ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-WE012

THE TECHNICAL DESIGN CONCEPT OF THE NEW ACCELERATOR CONTROL SYSTEM FOR PETRA IV

R. Bacher, T. Delfs, T. Tempel, T. Wilksen
Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

Abstract

At DESY, a technical design report is currently being prepared for the upgrade of the PETRAIII synchrotron light source to PETRAIV, a fourth-generation low-emittance machine. Within the framework of this planned project, the accelerator control system is also to be renewed and adapted to the growing user requirements. This concerns on the one hand the core components of the control system itself, but also the hardware interfaces, the technical and beam physics control applications, the data acquisition and archiving systems, and the entire supporting IT infrastructure. The paper reports on the details of the proposed technical design concept.

INTRODUCTION

With PETRA III, DESY operates one of the best storage ring X-ray radiation sources in the world. PETRA III is a 2300-metre-long storage ring feeding 24 user beamlines. It is operated either in brightness mode (480 equally distributed bunches, 120 mA stored beam) or in timing mode (40 equally distributed bunches, 100 mA stored beam), latter a unique feature of PETRAIII. Research groups from all over the world use the particularly brilliant, intense X-ray light for a variety of experiments - from medical to materials research.

DESY plans to expand it into an ultimate, high-resolution 3D X-ray microscope for chemical and physical processes. PETRA IV [1] will extend the X-ray view to all length scales, from the atom size to millimetres. Researchers can thus analyse processes inside a catalyst, a battery or a microchip under realistic operating conditions and specifically tailor materials with nanostructures. PETRA IV offers outstanding possibilities and optimal experimental conditions for industry. Special emphasis is also placed on the sustainable operation of the facility.

PETRA IV will replace the PETRA III facility and will be housed by the existing PETRA III buildings. An additional experimental hall will provide space for additional 18 user beamlines. In addition, a new synchrotron (DESY IV) will serve as booster between the existing electron source LINAC II and PETRA IV. Recently, research has begun on the development and construction of a 6 GeV laser plasma injector.

In 2020, a preparatory phase for the future project PETRA IV was initiated with the aim of submitting a Technical Design Report by mid-2023. Construction work is scheduled to begin in early 2027, followed by a commissioning phase in 2029.

The following chapter will describe the technical design concept of the accelerator control system of the future PETRA IV facility.

TECHNICAL DESIGN CONCEPT

The development and implementation of the future PETRA IV accelerator control system will be embedded in a long-term process to consolidate and simplify the whole accelerator control system landscape at DESY and to take advantage of synergies between the accelerator facilities operated by DESY. Support and maintenance of the existing control system framework used at PETRA III will not be continued beyond its expected lifetime. Central control system services such as data acquisition and archiving or configuration management are to be renewed and prepared for future requirements.

The accelerator control system of PETRA IV will closely follow the control system concept implemented at the European XFEL which is pulsed linear accelerator and free-electron laser. Therefore, the control system concept of PETRAIV will be adapted where necessary to the special needs of a storage ring X-ray radiation source.

Control System Framework

The Distributed Object-Oriented Control System (DOOCS) [2] will form the basis of the future control system of PETRA IV (Fig. 1). DOOCS is the established control system framework at FLASH, European XFEL and other conventional accelerator facilities operated by DESY, as well as advanced accelerator projects based on plasma wake field acceleration. Design features of DOOCS include:

- DOOCS follows the object-oriented design paradigm. Devices and data are objects. The basic entity is a device server representing some control system hardware or logic.
- DOOCS is based on a distributed client-server architecture. Each control system parameter is made accessible via network calls. Its transportation layer is currently based on the standardized, industrial RPC protocol with XDR data representation. Integration of the 0MQ protocol is in progress.
- The DOOCS server core library and the server API are written in C++. A variety of fieldbus and hardware interfaces are supported via device classes. These are accessible through additional libraries which can be linked as needed individually to the server core library.
- A fine-grained mechanism for access authorization as well as monitoring and extensive logging functions are provided.
- Libraries for creating client applications in C++, Java, Python or MATLAB are available either as a separate implementation or as C-bindings.

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-WE013

STRATEGY FOR MODERNIZING A 40-YEAR-OLD ACCELATOR CONTROL SYSTEM*

B. Harrison, Fermi National Accelerator Laboratory, Batavia, USA

Abstract

Modernizing the Fermilab accelerator control system is essential to future operations of the laboratory's accelerator complex. The existing control system has evolved over four decades and uses hardware that is no longer available and software that uses obsolete frameworks. The Accelerator Controls Operations Research Network (ACORN) Project will modernize the control system and replace end-of-life power supplies to enable future accelerator complex operations with megawatt particle beams. The project team is evaluating three design concepts, and the future deployment of artificial intelligence capabilities for accelerator operations is an important consideration. An overview of the ACORN Project will be presented, including R&D used for evaluating the conceptual designs in the context of requirements for future accelerator operations.

ACORN

Accelerator Controls Operations Research Network (ACORN) will modernize the accelerator control system and replace end-of-life accelerator power supplies to enable future operations of the Fermilab Accelerator Complex with megawatt particle beams. ACORN is a DOE 0413.3B project with an estimated project cost from $100M to $140M and a lifetime of 8 to 10 years. As an upgrade of the accelerator complex, the ACORN team is analyzing risk associated with current accelerator operations and is developing requirements for modernization.

THE CURRENT CONTROL SYSTEM

Fermilab's accelerator control system has enabled major scientific discoveries such as the top quark, Tevatron collider program, g-2 anomalous muon magnetic moment measurement, etc., all without taking beam down for major upgrades. All upgrades have been incremental without interrupting the beam delivery program.

