layer which allows GPU sharing between several VMs. To take advantage of all possible GPU-based features within a VM, it is usual to grant direct access to a dedicated GPU core via PCI passthrough.

A fast network interconnect between VMs and the data storage systems is another critical factor. The remote processing of large data sets in the order of up to 100 GB via network mount points requires a network bandwidth starting at ~10 Gbit/s. In this regard software virtualized network devices are reaching their limits. Providing near-native performance of high-speed network interconnects like 10 Gbit/s Ethernet or even 56 Gbit/s InfiniBand within VMs requires hardware virtualized approaches like Single Root - I/O Virtualization (SR-IOV) [11]. This PCI specification extension allows a single PCI Express (PCIe) I/O device to appear as multiple, separate devices — called Virtual Functions (VF) — a kind of a "light weighted" virtual PCIe device. Each VF can be allocated to one VM via PCI passthrough.

To ensure the best possible user experience, the provision of the virtualized environments has to be integrated within the IT systems of the corresponding synchrotron institution. Users should be able to authenticate with their existing federated accounts towards a web service providing their data sets and analysis infrastructure. New created VMs should contain the user-specific network mount points for the data set access. Pre-installed proprietary analysis software has to be activated automatically by contacting existing pre-defined license servers. Non-used allocated resources have to be reassigned to other users. In this case a mechanism is needed which "motivates" inactive user to release their unused resources, e.g. shut down and/or delete the VM.

In the next section, we will describe our prototypic implementation based on the basic conditions mentioned above.

IMPLEMENTATION

One goal of the ASTOR project foresees the provision of virtualized workstations for the execution of analysis software like Amira [2]. Our first implementation for this purpose is based on the VMware Horizon View [9] virtualization solution combined with the vSphere v5.5 hypervisor. We use NVidia GRID K2 GPU [10] resources, which are allocated to the VMs via *Virtualized Dedicated Graphics Acceleration (vDGA)* [12] using PCI passthrough. Each VM gets physical access to a dedicated high-end Kepler core with access to 4GB GDDR5 memory. Combined with the PCoIP [13] remote protocol, all important visualization APIs like OpenGL 4.1x or DirectX 11 are supported. Carefree interactive usage without annoying time delays is already possible from home office with a common DSL connection about 16 Mbit/s and better.

Access and processing of data sets should take place on remote network-attached storage systems, which requires a fast network interconnect. Therefore we will provide Infini-

Band support within VMs. Each VM gets direct access to a hardware-virtualized InfiniBand VF provided by an SR-IOV capable InfiniBand Host Channel Adapter. Deployment and management of running VMs and templates is performant enough with a NFS shared storage system accessible via 10 Gbit/s Ethernet.

All VMware resources are managed by our software layer called *Cloud Service Bus*, which was originally developed for the transparent provision of public and private IaaS cloud resources [14]. It utilizes VMware SDK and the Amazon Web Services (AWS) SDK for the control of the corresponding VMs/templates and appends value-added services. A unified SOAP-API provides all common functionalities.

Currently VMware does not provide any intelligent schedulers for initial placement supporting VMs with allocated PCIe devices within a multi-node cluster. Furthermore, allocation and release of PCIe resources during different VM/cluster states has to be triggered manually. Therefore we extended our management software functionality with an initial placement mechanism for VMs with PCIe devices. The placement decision of new created and existing VMs within the cluster is based on free available PCIe, RAM and CPU resources. This is an important feature for an automated VM deployment as cluster load balancing via live-migration of VMs¹ is not possible with direct hardware access to PCIe devices.

Our management software provides further essential features for the integration of IaaS resources into a holistic analysis workflow. Several prepared VM templates are available with different analysis software packages. Users are able to define their specific VM hardware configuration concerning their needs on-demand. Customization mechanisms within the Cloud Service Bus allow to pre-configure new created VMs with corresponding specific network mount points, license server information and more, as available VM templates are cleaned up by all personalized data.

Users can authenticate themselves with their existing Active Directory (AD) account towards the Cloud Service Bus. The complete resource usage and load statistics is monitored for general liability and billing. These informations are also accessible by the users for their personal purposes. It is possible to define billing algorithms based on the CPU, RAM and DISK consumption respectively wall time reservation of a VM. To prevent resource allocation by inactive users, credits can be assigned. In the case of exhausted credits, VMs will be shut down after a warning email.

Based on the available SOAP-API, we developed a cross-platform web service called YVAINE [15], with a special focus on a user friendly and intuitive design. Users should create and manage their resources with a few mouse clicks only. This service is the basis for the ASTOR web portal and will be combined with an overview concerning the users' data sets. The simple choice of an existing data set will create a specific VM with all necessary software packages, network

¹ Also known as VMware vMotion and used by the VMware Distributed Resources Scheduler (DRS) within a multi-note virtualization environment for load balancing.

mounts and required computing resources and provide the corresponding access credentials to the users. Figure 1. illustrates the architecture of our current Analysis-as-a-Service workflow implementation.

Analysis-as-a-Service (AaaS)

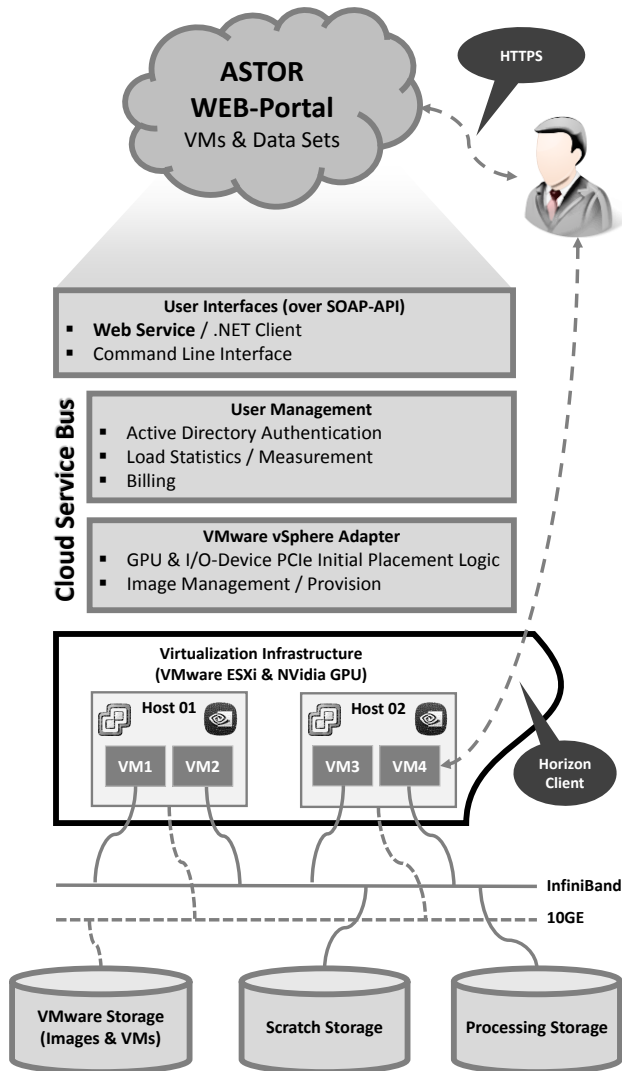


Figure 1: Analysis-as-a-Service Architecture. Users login to the ASTOR web portal, choose their specific data sets and get access credentials to their specific virtual machines for analysis, best suited to their needs.

CONCLUSION & OUTLOOK

The first prototype of the ASTOR "Analysis-as-a-Service" infrastructure is already a significant improvement compared to the traditional strategy with dedicated analysis workstations. Scientific users are flexible concerning their scheduled time and get professional high-end resources and fast access to their large data sets for data analysis at arbitrary locations.

Future plans foresee the support of more federated account solutions like Shibboleth. Furthermore we will improve the initial placement mechanisms based on the past and expected resource consumption of running VMs. The evaluation of alternative storage solutions like noSQL databases is another research aspect within the ASTOR project.

ACKNOWLEDGMENT

We want to thank the Federal Ministry of Education and Research in Germany for the funding of our research within the project ASTOR.

REFERENCES

- [1] ANKA – the Synchrotron Radiation Facility at the Karlsruhe Institute of Technology (KIT)
<http://www.anka.kit.edu/>
- [2] Amira 3D Software for Life Sciences
<http://www.fei.com/software/amira-3d-for-life-sciences/>
- [3] VG Studio MAX
<http://www.volumegraphics.com/produkte/vgstudio-max/>
- [4] OpenGL - The Industry's Foundation for High Performance Graphics
<http://www.opengl.org/>
- [5] CUDA – Parallel Programming and Computing Platform
http://www.nvidia.com/object/cuda_home_new.html
- [6] OpenCL – The open standard for parallel programming of heterogeneous systems
<http://www.khronos.org/opencl/>
- [7] Citrix XenDesktop
<http://www.citrix.com/products/xendesktop/overview.html>
- [8] Microsoft RemoteFX
[http://technet.microsoft.com/en-us/library/ff817578\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/ff817578(v=ws.10).aspx)
- [9] VMware Horizon View
<http://www.vmware.com/products/horizon-view>
- [10] NVidia GRID - Graphics-Accelerated Virtualization
<http://www.nvidia.com/object/grid-technology.html>
- [11] Single Root - I/O Virtualization (SR-IOV)
https://www.pcisig.com/specifications/iov/single_root/
- [12] Graphics Acceleration in View Virtual Desktops
<http://www.vmware.com/files/pdf/techpaper/vmware-horizon-view-graphics-acceleration-deployment.pdf>
- [13] PC over IP remote protocol
<http://www.teradici.com/pcoip-technology>
- [14] B2B- Infrastructure Integration via Cloud Service Bus
iX Kompakt: IT-Management in der Cloud, 04-2012, Pages 48–51, September, 2012
- [15] YVAINE, SCC News 01/2013, Karlsruhe, Germany
<http://www.scc.kit.edu/publikationen/scc-news.php>

MAKING IT ALL WORK FOR OPERATORS

I. Kriznar, Cosylab, Ljubljana, Slovenia
 S. Marsching, Auenos GmbH, Baden-Baden, Germany
 E. Hertle, E. Huttel, W. Mexner, N. J. Smale, A.-S. Mueller,
 KIT, Eggenstein-Leopoldshafen, Germany

Abstract

As the control system of the ANKA synchrotron radiation source at KIT (Karlsruhe Institute of Technology) is being slowly upgraded it can become, at key stages, temporarily a mosaic of old and new panels while the operator learns to move across to the new system. With the development of general purpose tools, and careful planning of both the final and transition GUIs, we have been able to actually simplify the working environment for machine operators. In this paper we will explain concepts, guides and tools in which GUIs for operators are developed and deployed at ANKA.

INTRODUCTION

The machine control system of the synchrotron radiation source ANKA at KIT (Karlsruhe Institute of Technology) is migrating from the ACS CORBA based control system to the Ethernet TCP/IP devices with an EPICS server layer and visualisation by Control System Studio (CSS). This migration is driven by the need to replace ageing hardware. Approximately 500 physical devices, are being gradually replaced (or have their I/O hardware changed) and are integrated to the EPICS/CSS control system. [1]

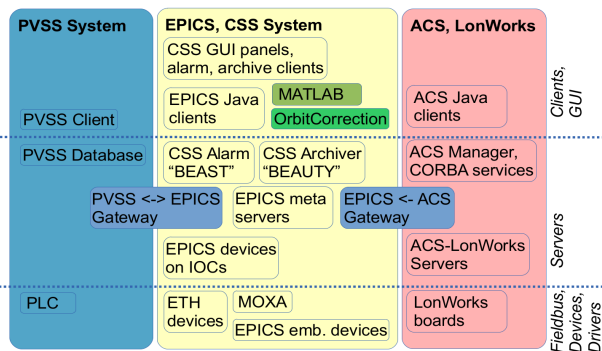


Figure 1: Patchwork of different components and technologies in ANKA control system.

The storage ring is generally operated at an energy of 2.5GeV with a typical beam current of 200 mA and a lifetime of 20 hours. Due to the finite lifetime the storage ring is emptied of electrons twice per day (8:00 and 18:00) and refilled. The refilling process involves a two stage accumulation process. The machine is then left unattended until the next injection time, which leads to two important demands on the control system, namely, robust and sensitive alarm notification in the case of reduced machine performance and a very intuitive GUI machine interface; as a single operator will only attend to the machine for approximately 10 hours per 2 months.

We can already say that the ANKA control system is based on EPICS. All core services and tool-kits are done with EPICS/CSS. However, there are still additionally other control systems, which need to be used in parallel or integrated into EPICS for various reasons. Namely PVSS and ACS (Figure 1).

GENERAL CONVENTIONS

CSS is based on Eclipse RPC and on Java, for which there exist several good references for building functional and pleasant GUI applications [2, 3]. In addition to these extensive documents ANKA has several simple guidelines that address most frequent mistakes, which are encountered:

- Use default look and feel, default fonts and sizes as provided by CSS/Eclipse.
- Design elements must be consistent across all applications and components. Buttons, check-boxes, radio-boxes, combo-boxes have well known and expected functionality; Don't misuse them or change behaviour. Don't change their labels, functionality and position in runtime ether.
- Some colours are reserved, borders are reserved for alarm notifications. It is important to use them only for designated purposes, don't confuse users with design choices that are similar to reserved use.
- Leave just the right space between components, organize them so they are equally spaced and lined to same base-lines. There are tools in CSS visual composition toolbar that help you do just that. This point should be trivial and obvious, but for some reason it is often not followed.

THE ANKA CLIENTS BUNDLE

ANKA has several distribution channels for different generations of control system clients. Distribution channels are kept in Subversion repository. Installation and updating are performed through Subversion operations on target computers. Subversion was chosen for its convenience over competing solutions. Subversion can effectively keep history of changes and it is used primarily for one way distribution, so something like Git would not be fully utilized and would increase complexity for simple update operations.

Subversion distribution channels of control system clients covers: old ACS client distribution, the PVSS distribution and the main ANKA client distribution. The main ANKA client distribution, called ANKA-Clients, is kept in two branches: the main trunk is used for storing stable releases running on all operator's computers. The

“work” branch is intended for development work. Since there are not so many developers, and they work on different devices, the chances for the conflicts are very low. During a shut-down period the development branch is copied/merged to the trunk, tested, and installed on all control system computers through subversion update feature.

THE LAUNCHER

ANKA Launcher is started from within ANKA-Clients distribution bundle and is the preferred way on how to start control system applications or perform certain automated tasks. The ANKA Launcher icon can be found on all operators machines (Figure 2).

Buttons in the launcher can execute scripts on local disk or on a remote server over SSH protocol. If the local computer has XServer installed (like Exceed in Windows), then it can open an application window from a remote Linux computer.

Launcher buttons and their tasks are defined in a launcher configuration file and act as groups of buttons. The following groups are defined:

- Common Clients group: start the ACS Java panels.
- CSS Clients group: opens the CSS Main, Alarm and Archive (Data Browser) applications.
- EPICS Clients group: opens some third party provided EDM EPICS panels from a remote Linux computer, needs local X11 server to work. These are: Libera Launcher panels, Matlab ML panel and BBB Expert panel.
- VNC Servers: button opens a VNC to a server.
- VNC Oscilloscopes: opens VNC to oscilloscopes.
- ACS Server group: starts/stops the central ACS system over SSH connection.
- EPICS IOCs and Servers group: starts/stops EPICS IOCs and servers over SSH connection.

THE CSS APPLICATIONS

There are two distinct type of users of the machine control system and therefore two distinct approaches to the control system GUI applications: machine operators, they need a stable and a predictable environment which helps to go through an optimized procedure which should give a reproducible end result: a stable orbit and beam. And the experts and machine developers: they work with vague procedures and they want to have all options open and tools available, because it is not known in advance what will be needed. Tools or procedures could be developed during work and then perhaps not used again.

Even at the design stage an application developer must decide if an application is intended for operators or experts. In most cases it is even desirable to develop

different panels or sets of panels of the same application or device specifically for these two user groups.

For operation purposes the following is considered the best approach:

- ANKA CSS main application window should be used if possible in full screen mode.
- Restart of CSS instance would always bring back same main screen layout and same panels.
- Sub-panels should be opened within the main screen and not in a separate window. As further specified in the document section below.

In case of expert panels we have additional requirements: expert panels must be equipped with warnings so that a casual operator would not accidentally run them. And dangerous operations, which are not used in day-to-day operation, must be protected with additional confirmation dialogues.

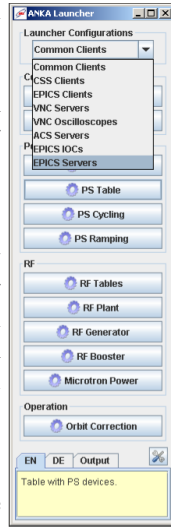


Figure 2: The ANKA Launcher.

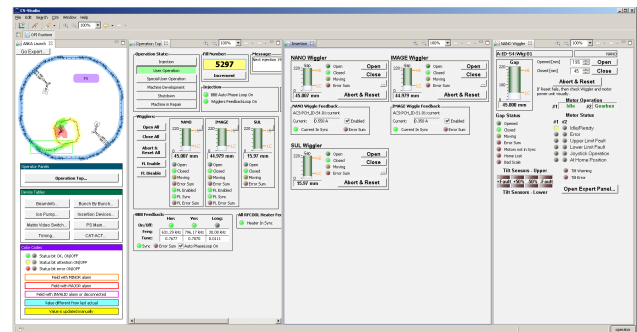


Figure 3: This is main CSS panel at ANKA and demonstrate on a case of wigglers panels the four levels of the panel hierarchy from the left to the right: i) the CSS launcher, ii) the main process/operation panel, iii) the overview device group panel and iv) the device details panel.

There are general guidelines on how to organize various panels within the main ANKA CSS application. They are demonstrated in Figure 3 and are described from left to right.

- **The CSS launcher** is considered the most important panel. It contains action buttons, which open process and overview panels in tabs to the right of the CSS launcher. The CSS launcher should be always visible. It has two modes: operator and expert.
- **Process oriented panels** contain controls for automated and semi-automated processes. Different process stages and procedure can be organized in different sub-tabs. The most distinguished process oriented panel is “top operation” panel, which gathers together all information and operations directly connected to the daily operation. By convention device widgets included (by linking container) in this panel are found in a file with the name “Top.opi”.
- **Overview oriented panels** should, by convention, open right to the process oriented panels. Here are panels which provide a group view for devices and panels for individual devices. An individual device widget is by convention found in a file called

“Overview.opi”, when included in the device group panel. An individual detailed device panel is usually opened from a device group overview panel and the a tab should appear right to the group overview panel. It allows access to the individual operations of the device and provides the most details about the device. Usually operators don't go down to this level, these panels are usually used by particular device experts. But still they should be operator-friendly, so OPI file should be named “Operator.opi” if possible.