ACORN is Fermilab's first opportunity to take a large-scale and deliberate approach to modernizing the control system by "standing on the shoulders of giants": building on the success of the current system, upgrading to modern architectures, hardware, user interfaces, software, development processes, documentation, and integrating with modern toolkits like EPICS.

ACORN is taking a requirements-centric approach, first by gathering functional requirements of the current system from interviews with the division; we're making sure we don't lose functionality when we upgrade.

Using requirements as a guide, ACORN is building an aggressive R&D plan to inform our conceptual and technical design process. We will work with AD experts to investigate and prototype ambitious new systems that take advantage of cutting-edge technology, with a focus on standardization and compatibility across the entire division (including with PIP-II).

Requirements and R&D will enable exotic new uses of the control system, such as applied robotics and AI/ML.

We're working with Idaho National Lab (INL) with their human factors team to develop a set of user interface and user experience design standards and frameworks to improve accessibility.

REQUIREMENTS AND RISKS

ACORN in coordination with the accelerator division is performing risk analysis for accelerator operations risks. As a part of that risk analysis, the risks will be categorized based on criticality and cost. Those categories will be used to rank the risks as a priority list to be addressed by the project.

Using the risk ranking, ACORN will define project scope in conjunction with operational and other project risks that may be discovered.

We know that we have more work than our funding can support. The risk ranking allows us to prioritize and include the most important work in the project scope.

For the systems that need modernization, we will develop functional requirements for the existing accelerator control system to help ensure there's no loss of functionality in the modernization process.

In addition to functional requirements of the existing system, we will identify new requirements needed to implement new capabilities.

RESEARCH AND DEVELOPMENT

ACORN is beginning the R&D process to establish knowledge required to define our conceptual design.

Some topics we know we will investigate include:

- Explore new data acquisition capabilities
- Support for AI/ML for accelerator operations
- Evaluation of 5G technology
- Support for robotics
- Evaluation of Experimental Physics and Industrial Control System (EPICS)

Data Acquisition

Our requirements strongly suggest a centralized architecture and we have begun exploring the implications of that choice. In Figure 1 you can see a preliminary block diagram of key components.

We think that a Data Lake structure for our centralized storage will allow us the flexibility to add storage systems as we implement new tools. Key features of the Data Lake we need to test include:

- Measure expected data flows from different types of front-ends

* FERMILAB-TM-2783-AD

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-WE022

APPLICATIONS OF TIMING READ-BACK SYSTEM IN J-PARC MAIN RING

M. Yang†, N. Kamikubota, K.C. Sato, N. Kikuzawa, J-PARC Center, KEK&JAEA, Japan
Y. Tajima, Kanto Information Service, Tsuchiura, Japan

Abstract

Japan Proton Accelerator Research Complex (J-PARC) timing system has been in operation since 2006. Over the past 16 years, there were trigger-failure events, and some of which seriously affected the operation of J-PARC accelerator, especially at Main Ring. To troubleshoot the source of such events more quickly, we decided to develop a timing read-back system to read back distributed timing signals at the device side. A PLC-type triggered scaler module was developed as a key of the system. It can count number of pulses in an accelerator cycle and store the counts in a momentary array. Using the module, customized read-back applications for various timing-related signals have been developed: (a) read-back a 25-Hz trigger clock, (b) read-back a pulsed bending magnet trigger, (c) read-back a magnet power-supply trigger. These applications were implemented successfully in J-PARC Main Ring, and demonstrated as countermeasure against past trigger-failure events. More details will be given in the paper.

INTRODUCTION

J-PARC (Japan Proton Accelerator Research Complex) is a high-intensity proton accelerator complex. It consists of three accelerators: a 400-MeV Linac (LI), a 3-GeV Rapid Cycling Synchrotron (RCS), and a 30-GeV slow cycling Main Ring Synchrotron (MR) [1, 2]. Since the initial beam in 2006, J-PARC has been improving beam power. Concerning MR, recent beam power is about 500-kW (50-kW) by fast (slow) extraction, respectively [3].

There are two time cycles used in J-PARC. A rapid cycle, 25 Hz is used at LI and RCS, and a slow cycle is used at MR. When MR delivers proton beams to Neutrino (NU) and Hadron (HD) Facilities, 2.48-s (fast extraction mode (FX)) and 5.20-s (slow extraction mode (SX)) cycles are used, respectively. Because the slow cycle determines the overall time behaviour of the accelerators, it is also called a "machine cycle."

The control system for J-PARC accelerators was developed using the Experimental Physics and Industrial Control System (EPICS) framework [4]. In addition, a dedicated timing system has been developed and operated since 2006 [5, 6]. It consists of one VME transmitter module and approximately 200 VME receiver modules. Event codes, which correspond to the information on beam destinations and on beam parameters, are distributed from the transmitter module to all the receiver modules. A fiber-optic cable network is used for event-code distribution using several optical-to-electrical (O/E) or electrical-to-optical (E/O) modules. According to the received event code, each receiver module generates eight independent delayed-trigger signals.

Since the first beam use began in 2006, the J-PARC timing system has contributed to a stable operation of the accelerators [6]. Nevertheless, some timing trigger-failure events have occurred during beam operation. During each recovery process against a failure, it was often difficult to find a definite module among many modules suspected. Such experiences have prompted us to develop a new module, triggered scaler module, which can read back signals generated by the J-PARC timing system [7, 8].

In this paper, three trigger-failure events occurred in J-PARC MR are introduced briefly. The applications of timing read-back system are described in details, followed by a future plan of new power-supply with embedded timing modules.