- **Expert panels** offer more functionality than what an operator needs in daily operation and they can be safely used only by device experts. The GUI is not expected to be polished up to the same level as an operator's panels. The detailed device expert panel is opened from device panel (with button “Open Expert Panel...”). It contains additional options not presented in Operator.opi, mainly to get them away from operators. Intended for expert use only by device experts. The OPI file for device expert panel is usually named “Expert.opi”.

Generally operators are able and allowed to reorganize tabs within panels as they find more usable. But because this is potentially dangerous and could lead to unwanted results, these changes are reversible. Each time CSS panel is restarted at ANKA its workspace is cleared and replaced with a template so they always appear looking the same way.

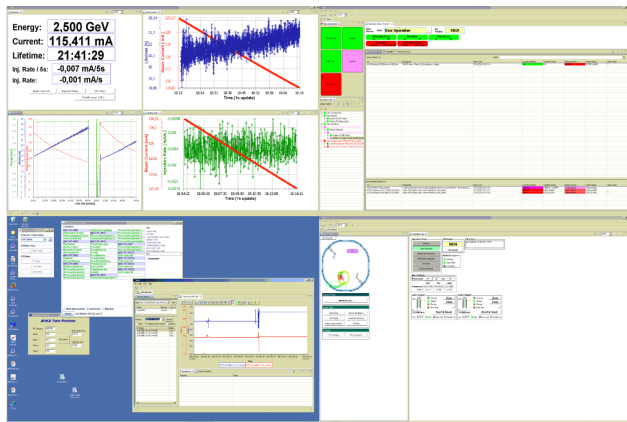


Figure 4: Layout of panels on operator's computer with four screens. From upper left corner clockwise: BeamInfo, Alarm, CSS Main and Archiving with some other control applications.

In the ANKA-Clients distribution bundle there are only three main CSS application short-cuts and several special purpose ones (Figure 4):

- ANKA CSS Main application: here all controls system CSS panels are included and launched within the main panels. Instead of separate windows they are intended to be opened as tabs and rearranged as necessary. For this reason CSS Main application window is used in full-screen mode and is usually the only window on the computer monitor.
- Alarm application: opens BEAST alarm perspective.
- Archiving application: opens the DataBrowser perspective to the Cassandra database.

- BeamInfo application: CSS panel which shows only beam parameter information and trends, interesting during injection and ramping.
- Special panels to be run on wall monitors.

Device Type Convention

Devices, which are of the same type, are represented in the top operator's panels also with the same CSS widgets (by linking container included OPI file) which provide standardized access to the common device values and operations; Even if the actual physical devices might be from different manufacturer and has a quite different set of features. For example, ANKA has several different kinds of power supplies and wigglers, yet they are all operated in the same way from the top operations panel. But of course they might have very different device detail panels, which reflect their actual capabilities.

This is possible by standardisation of a set of common PV names and types. The ANKA EPICS naming convention defines device parts of the PV and leaves the property part free. The ANKA Device Type Convention further defines which are common mandatory PVs and in what way a device must be supported in order to be included into standardized panels and automation tools. Usually these are PVs for the most common operations in a simplified format, from the point of view of operation, and device summary signals, in similar manner as earlier mentioned Status/ErrorSum.

CONCLUSION

The ANKA machine during normal user operation time is refilled twice a day and in the meantime is left unattended. Therefore, a single operator will only work with the machine for approximately 10 hours per 2 months. This puts strong constraints on the control system in that there needs to be a very intuitive GUI machine interface design. New panels and tools must be well designed and carefully introduced into operation in order not to lose trust in the control system. Operators expect to find panels on the same computer organized in the same way so they can easily go through the operation procedure after weeks of absence.

ACKNOWLEDGEMENT

We would like to thank all people who helped in the planning and upgrade of the ANKA control system, in particular Guenther Rehm, from the Diamond light source, for his continuous support. And we would like to mention Cosylab developers, who provide excellent development work and continuous support on EPICS panels and drivers according to our requirements and conventions.

REFERENCES

- [1] N.J. Smale et al., “The ANKA Control System: On a Path to the Future”, MOPPC099, ICALEPCS'13.
- [2] Design guidelines for Eclipse website: http://wiki.eclipse.org/User_Interface_Guidelines
- [3] Java development guidelines website: <http://www.oracle.com/technetwork/java/hig-136467.html>

HOW THE COMETE FRAMEWORK ENABLES THE DEVELOPMENT OF GUI APPLICATIONS CONNECTED TO MULTIPLE DATA SOURCES

R. Girardot, G. Viguiet, K. Saintin, M. Ounsy, A. Buteau
SOLEIL Synchrotron, Gif-sur-Yvette, France

Abstract

Today at SOLEIL, our end users require that GUI applications display data coming from various sources: live data from the Tango control system, archived data stored in the Tango archiving databases and scientific measurement data stored in NeXus/HDF5 files. Moreover they would like to use the same collection of widgets for the different data sources to be accessed.

On the other side, for GUI application developers, the complexity of data source handling had to be hidden. The COMETE framework has been developed to fulfil these allowing GUI developers to build high quality, modular and reusable scientific oriented GUI applications, with consistent look and feel for end users.

COMETE offers some key features to software developers:

- A data connection mechanism to link the widget to the data source
- Smart refreshing service
- Easy-to-use and succinct API
- Components can be implemented in AWT, SWT and SWING flavours.

This paper will present the work organization, the software architecture and design of the whole system. We'll also introduce the COMETE eco-system and the available applications for data visualisation.

CONTEXT

The SOLEIL ICA team is in charge of the control systems and data visualisation and reduction GUI for accelerators and 30 beamlines.

During the first years of SOLEIL construction, ICA team was focused on developing GUI application and TANGO [1] devices for the control systems.

Then the focus was set on providing data storage and management applications for technical and scientific data.

- For the technical data the Tango Archiving service [2] was developed with very demanding requirements on the GUI for data extraction and visualisation. Today a volume of about 10 TB of data coming from more than 30 000 Tango attributes are stored in Oracle databases.
- For scientific data, it was early decided to use the NeXus data format [3] to record measurements and metadata on all our beamlines. SOLEIL beamlines produce daily thousands of experimental NeXus files with sizes ranging from a few MB up to 100 GB.

Moreover, users needed to use uniformed GUI to view their data, whatever the origin.

Control system and supervision GUI applications were first developed using the ATK [4] toolkit which is intimately linked to TANGO.

On the other hand, for technical and scientific data visualisation and reduction, there was no toolkit available to quickly develop this kind of software not TANGO based.

It is in this context that SOLEIL launched the COMETE project [5] to propose a multi data source toolkit to help software engineers to develop applications independently of the data source.

THE COMETE SOLUTION

Architecture

COMETE is a framework composed of three parts:

1. A set of graphical components (widgets) that are completely dissociated from a data source or even a data type.
2. A data source compatible with the graphical component, each corresponding to a data type.
3. Between the widgets and the data source, a mediation layer in charge of adapting and transmitting data.

DataConnectionManagement

The DataConnectionManagement module is a layer that allows connection between two abstract entities, called "Target" and "Data Sources".

This module implements a Mediator pattern (Figure 1), as well as various other patterns such as Strategy, Observer etc. Mediator was chosen instead of MVC pattern because our two entities had to be completely independent from each other, to allow easily adding new widgets and sources.

The sources are produced by factories, which include some data refreshment mechanism.

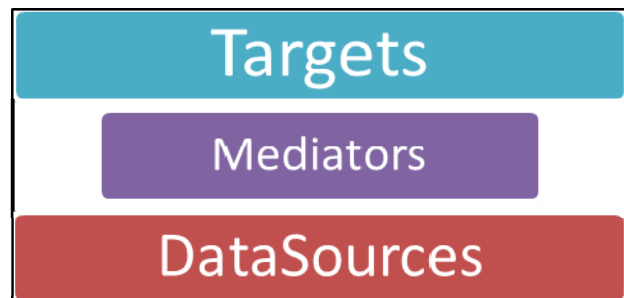


Figure 1: DataConnectionManagement pattern.

Widgets

COMETE Widgets are “**Targets**” specialisation. They intend to be components which comply with the interfaces described in CometeDefinition that makes them connectable to any data source.

Comete Widgets are available in three implementations (Swing, SWT & AWT), see Figure 2, and therefore integrate well into any existing JAVA software. Anyone can use this set of widgets because they are based on these three major toolkits.

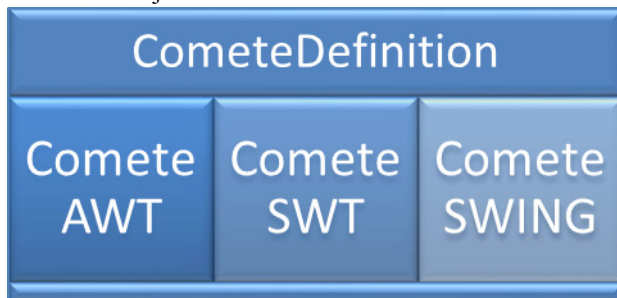


Figure 2: Comete Widget implementations.

Of course the COMETE framework allows adding easily new widgets. The current library of widgets can display:

- Scalar data (textfield, spinner, wheelswitch, slider, etc.)
- Spectrum data (chart viewer)
- Images (image viewer, tables)

Our CometeSwing image viewer (Figure 3) is based on ImageJ [6] which is a popular image processing application. ImageJ is already used by many scientists, so it is very easy for them to use our viewer and they can run their old macros through our Java applications. Since our viewer encapsulates ImageJ, it provides the same image processing features.

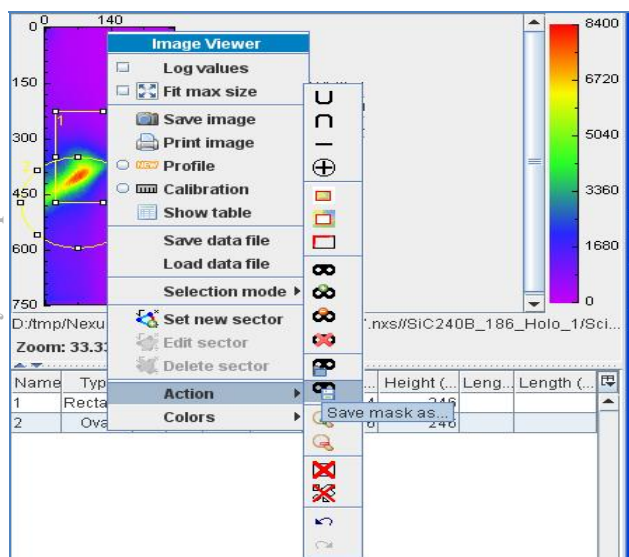


Figure 3: CometeSwing Image Viewer features.

DataSources

When someone wants to add a new data source to Comete, the only thing to do is to implement a class that extends AbstractDataSource. This guarantees that the COMETE mediator will be able to send data to the widget and vice-versa.

Moreover, to manage data refreshment you might need to implement your own IDataSourceFactory class that will instantiate your data sources.

Today the following data sources are available within the COMETE framework:

- A TANGO implementation to access Control System data
- A NeXus implementation to access scientific data from data reduction applications.
- A first SQL implementation to access technical data produced by the Tango Archiving system

CometeBox

The CometeBox module aims to simplify the use of COMETE for developers who intend to use the IDataSourceFactory.

When you want to connect a target to some source produced by an IDataSourceFactory, you have to do following steps:

- Instantiate your target
- Instantiate your IDataSourceFactory
- Instantiate a key
- Ask your IDataSourceFactory to produce your source from this key
- Instantiate your mediator
- Ask your mediator to connect your source to your target

And of course, **this allows accessing only 1 data.**

CometeBox simplifies this access, and **offers the possibility to automatically connect your target to some meta-data** around your source (*for example, a state or quality concerning your source*).

To connect your target to a source and all its meta-data, you have to do following steps:

- Instantiate your target
- Instantiate your CometeBox
- Instantiate a key
- Ask your CometeBox to connect your target to your key

This mechanism is illustrated in the following Use Cases and by the code examples in Figure 5.

Use Cases

A typical use case is to connect to the same target (for example: a chart widget) a Tango or a NeXus data source.

For example, anyone can superpose a spectrum coming from a NeXus file with another one from the TANGO control system (see Figure 4).

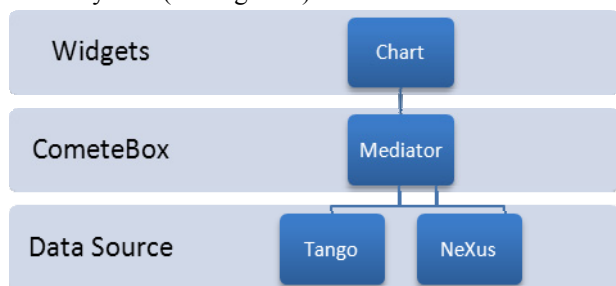


Figure 4: A widget connected to two sources.

This use case can be translated by the following code (Figure 5):

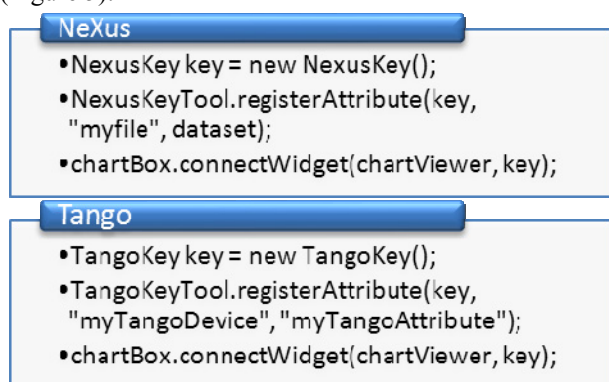


Figure 5: Connect a widget to two data source in 6 lines of code.

Application Examples

Here are two examples of graphical applications based on COMETE and using the same component list with different data sources (Figure 6 & Figure 7).

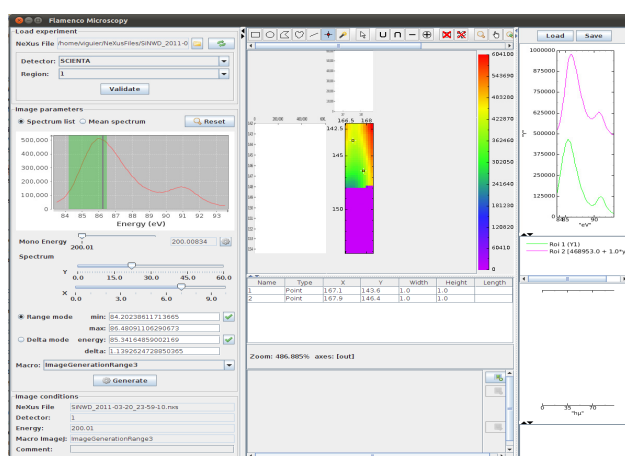


Figure 6: Data reduction application dedicated to microscopy.

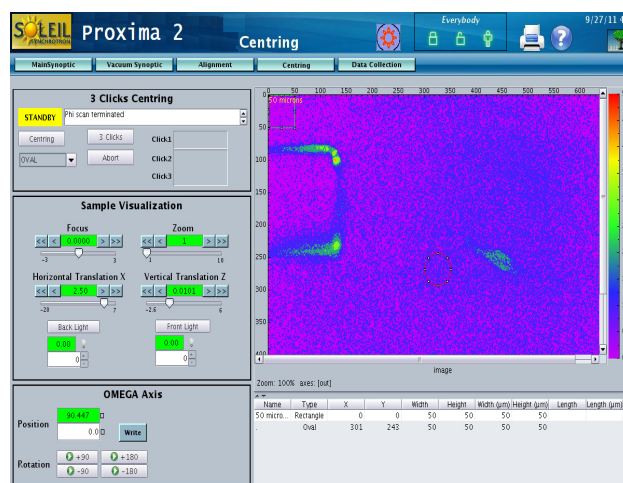


Figure 7: Data visualisation application on PX2 beamline.

CONCLUSION

COMETE is the result of about 10 years of experience on developing data visualisation and reduction applications at SOLEIL. It is now a mature and powerful framework widely and daily used by our developers.

SOLEIL is open to collaborations on the project:

- With potential **contributors** having advanced Java skills to participate to framework extension (new widgets, new data sources, data services, etc...)
- With anyone who would like to **use** existing framework to develop their own application, making feedback on their use and needs.

The easiest way to get in touch with the COMETE team is to contact them at:

comete@synchrotron-soleil.fr

REFERENCES

- [1] Tango Control System
<http://www.tango-controls.org>
- [2] J. Guyot, M. Ounsy, S. Pierre-Joseph Zephir "Status of the TANGO Archiving System", ICALEPS 2007
- [3] NeXus Data format
<http://www.nexusformat.org>
- [4] F. Poncet, J.L. Pons, "Tango Application Toolkit", ICALEPS'05, Geneva, Oct 2005.
- [5] Comete Repository
<http://sourceforge.net/apps/trac/comete/>
- [6] ImageJ: Image processing and analysis in Java
<http://rsb.info.nih.gov/ij/>

PANIC, A SUITE FOR VISUALIZATION, LOGGING AND NOTIFICATION OF INCIDENTS

S. Rubio-Manrique, F. Becheri, G. Cuní, D. Fernandez-Carreiras, C. Pascual-Izarra, Z. Reszela,
CELLS-ALBA Synchrotron, Barcelona, Spain

Abstract

PANIC is a suite of python applications focused on visualization, logging and notification of events occurring in ALBA Synchrotron Control System. Build on top of the PyAlarm Tango Device Server it provides an API and a set of graphic tools to visualize the status of the declared alarms, create new alarm processes and enable notification services like SMS, email, data recording, sound or execution of Tango commands. The user interface provides visual debugging of complex alarm behaviors, that can be declared using single-line python expressions. This article describes the architecture of the PANIC suite, the alarm declaration syntax and the integration of alarm widgets in Taurus user interfaces.

INTRODUCTION

ALBA[1], member of the Tango Collaboration[2], is a third generation Synchrotron lightsource in Barcelona, Europe. It provides synchrotron light since 2012 to users in its 7 beamlines, with 2 more under construction.

PANIC is an Alarm System running on top of the Tango Control System to provide periodic evaluation of user-specified alarm formulas, automatic actions and notification whenever formulas evaluate to True, logging of the control system status when this occurs and later monitoring and supervision of the evolution of the system.