PAST TRIGGER-FAILURE EVENTS

There have been some unexpected trigger-failure events in the past 16 years [8, 9]. Herein, three events are briefly given.

- In November to December 2016, the accelerator operation was suspended several times a day, because of the faults of a beam diagnostic system. The investigation showed that an O/E module used for 25-Hz trigger clock produced irregular signals. After replacing the module, the problem was solved.
- In January 2018, a delayed trigger which excites the pulsed bending magnet was stopped. Thereby, few miss-controlled beams went to an undesirable destination, Material and Life Science Experimental Facility (MLF). Soon we found that a fuse in a trigger-fanout module was broken. After replacing the module, the problem was solved.
- In November 2015, a bad quality beam appeared during stable beam delivery to HD. Such beams appeared a few times per month [7]. After about six months troubleshooting, we finally found that a timing receiver module for one of the MR steering magnets showed momentary errors by external common-mode noises. The problem was solved by adding ferrite cores to metal cables.

The first two events showed that a single failure of one module causes a critical problem in an accelerator. It showed no alert from neither the timing system nor the control system. Thus, it was unable to find the problems remotely. The third event was a very low-rate error, a few

† yangmin@post.kek.jp

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-WE023

AVOIDING AND MANAGING PITFALLS OF CONTROL SYSTEM PROJECTS

A. Jesenko*, G. Pajor, Cosylab d.d., Ljubljana, Slovenia

Abstract

Cosylab is the leading provider of software solutions for the world's most complex, precise, and advanced systems. We also provide software products and services to the largest medical device manufacturers and cancer centres worldwide. In our 20-year history, Cosylab has worked on hundreds of multi-year projects worldwide.

Software projects run the highest risk of cost and schedule overruns. On average, large IT projects run 45 percent over budget and 7 percent over time.

There are some common pitfalls when defining and executing projects in the Control Systems field. This article presents some concepts on how to address them, such as: having a clear definition of a project's scope, budget, and timeline; doing project risk management; having a clear division of responsibilities; and having a project sponsor from the management.

INTRODUCTION

Control system development is an engineering discipline like many the others, often with a quite complex life cycle. Jumping right into prototyping and coding can bring some initial results quickly but is very prone to a myriad of risks and undesired results, such as an unclear scope, a lack of acceptance criteria and vaguely defined responsibilities. This can consequently manifest in never-ending development, an unnecessary multiplication of effort and non-existent or sub-par documentation.

Control systems are an engineering discipline like all the others, but with an even more complicated cycle:

1. Write specifications
2. Architecutre
3. Design
4. Prototyping – probably the only fun part
5. Test procedures
6. Implementation (coding) – the only software part
7. Documentation
8. Testing
9. Debuging
10. Acceptance

Each control system project has the following entities

- Consumer: the entity that is ordering or is otherwise interested in getting parts of (or the whole) control system.
- Supplier: the entity that is supplying parts of (or the whole) control system to the consumer.

A control group can have either of these roles; it can be the consumer ordering the control system project from another entity and it can also be the supplier supplying the control system to their own or another institution. One part of the control group can also assume one role, and another part of the same group can assume the other role. In all these cases the basic concepts described in this article remain the same.

* anze.jesenko@cosylab.com

PROJECT PLANNING

Quite intuitively, project planning is a crucial process for any project. The better you define the project and how it will be closed in the beginning, the less unfinished scope will remain in the end. This can result in a lower total cost of ownership. Remember that there will be some work after the project is complete and this work should also be planned for. In any case, don't count on defining the project scope while the project is ongoing (see also "What about Agile?").

The basic elements that define a project are:

1. the scope (what will be done, and who will do it),
2. the budget (how much will it cost),
3. the risks (what can go wrong).

The first steps in creating a project plan are to define the scope of the project, and to find out the time and budget constraints. After that, the scope is broken down into a "work breakdown structure" (WBS), and the risk management process is started.

Project Scope

The project's scope is usually documented into a statement of work (SoW). This is an important document with details about:

- What was agreed would be done, and by whom.
- What will not be done in the scope of the project.
- What are the deliverables.
- How acceptance of deliverables will be done.
- What assumptions were made.

Project Timeline and Budget

A rule of thumb to estimate the budget for an accelerator control system is to take 10 - 15 % of the total budget. To start planning a more detailed timeline and budget, work should first be broken down into a work breakdown structure (WBS). An effort estimation should then be done for each task in the WBS. This is a good way to gain knowledge about how much effort from people of each profile you will need for the project. Note that the effort estimations will only be accurate if they are done by persons experienced doing them.

Once we have a WBS done, we need to know when we will be ready to start each of the tasks. Now it's useful to create a critical path. By looking at a critical path, we can tell prerequisites and a timeline for each of the tasks and the project in general.

Then we can set some milestones for the project. It's a good idea to plan monthly milestones, in which the team

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-TH011

BUILDING CONTROL SYSTEM REMOTELY

M. Lukaszewski*, K. Klys, E9 Controls Limited, London, England

Abstract

Building a control system for a scientific facility is a complex process that requires significant time coordination and the cooperation of hundreds of people. The latest of our control system for the 10 J 100 Hz laser system happened to be built when access to the lab was far more difficult, due to the pandemic: travel was much more complex, and local restrictions at one point prevented a large number of people from being in the lab simultaneously. This paper shows how, despite the demanding working conditions, we built the control system for the 10 J 100 Hz laser. The control system was designed in collaboration with CLF (Central Laser Facility, Science and Technology Facilities Council, UK) and HiLASE (Institute of Physics, Czech Academy of Sciences). The process of building the laser control system was divided into stages and we had to rely on remote working. This article discusses how we adapted each stage to work remotely, what tools we used, how we minimized risks, and what we would have done differently if we had started from scratch.