Elements of the PANIC Alarm System have been deployed at ALBA[3] since the start of the construction phase. Developed to provide stand-alone monitoring during the vacuum installation of the accelerators it evolved into a versatile system in which many tools interact to provide not only a monitoring tool, but a supervisor service on top of a Tango Control System.

Other Alarm Systems existed already in Tango. PANIC was inspired on Elettra[4] (C++) and Soleil (Java) alarm systems; but focused on exploiting the versatility of python[5] to process rules on runtime, allowing operators and engineers to develop complex logics in rules[6].

THE PANIC ECOSYSTEM

The PANIC Alarm System is completely integrated in Tango[7]. Although it can work without a Tango Database using files as configuration, it develops its complete functionality when interacting in a complete Tango Control System.

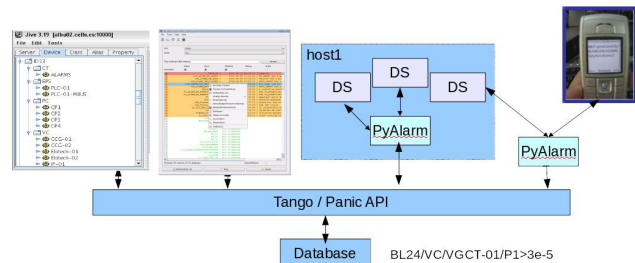


Figure 1: Architecture of the PANIC Alarm System.

The different elements that are part of a PANIC Alarm System (Fig. 1) are:

- PANIC Api: Alarms Database API and rule evaluator. Provides a unique view of the system and coherence between all devices and UI's. It encapsulates all the behavior that is later executed in the User interface or Device Servers.
- PyAlarm Device Server: Tango Device Server that, on top of the PANIC Api, executes alarm rules periodically and trigger the configured actions.
- ProcessProfiler Device Server, a Tango Device to inspect current performance of a linux system, exporting memory and cpu usage of all running processes to be used as source for Alarms.
- Festival Device Server: providing speech and pop-up notification in user terminals.
- PANIC User Interface: manager of the PyAlarm devices, editor of formulas and browser of the Tango Control System from alarms point of view.
- PANIC Tau Toolbar: simple Alarm viewer, restricted to alarms related to attributes shown in a running Tau application.
- Taurus Search Bar: it provides a search engine that indexes relationships between devices, attributes, properties, labels and alias within Tango.
- MySQL Databases: instead of having its own schema PANIC relies in databases of the Tango Control System for configuration (Tango DB) and logging (Tango Archiving DB).
- Snap Viewer: widget that browses the Tango Snapshotting Database (developed by Soleil Institute) where alarm logs and related attribute values are stored.
- Alarm NoSQL database: alternative database for unified configuration and logging, under development by Max IV team.

- Alarm Web Reports, static HTML pages generated by PyAlarm summarizing current status of all variables related to the alarm.

ALARM FORMULAS EVALUATION

Formulas are evaluated by the Panic API using the Fandango[8] library and the `fandango.tango.TangoEval` Class. This object is a singleton for each process and shares cache and all Attribute/Device proxy between all threads.

Alarm formula syntax accepts simple value comparisons, but also list comprehensions, regular expressions and aggregating existing alarms:

- MAX_P: BL24/VC/VGCT-01/P1>3e-5
- ALARM_P: any([q in (ATTR_ALARM,) for q in FIND(BL/VC/VG-*/*P*.quality)])
- ANY_P: MAX_P or ALARM_P

Extended Syntax, Macros

Syntax available in formulas have been extended with some methods called Macros. Those methods provide a pre-parsing of the formula to execute in-place replacement before executing the formula.

Some examples of macros available:

- FIND(regex): that replaces regexp by all attribute names matching the regular expression.
- GROUP(regex) : this allows to group many alarms as a single one, it returns a boolean flag if any matching alarm was activate in the last cycle.
- CACHE[attribute/alarm][-i]: returns access to the previous values of an attribute, keeping a buffer of size `AlarmThreshold+1`.

Extended Tango Attribute Name Syntax

Alarm formulas allow extended attribute names:

some/device/name{/attribute}{.FIELD}

Table 1: Examples of Using Extended Attribute Names

State (when no attribute is given)	BL22/CT/EPS-PLC-01 == FAULT
.value (optional)	BL22/CT/EPS-PLC-01/CC1_AF.value > 1e-5
.time (attribute not updated)	BL22/CT/EPS-PLC-01/CPU_Status.time < (now-60)
.quality (standard tango qualities)	BL22/CT/EPS-PLC-01/OP_WBAT_OH01_01_TC11.quality == ATTR_ALARM
.delta (e.g. for an open valve that just closed)	BL22/CT/EPS-PLC-01/VALVE_11.delta == -1
.exception (matched when the attribute is unreadable)	BL22/CT/EPS-PLC-01/I_Dont_Exist.exception
.all (full attribute struct with all previous fields embedded)	[x.time<now-60 or x.exception for x in FIND(BL/PLC/01/*.*all)]

To provide full Tango functionality with a compact syntax the Alarms allow to express attribute names using extensions (Table 1). Those extra fields provide extra information from the device (even the type of exception when are unreadable) that enable complex alarms.

Those extra fields are also parsed whenever macros and regular expressions are used.

Triggering Automatic Actions from Panic

The syntax of Alarm receivers can be used to execute commands in outer devices, this kind of operations can trigger notification but also control actions. In the Table 2 example, a formula is used to automatically open a front-end whenever protection systems[9] allow it, and simultaneously notify beam-line scientists by pop-up.

Table 2: Example of Automated Front-End Opening Using PANIC. In this example an alarm triggers a PLC command and a pop-up notification in the beamline computers.

Formula	bl/ct/plc-01/FE_AUTO and host:10000/chan/ct/fe/value and bl/ct/plc-01/BL_READY and not bl/ct/plc-01/fe_open and not bl/ct/-plc-01/fe_control_disabled
Receiver 1	ACTION(alarm:attribute,bl/ct/plc-01/OPEN_FE,1)
Receiver 2	ACTION(alarm:command:test/notif/b lmachine/popup,\$ALARM,\$DESCRIPTION,15)

PYALARM DEVICE SERVER

Each PyAlarm Tango Device contains a unique thread that evaluates a set of Alarm formulas written in python. Alarms are stored in the Tango Database as Device Properties: AlarmList for formula, AlarmReceivers for notifications, AlarmSeverities for categorization and AlarmDescriptions containing the message to be shown in notification and UI's.

The PyAlarm thread will keep a continuous polling thread (independent of Tango polling thread) evaluating the set of alarm rules and, depending on the device configuration, triggering notifications and actions when the alarm formula evaluates to True; and later managing Reminder, Reset, Acknowledge actions if the alarm state oscillates, recovers to False or is acknowledged by user.

Notifications by email and file logging are embedded in the PyAlarm device, while SMS, Speech and Popup require external device servers or plugins.

Distributing Alarms between Servers

A single PyAlarm device server can group many devices, but the threading configuration is unique to each device. Grouping the devices by targeted attributes allows to minimize attribute reading, as all formula evaluators share cache and the last value read per attribute. A reasonable balance should be kept to avoid massive

Tango clients with thousands of connections from the same machine.

PyAlarm Device Configuration

Parameters that are independent for each device are controlled by 13 properties:

- AlarmList: each Alarm is unique for the whole Tango Control System, the Panic API keeps consistency between all PyAlarm devices.
- Enabled, PollingPeriod, AlarmThreshold, AutoReset: All this integer properties will control the startup delay of the device, the frequency of alarm evaluation, the number of triggers required to activate the alarm and time after which the alarm will be auto-reset when the condition is not active anymore.
- Reminder, AlertOnRecover, FlagFile, LogFile, HtmlFolder: Those properties will control additional notification and logging during the Activation/Reset cycle.
- CreateNewContexts, UseSnap: enable Alarm and attribute values logging in the Tango Snapshotting database. The list of attributes to be recorded for each alarm is automatically generated from the formula, but can be later modified by user.
- UseTaurus, UseProcess: Control how the access to attributes is done, either using Taurus to choose between polling and events or doing polling thread in a background process to optimize speed.

PANIC GUI APPLICATION

The Panic GUI (Fig. 2) shows the list of active or declared alarms. It provides several filters to search alarms: by state (active/inactive), activation time, severity, subsystem, receiver or historic values.

A text search is also provided that allow to locate alarms by any of the attributes used in formula or words used in description.

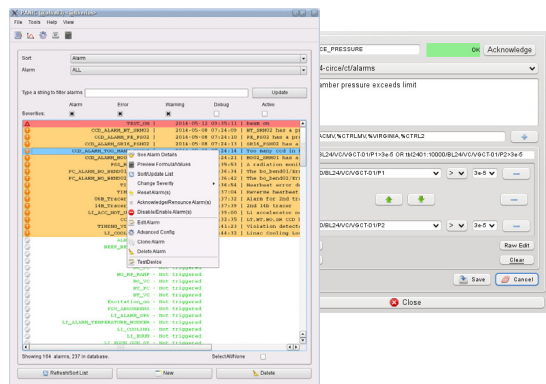


Figure 2: Alarm list and Alarm editor widgets.

For each alarm the menu allows to Reset or Disable the alarm, edit it, modify the PyAlarm device configuration, inspect the logging recorded in the Snapshotting database and access the "Alarm Calculator", a preview panel that

allows to execute any formula using the Alarm syntax to preview the behaviour of the alarm when applied.

CONCLUSION

The latest release of the PANIC Alarm System has evolved into a complete system that is becoming one of the main tool used by our operators and scientists. The strong pillar of the system is the reliability of PyAlarm, in operation for the last 6 years.

All the elements described in this paper are available in the Tango Device Servers repository in Sourceforge; thanks to the Tango collaborative effort now Panic is also used by other Synchrotrons like Max IV. The main focus in the next iterations will be further integration in Taurus and Sardana[10] to achieve certain usage uniformity between the several application involved.

It is still a fact that Tango has three different Alarm Systems instead of a unique tool. It is motivated by the three big families of developers (C++, Java, Python) being naturally attached to their preferred language. But, despite of having different rule-processing engines, actually some contacts have been made to increase the interaction between the three systems to allow notification services, graphical applications and databases to be combinable in the future.

ACKNOWLEDGEMENTS

The ALBA Accelerators division for the help in the debugging and refinement of the PANIC tools. A.Pearsson and A.Milán from Max IV institute in their effort to adapt and extend Panic as a tool for machine scientists.

REFERENCES

- [1] ALBA website: <http://www.cells.es>
- [2] TANGO website: <http://www.tango-controls.org>
- [3] S.Rubio, et al. "Extending Alarm Handling in Tango", ICALEPCS'11, Grenoble, France (2011)
- [4] Lorenzo Pivetta, "Development of the Tango Alarm System", ICALEPCS 2005, Geneva, Switzerland
- [5] D.Fernández et al. "Alba, a Tango based Control System in Python", ICALEPCS'09, Kobe, Japan (2009)
- [6] S.Rubio et al., "Dynamic Attributes and other functional flexibilities of PyTango", ICALEPCS'09, Kobe, Japan (2009)
- [7] A.Götz, E.Taurel et al, "TANGO V8 – Another Turbo Charged Major Release", ICALEPCS'13, San Francisco, USA (2013)
- [8] Fandango website: <http://www.tango-controls.org/Documents/tools/fandango/fandango>
- [9] S.Rubio, et al. "PyPLC, A VERSATILE PLC-TO-PC PYTHON INTERFACE", PCaPAC'14, Karlsruhe, Germany (2014)
- [10] T.Coutinho et al., "Sardana, The Software for Building SCADAS in Scientific Environments", ICALEPCS'11. Grenoble, France (2011)

List of Authors

Bold papercodes indicate primary authors

— A —		
Abe, T.	WC0205	
Agrawal, R.K.	TC0102, TC0202, FP0004	
Amand, F.	TC0103	
Ameil, F.	TC0201	
Antoniuzzi, L.	WP0017	
Antoniotti, F.	WP0030	
Aytac, S.	FP0029	
— B —		
Bacher, R.	FC0201	
Bär, R.	WP0004, WPI01, TC0201, TC0304, FP0017, FP0022, FP0024, FPI03	
Bai, J.N.	TC0304, FP0024	
Balzer, B.M.	FP0002, FPI02	
Bansal, A.	TC0202	
Bao, Y.W.	WP0025	
Barpande, K.G.	TC0202	
Bartkiewicz, P.K.	FP0031	
Bassato, G.	WP0018	
Becheri, F.	FC0206	
Beck, D.H.	WP0004, WPI01, TC0301, TC0304, FP0022, FP0024, FPI03	
Bellato, M.A.	WP0016, FP0014	
Bellorini, F.	WP0030	
Betz, C.	WP0004, WP0007, WPI01	
Bisegni, C.	TC0204	
Blanchard, S.	WP0030	
Bobnar, J.	TC0207	
Boeckmann, T.	WC0203	
Boivin, J-P.	WP0030	
Bonn, M.	FC0202	
Bräuning, H.	WP0006	
Brandl, G.	FP0027	
Brosi, M.	FP0002, FPI02	
Buteau, A.	FC0204	
— C —		
Caselle, C.M.	WC0201, FP0002, FPI02	
Catani, L.	TC0204	
Cerff, K.	FP0018	
Chang, Y.-T.	WP0033, WP0034	
Chauhan, A.	TC0202, FP0004	
Chen, J.	WP0033, FP0030, FPI05	
Chen, Z.C.	FP0013	
Cheng, Y.-S.	WP0033, WP0034, WP0035, FP0030, FP0032, FPI05	
Chevtsov, P.	WP0031, WP0032	
Chilingaryan, S.A.	WC0201, FP0001, FP0002, FP0026, FPI01, FPI02,	
Chiu, P.C.	FC0202	
	WP0033, WP0035, FP0030, FPI05	
Ciuffetti, P.	TC0204	
Clausen, M.R.	WC0203, FC0205	
Cleva, S.	WC0103	
Contran, M.	WP0016	
Coutinho, T.M.	WC0206	
Cox, G.	WP0039, WPI05	
Cuní, G.	WC0206, FP0011, FC0206	
— D —		
Dabain, Y.S.	WP0024	
Davies, J.	FP0022, FPI03	
Deghaye, S.	WP0006	
Dhammatong, Ch.	FP0028, FPI04	
Di Pirro, G.	TC0204	
Dong, J.M.	WP0026	
Dong, K.J.	WP0025	
Dritschler, T.	WC0201, FP0001, FP0002, FPI01, FPI02	
Duellmann, D.	FPI01, FPI02	
Duval, P.	FPI01, FPI02	
— E —		
Ehrlichmann, H.	FPI01, FPI02	
— F —		
Falcón Torres, C.M.	WC0206	
Fantinel, S.	FP0014	
Faragó, T.	WC0201, FP0001, FPI01	
Fatnani, P.	TC0102, TC0202, FP0004	
Faulhaber, E.	FP0027	
Felder, C.	FP0027	
Fernandez-Carreiras, D.	WC0206, FP0011, FC0206	
Ferrand, T.	FP0024	
Fiedler, S.	WP0011, WP0012, WPI02	
Fitzek, J.	WP0004, WP0005, WPI01, TC0101	
Foggetta, L.G.	TC0204	
Frommberger, F.	WP0003	
Fukui, T.	WC0204, TC0205	
Furukawa, K.	TC0206, FP0007	
Furukawa, Y.	WC0205	
— G —		
Gagliardi, F.	WI01, FPI01	
Galletti, F.	TC0204	
Gama, J.	WP0030	
Gangopadhyay, S.	TC0202	
Gargana, R.	TC0204	

Gaspersic, G.G. WP0029
 Geng, H.L. FP0013
 Giacchini, M.G. WP0016, WP0017, WP0018, FP0014
 Girardot, R. FC0204
 Glass, B.R. TC0306
 Gobin, R. FP0019
 Gomes, P. WP0030
 Gothwal, P. TC0202
 Grout, V. FP0022, FPI03
 Gryczan, G. TIOM02
 Guijarro, M. WP0002
 Guilloud, C. WP0002
 Gupta, A.M. TC0202

— H —

Haas, D. FP0018, FP0026
 Hackler, T. WP0007
 Hahn, A. TC0304
 Hancock, M.D. WP0038
 Harder, D.A. WC0207
 Hatje, J. WC0203
 Hatsui, T. WC0205
 He, P. WC0207
 Hertle, E. WP0028, FC0203
 Hiller, N. WP0013, WPI03
 Hillert, W. WP0003
 Hintjens, P. TIOA01
 Hsu, K.T. WP0033, WP0034, WP0035, FP0030, FP0032, FPI05
 Hsu, S.Y. WP0033
 Hu, K.H. WP0033, WP0035, FP0032
 Hu, S. WP0001
 Hu, Y.M. WP0025
 Huang, C.H. WP0033, WP0034, FP0030, FPI05
 Hütter, H.C. WP0005, TC0101
 Huhmann, R. WP0029
 Huttel, E. WP0028, FC0203

— I —

Iitsuka, T. FP0006
 Ikeda, H. FP0016
 Ischebeck, R. WP0031
 Ishii, M. WC0204
 Ismail, A. WP0023, WP0024
 Iwasaki, Y. WP0022

— J —

Jakel, J. FP0018
 Janardhan, M. TC0202
 Jejkal, T. FP0034
 Joti, Y. WC0205
 Jover-Mañas, G. WC0206
 Joyce, M.E. FC0106

Judin, V. WP0013, WPI03, FP0002, FPI02
 Jülicher, S. WP0004, WPI01

— K —

Kago, M. WC0204
 Kameshima, T.K. WC0205
 Kamikubota, N. FP0006
 Kan, C.X. WP0025
 Kato, Y. FP0016
 Keesee, M. FC0106
 Kikuzawa, N. FP0016
 Killenberg, M. WC0101
 Kitegi, C.A. WC0207
 Kling, A. FP0031
 Klinkhieo, S. FP0028, FPI04
 Klor, J. WC0206
 Klysubun, P. FP0028, FPI04
 Knaster, J. FP0019
 Kojima, T. FP0019
 Kolozhvari, A. WP0011, WP0012, WPI02
 Komel, M. FP0015
 Kopmann, A. WC0201, WC0202, FP0001, FP0002, FP0026, FPI01, FPI02, FC0202
 Kopylov, L. WP0030
 Korth, O. WC0203
 Kosuge, T. WP0019, WP0020, WPI04
 Kozak, T. WC0101
 Krause, U. WP0004, WPI01
 Kreider, M. TC0304, FP0022, FPI03, TC0301, FP0024
 Krepp, S. WP0029
 KriÅ¼nar, I. WP0028, FC0203
 Krüger, J. FP0027
 Kruk, G. TC0101
 Kubarev, V.V. WP0027
 Kudou, T. TC0206, FP0007
 Kumar, D. WP0029
 Kuo, C.H. WP0033, WP0034, WP0035, FP0030, FP0032, FPI05
 Kusano, S. TC0206, FP0007