INTRODUCTION

This paper has been written to share the methods and tools used to build a control system for the 100 Hz 10 J laser system built in collaboration with CLF (Central Laser Facility, Science and Technology Facilities Council, UK) and HiLASE (Institute of Physics, Czech Academy of Sciences). The paper aims to share findings and processes, and evaluate risk awareness.

REMOTE WORK

The main engineering effort started shortly after the first wave of pandemic restrictions, thus making it impossible to work directly in the laboratory. Experience with previous deployments was leveraged, and it was decided that the core integration layer be built remotely without accessing the actual devices. This approach was possible because proper software practices were followed from the beginning, such as: expressive documentation, simulation of various system components, continuous integration during development, automation of repeatable tasks and open communication with stakeholders. All of these aspects will be discussed below.

RISK MITIGATION

Building a scientific system is an arduous and lengthy process. It can be broken down into individual stages, including design, tendering, subsystem construction and equipment integration. The process ends with the implementation of the control system and functional testing of the entire system.

* marcin.lukaszewski@e9controls.com

However, problems with tenders, broken supply chains and other factors affecting the work process, which are not the subject of this article, cause numerous delays that increase the risk of error during integration and negatively affect the developer experience.

To mitigate the risks arising from delays, several strategies have been implemented to react effectively in a rapidly changing environment. It should be noted that a large proportion of the problems that arise during the development of software for controlling equipment are due to a lack of adequate information or late acquisition of such information.

Involvement Therefore, it is essential to participate in the design work, where the parameters of each device are detailed. At this point, each change is "cheap and easy" to make, which provides an opportunity for finding errors regarding device communication and with possible impact on the quality of the integration of devices into the control system.

Regular Contact Also, direct and regular contact with equipment suppliers allows changes in the control software to be tracked, potentially affecting the integration efforts.

Device Grouping Off-the-shelf equipment that was widely available, or was from large suppliers was given the lowest priority. It was assumed that basic integration for these devices would be available in the environment or that the system would be relatively simple to build, given the availability of documentation.

Devices used in the past, or with similar functionality to those already deployed were given medium priority. It was assumed highly likely that software for such devices either already existed in the community, or could be used after modification.

The highest priority was given to new equipment, which was previously unknown and also from suppliers unfamiliar to the laboratory. Equipment from small suppliers was also included in this group. These devices were integrated first, which resulted in the broadest possible time window to develop a working version.

INFRASTRUCTURE

Production Environment – Laboratory

The control system infrastructure in the laboratory consists of five powerful servers that control data collection and devices. The servers are connected via a high-speed Ethernet network to which all devices in the system are connected. The servers run under Linux Ubuntu LTS. The software controlling the system is based on the Experimental Physics and Industrial Control System (EPICS) framework, described later in this article. The lab has other systems with a similar

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-TH013

UPGRADING THE IRRAD CONTROL SYSTEM GUIs USING OPEN-LICENSE AND CROSS-PLATFORM TECHNOLOGIES*

B. Gkotse†[1], A. Abdulhalim, A. S. Mølholm, F. Ravotti
Experimental Physics Department, CERN, Geneva, Switzerland
P. Jouvelot, Mines Paris, PSL University, Paris, France
[1]also at Mines Paris, PSL University, Paris, France

Abstract

The CERN Proton Irradiation Facility (IRRAD) is a reference facility in high-energy physics for the qualification of detectors, materials, and electronic components against radiation. A proton beam with a momentum of 24 GeV/c is delivered from the CERN PS accelerator to IRRAD and impinges on the components being tested, placed on remotely controlled movable stages. This equipment, operated by dedicated control systems, allows for the precise positioning of components in or out of the beam and facilitates the handling of irradiated components, while minimising the radiation received by the IRRAD operators.

Originally, the implementation of the Graphical User Interfaces (GUIs) of the IRRAD control system was based on proprietary software, thus limiting it to specific operating system. To address the issues linked to such dependencies in terms of openness, ease of development and access to state-of-the-art technologies, new GUIs have been designed and developed with open-license cross-platform software. In this paper, the IRRAD control system software architecture is detailed, and the lessons learned while implementing these new feature-rich GUIs are presented.

INTRODUCTION

The Proton Irradiation Facility at CERN (IRRAD) is an infrastructure dedicated to performing radiation hardness testing (irradiation experiments) on detectors, electronics, and materials. The IRRAD facility, located in the East Area of the CERN accelerator complex, receives a proton beam from the Proton Synchrotron (PS) accelerator with a momentum of 24 GeV/c in spills of 400 ms and a Gaussian shape typically 12×12 mm^2 FWHM wide [1]. The components that need to be tested (Devices Under Test, or samples) can be placed along the beam trajectory on the top of nine remotely movable stages, called IRRAD tables. The IRRAD tables have three degrees of freedom and can be used to move the samples horizontally (x-axis), vertically (y-axis) or rotate them with an angle (θ) with respect to the beam axis. These tables allow for positioning samples in a volume of up to $20 \times 20 \times 50$ cm^3 in and out of beam, or performing a scan (e.g. asynchronously move the samples across the beam direction in order to extend the irradiated portion of the samples). Placing or removing samples on the IRRAD tables requires accessing the irradiation area, which can be done only once per week, when the beam is stopped. However, for smaller samples with dimensions up to $5 \times 5 \times 20$ cm^3, a conveyor system (shuttle) can be used; this can move samples from the outside to the irradiation zone, following a 9m-path without the need of stopping the beam.

During CERN Long Shutdown 1 (2012-2014), a new hardware infrastructure had been put in place at IRRAD and software Graphical User Interfaces (GUIs) had been built, based on CERN-supported proprietary software, for the control of its tables and shuttle. These GUIs were sufficient for operating during CERN Run 2 (2014-2018) [2]. Nevertheless, several restrictions on portability, dependencies on specific platforms and requests for additional functionalities have since deemed necessary the upgrade of the control system GUIs using current open-license and cross-platform technologies. Presenting and discussing this important transition is the subject of this paper.

This article first provides an overview of the hardware components and infrastructure used in the IRRAD control systems. Then, we describe the open-licence and cross-platform technologies used for the development of the new Graphical User Interfaces (GUIs), explaining our software choices and architecture. Details of the new functionalities, software architecture and database schemes are provided for both IRRAD tables and shuttle control systems. Finally, we discuss the lessons learned during this software transition, before concluding and introducing future work.

IRRAD CONTROL SYSTEMS HARDWARE

Since the operation of the IRRAD equipment happens in a radiation environment, the hardware chosen for the tables, shuttle and associated control systems is custom-made, had to ensure radiation tolerance and be easily customizable depending on the experimental user requests. For both systems, some common components have been used but still certain differences remain. Details about the hardware infrastructure are provided in the following paragraphs.

IRRAD Tables

Each IRRAD table uses two stepper motors, one for horizontal movement and one for the rotating axis, while an AC motor is used for vertical movements. Figure 1 shows 3 of the IRRAD tables of the first zone of irradiation in the IRRAD facility. The three motors are controlled using an M300 microprocessor. The communication with the microprocessor is performed through the RS232 serial protocol. Since there are nine of these tables that require this type

* This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 654168.
† Blerina.Gkotse@cern.ch

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-TH014

HOW GANIL PLAN TO USE WEB TECHNOLOGIES TO UPDATE THE CONTROL SYSTEM USER INTERFACE

C.H. Haquin†, O. Delahaye, C.H. Patard, F. Pillon, J. Pivard, G. Senecal, GANIL, Caen, France

Abstract

The Grand Accelerateur National d'Ions Lourds (GANIL) Control System was developed in the first half of the nineties with ADA language. The user interfaces use MOTIF and XRT widgets. User interfaces have become more and more difficult to modify and there is a risk of obsolescence. GANIL plan to replace these old technologies and web technologies are anticipated. This paper will present the strategy defined to make the switch.

CURRENT SYSTEM

The control system of the original GANIL accelerator (2 ions sources, 5 cyclotrons), called GANICIEL, relies on several software (about 65), entirely developed in ADA language specifically for this machine thirty years ago, by the developers who were present at that time.

There are two main software layers (see Fig. 1), the first one, the human machine interfaces layer (about 45 software), corresponds to all the software required to tune the accelerator by operators from the control room. The other, the equipment control layer, comprehends all the real-time servers (about 10 software multi instantiated on 31 VME) interacting with equipment to handle Human Machine Interface's (HMI) requests and ensure parameters setting and data monitoring.

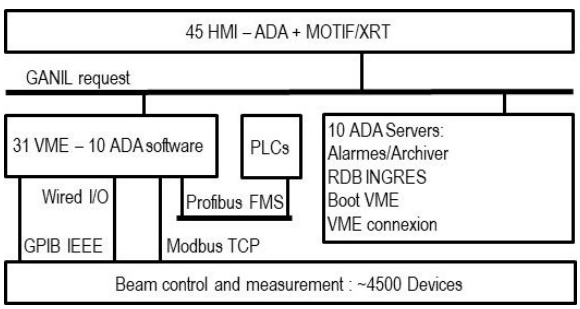

Figure 1: Current GANIL control system architecture.

To operate, both layers rely on relational databases (INGRES technology), at HMI layer level, widget are powered by MOTIF/XRT technology, at the equipment control layer level, VxWorks 6.9 operating system is required to run VME software.

In between the two layers, there is a batch (about 10) of software servers providing services for data bases access, alarm monitoring, data archiving and minimization of the Transmission Control Protocol/Internet Protocol (TCP/IP) connection between HMI and Versa Module Europa (VME).

GANICIEL Request Concept

The communication between the HMI layer and the equipment control layer is achieved over TCP/IP with a homemade request system providing simple and complex order services. Simple order offer the possibility to interact quickly with an equipment by sending unitary order with no reply, complex order gives the possibility to execute a sequence of order on an equipment and receive a report with only one transaction between a HMI and a VME.

Technical Threats and Opportunities

ADA Language The ADA language appeared in the early eighteen with ADA83 and was enhanced with revision ADA95, ADA2005 and ADA2012. Software developed in ADA are renowned to be robust, its attractiveness increased with the needs for security for payment systems, ADA is a technology we can rely on for the future. However, it is not widespread, the lack of skilled people and the weak attraction of the young developers for it are the main drawback that pushes us to reduce its use. Currently there are almost 3 MLoC in ADA for the whole control system (see Table 1).

Table 1: ADA Source Code Distribution

	Million Lines of Code
HMI, graphical rendering	1,0
HMI, command/algorithm	0,5
Total for HMI layer	1,5
VME ADA software	0,3
ADA servers	0,4
Total	2,7

HMI Layer There are two kind of HMI, one dedicated to equipment command and one to the so-called algorithmic command, equipment commands correspond the simplest HMI dedicated to a single equipment like basic Input/Output (I/O), driving a power supply, insertions. Algorithmic command HMI implement complex sequences like cycling a cyclotron or optimizing the beam transmission, they can involve several equipment potentially spread-out on several crates.