— L —

Lai, L.W. FP0013
 Larrieu, T. L. FC0106
 Lee, D. WP0033, FP0032
 Leng, Y.B. FP0013
 Lenz, A. FP0027
 Li, C. FP0008, FP0009
 Li, K.N. WP0025
 Li, M. WP0026
 Liao, C.Y. WP0033, FP0030, FP0032, FPI05
 Lilienthal, C. TIOM02
 Liu, B.J. WP0018

Liu, G. **FP0008**, **FP0009**
Luo, H. **WP0026**

— M —

Mansouri Sharifabad, M. **WP0023**, **WP0024**
Mao, R.S. **WP0026**
Marqueta Barbero, A. **FP0019**
Marsching, S. **WC0101**, **WP0028**, **FC0107**,
FC0203, **WP0013**, **WPI03**
Martlew, B.G. **WP0038**, **WP0039**, **WPI05**
Maruyama, T. **WC0204**
Maslov, P.A. **TC0303**, **FP0015**
Matias, E. D. **WP0001**
Matsumoto, T. **WC0205**
Matsushita, T. **TC0205**
Matthies, S. **WP0006**
Mauch, V. **WC0202**, **FC0202**
Mazzitelli, G. **TC0204**
Mehle, M. **WC0101**
Mein, C. **FC0205**
Merh, B.N. **TC0202**, **FP0004**
Merker, S. **WP0030**
Mexner, W. **WC0202**, **WP0028**, **FP0026**,
FP0034, **FC0202**, **FC0203**
Meyer, K.A. **TC0303**
Meykopff, S.M. **WP0009**, **WP0010**
Michelotti, A. **TC0204**
Mikheev, M.S. **WP0030**
Mishra, R. **TC0202**
Miyahara, F. **TC0206**, **FP0007**
Momper, E. **WP0007**
Montis, M. **WP0016**, **WP0017**, **WP0018**,
FP0014
Müller, A.-S. **WP0013**, **WP0028**, **WPI03**,
FP0002, **FPI02**, **FC0203**
Müller, R. **WP0005**, **TC0101**
Musardo, M. **WC0207**

— N —

Nagatani, Y. **WP0019**, **WP0020**, **WPI04**
Narita, T. **FP0019**
Nasse, M.J. **WP0013**, **WPI03**
Navathe, C.P. **TC0202**, **FP0004**
Nishiyama, K. **FP0019**

— O —

Oates, A. **WP0039**, **WPI05**
Ohshima, T. **WC0204**
Okada, K. **WC0205**
Okumura, Y. **FP0019**
Ondreka, D. **WP0005**, **FP0024**
Ouchi, N. **FP0016**
Ounsy, M. **FC0204**

— P —

Pascual-Izarra, C. **WC0206**, **FC0206**
Pawlowski, B. **FP0031**
Pedersen, B. **FP0027**
Penning, J. **WC0203**, **FC0205**
Pereira, H.F. **WP0030**
Perez, M. **WP0002**
Peters, C.E. **TC0305**
Petrosyan, L.M. **WC0101**
Petzold, L. **FP0002**, **FPI02**
Pigny, G. **WP0030**
Piotrowski, A. **WC0101**
Pleško, M. **WP0029**, **TC0103**
Poggi, M. **WP0018**
Power, M.A. **TC0305**
Prados, C. **TC0304**, **FP0024**
Prędkie, P. **WC0101**
Preecha, C.P. **FP0028**, **FPI04**
Proft, D. **WP0003**

— R —

Raasch, J. **FP0002**, **FPI02**
Rapp, V. **WC0102**
Rauch, S. **TC0304**, **FP0017**, **FP0024**
Ressmann, D. **WC0202**, **FP0026**, **FP0034**,
FC0202
Reszela, Z. **WC0206**, **FP0011**, **FC0206**
Rickens, H.R. **WC0203**
Rio, B. **WP0030**
Ristau, U. **WP0011**, **WP0012**, **WPI02**
Rosanes Siscart, M. **WC0206**
Rota, L. **FP0002**, **FPI02**
Rubio, A. **FP0011**
Rubio-Manrique, S. **FP0011**, **FC0206**
Ruprecht, R. **WP0013**, **WPI03**

— S —

Sahoo, G.K. **FP0031**
Saifee, K. **TC0202**, **FP0004**
Saintin, K.S. **FC0204**
Sakaki, H. **FP0019**
Saleh, I. **WP0023**, **WP0024**
Sanchez Valdepenas, D. **WP0007**
Satoh, M. **TC0206**, **FP0007**
Scafuri, C. **WC0103**
Schmidt, Ch. **WC0101**
Schmitt, M. **FP0018**
Schoeneburg, B. **WC0203**
Schuh, M. **WP0013**, **WPI03**
Schweizer, C.S. **WP0007**
Schwinn, A. **WP0006**
Seema, M. **TC0202**
Serednyakov, S.S. **FP0010**, **WP0027**
Sheth, Y.M. **TC0202**, **FP0004**
Siegel, M. **FP0002**, **FPI02**

Simon, H. WP0007
 Sliwinski, W. WC0102
 Slominski, C.J. FC0106
 Smale, N.J. WP0028, FP0002, FPI02, FC0203
 Srivastava, B.S.K. TC0102, TC0202
 Stecchi, A. TC0204
 Steinmann, J.L. FP0002, FPI02
 Stern, M. WP0007
 Stevanovic, U. WC0201
 Stotzka, R. FP0034
 Su, S.Y. WP0025
 Sugimoto, T. WC0205, TC0205
 Suradet, N. FP0028, FPI04
 Susnik, T. WC0101
 Suwada, T. TC0206, FP0007

— T —

Takahashi, H. FP0019
 Tanabe, T. WC0207
 Tanaka, R. WC0205, TC0205
 Tanner, R. WP0001
 Terpstra, W.W. TC0301, TC0304, FP0022, FP0024, FPI03
 Thieme, M. WP0004, WPI01, FP0017
 Turner, D.L. FC0106

— V —

Vestergard, H. WP0030
 Viguier, G. FC0204
 Vincelli, R. WP0004, WPI01
 Vogelgesang, M. WC0201, FP0001, FP0002, FPI01, FPI02
 Vondrous, A. WC0202, FP0034
 Vranković, V. WP0032

— W —

Wang, C.-J. WP0033
 Wang, J.G. FP0008, FP0009

Wang, M. WP0026
 Wang, X.F. WP0025
 Weber, M. FP0002, FPI02
 Wiebel, M. WP0008
 Wilgen, J. WP0010
 Wilson, J.T.G. FP0012
 Wright, G. WP0001
 Wu, C.Y. WP0033, FP0030, FP0032, FPI05
 Wu, H. TC0207
 Wuensch, S. FP0002, FPI02
 Wychowaniak, J. WC0101

— X —

Xuan, K. FP0008, FP0009

— Y —

Yabashi, M. WC0205
 Yadav, R. TC0202
 Yamada, S. WP0021
 Yamaga, M. WC0205
 Yamamoto, N. FP0006
 Yan, Y.B. FP0013
 Yang, K. FP0008
 Yang, T. WP0025
 Yoshida, S.Y. FP0006
 Yoshii, A. FP0016
 Yoshioka, M. WC0204
 You, Q.B. WP0025
 Yu, Y.S. WP0025

— Z —

Zaera-Sanz, M. WP0007
 Žagar, K. WC0101, TC0303, FP0015
 Zhao, T.C. WP0026
 Zheng, K. FP0008
 Zweig, M. TC0304, FP0024

Institutes List

ANL

Argonne, Illinois, USA

- Peters, C.E.
- Power, M.A.

Aquenos GmbH

Baden-Baden, Germany

- Marsching, S.

BINP SB RAS

Novosibirsk, Russia

- Kubarev, V.V.
- Serednyakov, S.S.

BNL

Upton, Long Island, New York, USA

- Harder, D.A.
- He, P.
- Kitegi, C.A.
- Musardo, M.
- Tanabe, T.

BSC

Barcelona, Spain

- Gagliardi, F.

CEA/IRFU

Gif-sur-Yvette, France

- Gobin, R.

CELLS-ALBA Synchrotron

Cerdanyola del Vallès, Spain

- Becheri, F.
- Cuní, G.
- Falcón Torres, C.M.
- Fernandez-Carreiras, D.
- Jover-Mañas, G.
- Klorá, J.
- Pascual-Izarra, C.
- Reszela, Z.
- Rosanes Siscart, M.
- Rubio, A.
- Rubio-Manrique, S.

CERN

Geneva, Switzerland

- Antoniotti, F.
- Bellorini, F.
- Blanchard, S.
- Boivin, J-P.
- Deghaye, S.
- Duellmann, D.
- Gama, J.

- Gomes, P.
- Kruk, G.
- Pereira, H.F.
- Pigny, G.
- Rio, B.
- Sliwinski, W.
- Vestergard, H.

CIAE

Beijing, People's Republic of China

- Bao, Y.W.
- Dong, K.J.
- Hu, Y.M.
- Kan, C.X.
- Li, K.N.
- Liu, B.J.
- Su, S.Y.
- Wang, X.F.
- Yang, T.
- You, Q.B.
- Yu, Y.S.