There are 1.5 Million Lines of Code (MLoC) for the HMI layer, 1 MLoC for the graphical rendering, the remaining half MLoC for the commands and algorithm. The widgets used for the graphical rendering are issued from the MOTIF/XRT libraries which are not commercialised and maintained since 2016, workmate that are now retired managed to keep these libraries up and running at each hosts renewal (every five years). We do not have the MOTIF/XRT skills anymore, there is a risk that we don't manage to repeat the performance for the next host renewal (2026) and we may face a blocking software or hardware incompatibility.

Equipment Control Layer The equipment control layer has a typical early nineteen architecture based on 31 VME crates allowing the control of the 4500 devices of

† christophe.haquin@ganil.fr

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THPP1

INAU: A CUSTOM BUILD-AND-DEPLOY TOOL BASED ON Git

L. Pivetta*, A.I. Bogani†, Elettra Sincrotrone Trieste, Basovizza (TS), Italy

Abstract

Elettra Sincrotrone Trieste is currently operating two light sources, Elettra, a third generation synchrotron, and FERMI, a free electron laser. Control systems are based on a number of diverse systems, such as VME-based front-end computers, small embedded systems, high performance rack-mount servers and control room workstations. Custom device drivers and hard real-time applications have been developed during the years, exploiting the technologies adopted such as RTAI and Adeos/Xenomai, which make a massive update demanding. Modern CI/CD tools are then not available for legacy platforms, and a custom tool, integrating git and a database back-end to build and deploy software components based on release tags has been developed.

INTRODUCTION

At Elettra and FERMI some complex subsystems, such as the global orbit feedback or the hard real-time interfacement of sensors and actuators, have been implemented exploiting the capabilities of GNU/Linux together with hard real-time extensions. Moreover, most of the interfacement of legacy equipment is made by VME based digital and analog I/O boards running on PowerPC single board computers. A number of device drivers and hard real-time applications have been developed in the years, making a complete system upgrade unfeasible. This turns into an effective limitation against adopting and using modern Ci/CD tools and procedures, which are not compatible with legacy systems.

LEGACY SYSTEMS, AND NOT

Different platforms are in use at Elettra and FERMI, depending on generation and the specific integration requirement. Leaving out some oldest systems running Microware OS-9, which are no more developed, and thus, not supported by INAU, several generations of GNU/Linux based single board computers and rack-mount servers are in use in front-end systems:

- Emerson MVME5100 and MVME6100, running Debian 3 (Linux kernel 2.4)
- Emerson MVME7100, running Ubuntu 7.10, (Linux kernel 2.6)
- Supermicro Intel-based rack-mount servers, running Ubuntu 10.04 (Linux kernel 2.6)
- Supermicro Intel-based rack-mount servers, running Ubuntu 14.04 (Linux kernel 3.14)
- Artesyn MVME2500, running Flop (Linux kernel 4.7)
- Jetway NUC, running Flop (Linux kernel 4.7)
- Beaglebone White and Black, running Flop (Linux kernel 3.14)
- Beaglebone AI, running Voltumna (Linux kernel 5.4)
- Altera Sockit, running Voltumna (Linux kernel 5.4)
- Supermicro Intel-based rack-mount servers, running Voltumna (Linux kernel 5.4)
- Dell Intel-based rack-mount servers, running Voltumna (Linux kernel 5.4)

Moreover, each accelerator deserves a central control system cluster, based on Proxmox, running a large number of virtual machines, where all the standard network services, as well as the Tango databases and the high-layer Tango devices run. All these virtual machines run Ubuntu 18.04 LTS. Control room workstations currently run Ubuntu 18.04 LTS as well.

As a side note, Voltumna is also available as virtual machine image, providing an effective approach to a totally reproducible deployment for virtual machine hosts, based on revision-control.

DEVELOPMENT ENVIRONMENT

Depending on the target system, different approaches are used to build the applications. For legacy VME based systems, as well as older intel-based servers, native target compilation/build, based on reference development hosts, is in place. Newer systems, based on FLOP [1] and, more recently, on Voltumna Linux, exploit a dedicated cross compilation environment.

Since 2019 Git is used for revision control of all software components developed in-house. For control systems software, a structure has been defined to easily navigate repositories, and conventions are in place to guarantee a strict consistency within Git repositories.

DEPLOYMENT REQUIREMENTS

Control system integrity and robustness requirements prevent from spreading administrator access rights to all the developers. Instead, system administrators take care of maintaining and servicing the control systems hosts. However, developers want/need flexibility, and not to depend on sysadmin for installing or updating applications and/or specific application configuration files. On the other side, there is the requirement to keep trace of who installed, updated or downgraded what, when and where. This requirement can be easily solved using modern CI/CD tools on reasonably recent hosts, but is unfeasible with old systems. To make this available on legacy platforms a custom build-and-deploy tool has been developed, named INAU which stands for AUtomatic INstallation, in Italian.

* lorenzo.pivetta@elettra.eu
† alessio.bogani@elettra.eu

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THPP2

EPICS MODULE FOR BECKHOFF ADS PROTOCOL

Jernej Varlec*, Jure Varlec, Žiga Oven, Cosylab d.d., Ljubljana, Slovenia

Abstract

With increasing popularity of Beckhoff devices in scientific projects, there is a rising need for their devices to be integrated into EPICS control systems. Our customers often want to use Beckhoff PLCs for applications that must handle many inputs with fast cycle times. How can we connect Beckhoff devices to EPICS control systems without sacrificing this performance?

Beckhoff offers multiple possibilities when it comes to interfacing with their PLCs or industrial PCs, such as Modbus, OPC UA, or ADS protocol. While all of these could be used for the usual use cases, we believe that for more data intensive applications, ADS works best. For this reason, Cosylab developed an EPICS device support module that implements advanced ADS features, such as ADS sum commands, which provide fast read/write capabilities to your IOCs.

INTRODUCTION

When designing EPICS device support, there are two questions one usually considers: how the device support will communicate with the target devices, and which representations of data must be supported.

ADS for Communication

Invented by Beckhoff, *Automation Device Specification* (ADS) [1] is an open protocol that is used for interconnecting various Twincat software modules the company provides, such as event logger, HMI framework, or Twincat PLC runtimes, among others. These modules are considered as being independent virtual devices and form a server/client relationship; an example of this is a field on the HMI screen getting an update from a Twincat Runtime, all via ADS.

Within this relationship, as shown in Fig. 1, servers and clients communicate with ADS request/response messages which need to be, somehow, routed to the correct ADS device. This message routing is the responsibility of the AMS router, which is part of every Beckhoff Twincat device. The router checks the AMS header part of the message and reads the target port and address from it. Beckhoff provides fixed specification, called AMS ports [2], for their internal software modules. For example, typical Twincat 3 runtime uses AMS port 851.

The AMS ports are the first identifier required for communication within the Twincat system, the second part are the *AMS NetIDs*. These look similar to IP addresses with additional octets appended, and often they are: Users may find it easiest to just use their device's IP address, and then append "`.1.1`"[1] to it. Note that the AMS NetID can be anything, as long as it is unique.

In general, there are two types of ADS communication, asynchronous and notification. Asynchronous is exactly what it says on the tin: the client sends the request message to the server, then continues to operate normally until the server provides the client with the response, be it success or error. Notification-based communication, on the other hand, allows the clients to register themselves to the server, which then autonomously provides updates when values client registered change.

Beckhoff provides the ADS client functions in a C++ library published on Github [3].

IEC 61131-3 Data Types

Another thing to consider is what data types are used by a Twincat runtime. Twincat supports all the typical data types one would expect, such as signed an unsigned integer types up to 64-bit width, 32- and 64-bit floating point numbers, support for arrays and strings, pointer and reference types, and structures. But, since it conforms to the IEC 61131-3 standard [4], it also supports some more 'exotic' data types, such as 32-bit *DATE*, *DATE_AND_TIME*, *TIME_OF_DAY*, and their 64-bit versions, and a generic type, called *ANY*.

The list of supported types is quite long and supporting them all could prove to be a challenge. Another thing we must consider is the data types that EPICS records support. For example, *longin* record supports 64-bit signed integers, which means support for writing 64-bit unsigned types into this record type could prove to be problematic.

Of course, speed and stability should be considered as well. One might not think about it much as long their use case requires mere hundreds of reads or writes at a time,

[1] `.1.1.` is used here as a typical example

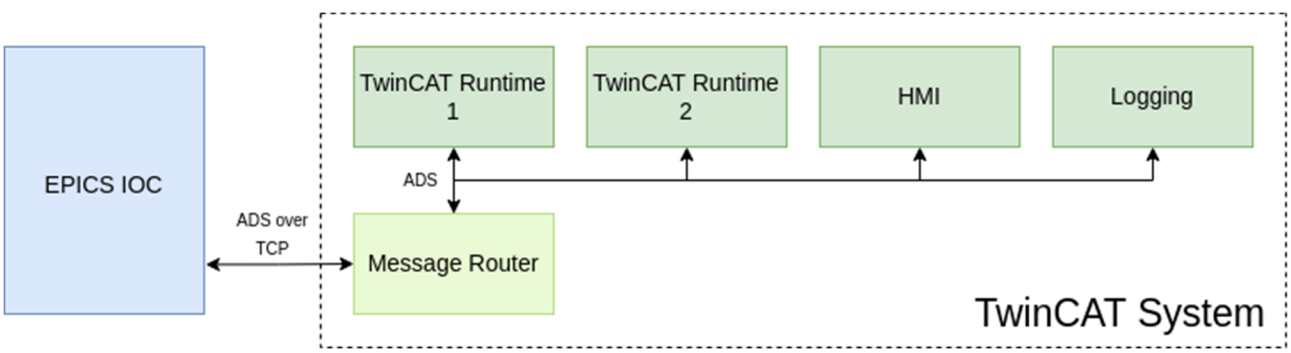

Figure 1: ADS communication via the Message router.

* jernej.varlec@cosylab.com

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THPP3

STORAGE RING MODE FOR FAIR

Andreas Schaller, Jutta Fitzek, Hanno Christian Hüther, Raphael Müller,
Benjamin Peter, Anneke Walter, GSI, Darmstadt, Germany

Abstract

For the future Facility for Antiproton and Ion Research (FAIR), which is currently under construction, a new Control System is being developed and already used at major parts of the GSI facility. The central component for Settings Management within the FAIR Control System is based on CERN's framework "LHC Software Architecture" (LSA) and enhanced by FAIR specific features. One of the most complex features is the control mechanism of storage ring operations, the so-called Storage Ring Mode. This operation mode allows to manipulate device settings while the beam is circulating in the ring. There are four different types of possible changes in the Storage Ring Mode: skipping, repetition, breakpoint and manipulation. The Storage Ring Mode was developed in late 2019 and first used with beam in 2020 at the existing heavy ion Storage Ring ESR at GSI. This contribution illustrates in detail how the Storage Ring Mode is implemented within LSA and other subsystems. It also shows how it is operated using the Expert Storage Ring Mode application.