CLS

Saskatoon, Saskatchewan, Canada

- Hu, S.
- Tanner, R.
- Wright, G.

Consorzio Laboratorio Nicola Cabibbo

Frascati, Italy

- Gargana, R.
- Michelotti, A.

Cosylab

Ljubljana, Slovenia

- Amand, F.
- Bobnar, J.
- Gaspersic, G.G.
- Komel, M.
- Križnar, I.
- Kumar, D.
- Maslov, P.A.
- Mehle, M.
- Meyer, K.A.
- Pleško, M.
- Susnik, T.
- Žagar, K.

DESY

Hamburg, Germany

- Aytac, S.
- Bacher, R.
- Bartkiewicz, P.K.
- Boeckmann, T.
- Clausen, M.R.

- Duval, P.
- Ehrlichmann, H.
- Hatje, J.
- Killenberg, M.
- Kling, A.
- Korth, O.
- Meykopff, S.M.
- Pawlowski, B.
- Penning, J.
- Petrosyan, L.M.
- Rickens, H.R.
- Sahoo, G.K.
- Schmidt, Ch.
- Schoeneburg, B.
- Wilgen, J.
- Wu, H.

Elettra-Sincrotrone Trieste S.C.p.A.

Basovizza, Italy

- Cleva, S.
- Scafuri, C.

ELSA

Bonn, Germany

- Frommberger, F.
- Hillert, W.
- Proft, D.

EMBL

Hamburg, Germany

- Fiedler, S.
- Kolozhvari, A.
- Ristau, U.

ESRF

Grenoble, France

- Coutinho, T.M.
- Guijarro, M.
- Guilloud, C.
- Perez, M.

FastLogic Sp. z o.o.

Łódź, Poland

- Piotrowski, A.

Glyndŵr University

Wrexham, United Kingdom

- Davies, J.
- Grout, V.
- Kreider, M.

GSI

Darmstadt, Germany

- Ameil, F.
- Bär, R.
- Beck, D.H.

- Betz, C.
- Bräuning, H.
- Ferrand, T.
- Fitzek, J.
- Hackler, T.
- Hahn, A.
- Huhmann, R.
- Hüther, H.C.
- Jülicher, S.
- Krause, U.
- Kreider, M.
- Krepp, S.
- Matthies, S.
- Momper, E.
- Müller, R.
- Ondreka, D.
- Prados, C.
- Rapp, V.
- Rauch, S.
- Sanchez Valdepenas, D.
- Schweizer, C.S.
- Schwinn, A.
- Simon, H.
- Stern, M.
- Terpstra, W.W.
- Thieme, M.
- Vincelli, R.
- Wiebel, M.
- Zaera-Sanz, M.
- Zweig, M.

IAP

Frankfurt am Main, Germany

- Bai, J.N.

IFMIF/EVEDA

Rokkasho, Japan

- Knaster, J.
- Marqueta Barbero, A.
- Okumura, Y.

IHEP

Moscow Region, Russia

- Kopylov, L.
- Merker, S.
- Mikheev, M.S.

imatix

Brussels, Belgium

- Hintjens, P.

IMP

Lanzhou, People's Republic of China

- Dong, J.M.
- Li, M.
- Luo, H.
- Mao, R.S.

- Wang, M.
- Zhao, T.C.

INFN- Sez. di Padova

Padova, Italy

- Bellato, M.A.

INFN-Roma II

Roma, Italy

- Catani, L.

INFN/LNF

Frascati (Roma), Italy

- Bisegni, C.
- Ciuffetti, P.
- Di Pirro, G.
- Foggetta, L.G.
- Galletti, F.
- Mazzitelli, G.
- Stecchi, A.

INFN/LNL

Legnaro (PD), Italy

- Antoniazzi, L.
- Bassato, G.
- Contran, M.
- Fantinel, S.
- Giacchini, M.G.
- Montis, M.
- Poggi, M.

J-PARC, KEK & JAEA

Ibaraki-ken, Japan

- Yamada, S.

JAEA/J-PARC

Tokai-Mura, Naka-Gun, Ibaraki-Ken, Japan

- Kato, Y.
- Kikuzawa, N.
- Yoshii, A.

JAEA

Ibaraki-ken, Japan

- Ikeda, H.
- Kojima, T.
- Narita, T.
- Ouchi, N.
- Sakaki, H.
- Takahashi, H.

**Japan Atomic Energy Agency (JAEA), International Fusion
Energy Research Center (IFERC)**

Rokkasho, Kamikita, Aomori, Japan

- Nishiyama, K.

JASRI/SPRING-8

Hyogo-ken, Japan

- Abe, T.
- Furukawa, Y.
- Hatsui, T.
- Ishii, M.
- Joti, Y.
- Kago, M.
- Kameshima, T.K.
- Matsumoto, T.
- Matsushita, T.
- Okada, K.
- Sugimoto, T.
- Tanaka, R.
- Yamaga, M.

JLab

Newport News, Virginia, USA

- Joyce, M.E.
- Keesee, M.
- Larrieu, T. L.
- Slominski, C.J.
- Turner, D.L.

Kanto Information Service (KIS), Accelerator Group

Ibaraki, Japan

- Iitsuka, T.
- Yoshida, S.Y.

KEK

Ibaraki, Japan

- Furukawa, K.
- Kamikubota, N.
- Kosuge, T.
- Miyahara, F.
- Nagatani, Y.
- Satoh, M.
- Suwada, T.
- Yamamoto, N.

KIT

Karlsruhe, Germany

- Balzer, B.M.
- Bonn, M.
- Brosi, M.
- Caselle, C.M.
- Cerff, K.
- Chilingaryan, S.A.
- Dritschler, T.
- Faragó, T.
- Haas, D.
- Hertle, E.
- Hiller, N.
- Huttel, E.
- Jakel, J.
- Jejkal, T.
- Judin, V.
- Kopmann, A.
- Marsching, S.
- Mauch, V.

- Mexner, W.
- Müller, A.-S.
- Nasse, M.J.
- Petzold, L.
- Raasch, J.
- Ressmann, D.
- Rota, L.
- Ruprecht, R.
- Schmitt, M.
- Schuh, M.
- Siegel, M.
- Smale, N.J.
- Steinmann, J.L.
- Stevanovic, U.
- Stotzka, R.
- Vogelgesang, M.
- Vondrous, A.
- Weber, M.
- Wuensch, S.

Mighty Oaks

Victoria, BC, Canada

- Matias, E. D.

Mitsubishi Electric System & Service Co., Ltd

Tsukuba, Japan

- Kudou, T.
- Kusano, S.

MLZ

Garching, Germany

- Brandl, G.
- Faulhaber, E.
- Felder, C.
- Krüger, J.
- Lenz, A.
- Pedersen, B.

National Instruments China

Shanghai, People's Republic of China

- Yang, K.
- Zheng, K.

National Instruments

Austin, Texas, USA

- Glass, B.R.

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

- Serednyakov, S.S.

PSI

Villigen PSI, Switzerland

- Chevtsov, P.
- Ischebeck, R.
- Vranković, V.

Raja Ramanna Centre For Advanced Technology

Indore, India

- Bansal, A.
- Fatnani, P.

RIKEN SPring-8 Center, Innovative Light Sources Division

Hyogo, Japan

- Fukui, T.

RIKEN SPring-8 Center

Sayo-cho, Sayo-gun, Hyogo, Japan

- Ohshima, T.
- Yabashi, M.

RIKEN/SPring-8

Hyogo, Japan

- Maruyama, T.

RRCAT

Indore (M.P.), India

- Agrawal, R.K.
- Barpande, K.G.
- Chauhan, A.
- Fatnani, P.
- Gangopadhyay, S.
- Gothwal, P.
- Gupta, A.M.
- Janardhan, M.
- Merh, B.N.
- Mishra, R.
- Navathe, C.P.
- Saifee, K.
- Seema, M.
- Sheth, Y.M.
- Srivastava, B.S.K.
- Yadav, R.

SAGA

Tosu, Japan

- Iwasaki, Y.

SESAME

Allan, Jordan

- Dabain, Y.S.
- Ismail, A.
- Mansouri Sharifabad, M.
- Saleh, I.

USTC/NSRL

Hefei, Anhui, People's Republic of China

- Li, C.
- Liu, G.
- Wang, J.G.
- Xuan, K.

SES

Hyogo-pref., Japan

- Yoshioka, M.

WPS

Hamburg, Germany

- Gryczan, G.
- Lilienthal, C.

SINAP

Shanghai, People's Republic of China

- Leng, Y.B.
- Yan, Y.B.

SLRI

Nakhon Ratchasima, Thailand

- Dhammatong, Ch.
- Klinkhieo, S.
- Klysubun, P.
- Preecha, C.P.
- Suradet, N.

SOLEIL

Gif-sur-Yvette, France

- Buteau, A.
- Girardot, R.
- Ounsy, M.
- Saintin, K.S.
- Viguier, G.

SSRF

Shanghai, People's Republic of China

- Chen, Z.C.
- Geng, H.L.
- Lai, L.W.

STFC/DL

Daresbury, Warrington, Cheshire, United Kingdom

- Cox, G.
- Hancock, M.D.
- Martlew, B.G.
- Oates, A.
- Wilson, J.T.G.

TUL-DMCS

Łódź, Poland

- Kozak, T.
- Prędkie, P.
- Wychowaniak, J.

University of Hamburg

Hamburg, Germany

- Mein, C.