MOTIVATION

To reliably ensure deterministic behavior, the new Timing System for FAIR executes predefined event sequences. The first use case realized in the FAIR Control System was synchrotron operation, based on fully pre-planned schedules, which has been successfully used since 2015.

In 2019, the FAIR Control System was enhanced to support storage ring operation [1]. The old control system supported storage ring operation at GSI since 1990, with beams often being stored for several hours and up to a week. The Storage Ring Mode of the new control system now also supports the required flexibility and allows interactive schedule changes, including in-cycle modifications of stored beams.

The basics of Storage Ring Mode were presented at ICALEPCS in 2021 [1]. This contribution provides more insights into the realization of its features.

STORAGE RING MODE - COUPLING WITH SOURCE MACHINE

Storage Ring Mode allows for interactive, dynamic operation and very long, arbitrary beam storage times. Therefore, event sequences in the Timing System for storage rings are usually completely independent and separated from the event sequences of other accelerators.

However, when transferring beam from a source accelerator into a storage ring, the event sequences of both machines have to be synchronized. This is realized by a so-called coupling mechanism. The FAIR Control System currently supports two types of coupling: strong and weak coupling.

Strong Coupling

Strong coupling guarantees that the beam is properly injected by having the storage ring explicitly request beam and wait until it is ready to be injected.

As shown in the left part of Fig. 1, the storage ring prepares for beam injection, requests beam from the injector, and finally enters a wait loop. The injector starts creating the beam once the overall schedule allows it. When beam is ready, the injector sends a command to the storage ring to leave the wait loop. Extraction from the injector and injection into the storage ring then continue synchronously.

This type of coupling is used by the ESR Heavy Ion Storage Ring when receiving beam from the SIS18 ring.

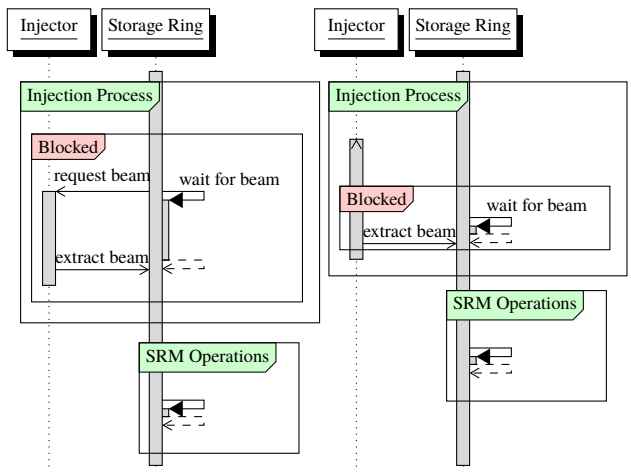

Figure 1: Sequence diagram of a strong coupling (left) and weak coupling (right).

Weak Coupling (Fire and Forget)

With weak coupling, the injector *always* produces and extracts beam without making sure that the storage ring is ready for transfer ("fire and forget"). This coupling mode is used to allow the storage ring to serve other experiments while the beam is produced, instead of just waiting.

As shown on the right part of Fig. 1, the injector produces beam without an explicit beam request. When beam is ready, the injector notifies the storage ring to leave the wait loop. At this point, the storage ring must be ready for injection when the injector starts extraction. Operators achieve this by carefully aligning the schedules of the injector and the storage ring. If the storage ring is not ready for beam at this point, the notification is ignored and beam is lost.

This mode is used at CRYRING@ESR [2], enabling CRYRING to accept beams from its local linear accelerator in synchrotron mode while waiting for ESR's beam.

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THPP5

LASER PULSE DURATION OPTIMIZATION WITH NUMERICAL METHODS

Francesco Capuano[1,*], Davorin Peceli, Bedřic Rus, Alexandr Špaček
ELI, Institute of Physics, Czech Academy of Sciences, Dolní Brežany, Czech Republic
Gabriele Tiboni[†], Politecnico di Torino, Turin, Italy
[1]also Politecnico di Torino, Turin, Italy

Abstract

In this study we explore the optimization of laser pulse duration to obtain the shortest possible pulse.

We do this by employing a feedback loop between a pulse shaper and pulse duration measurements. We apply to this problem several iterative algorithms including gradient descent, Bayesian optimization and genetic algorithms, using a simulation of the actual laser represented via a semi-physical model of the laser based on the process of linear and nonlinear phase accumulation.

INTRODUCTION & RELATED WORKS

L1-Allegra System

L1 Allegra system is high power laser system developed in ELI Beamlines in Czech Republic designed to deliver <20fs pulses with energy higher than 100 mJ at a repetition rate of 1 kHz. This system is based on the amplification of frequency-chirped pico-second pulses in an Optical Parametric Chirped Pulse Amplification (OPCPA) chain consisting of seven amplifiers. L1 contains 7 OPCPA stages that are pumped by 5 diode pumped lasers based on commercial Ybdoped thin-disk Regenerative Amplifiers (RA) DIRA 200-1. All of the pump lasers generate pulses at 1030 nm with 1 kHz repetition rate and also include stretcher, compressor and an LBO crystals for second harmonic generation (SHG) of pulses at 515 nm. To achieve high SHG efficiency several features of the pump laser system should be optimized. One of these features is the laser pulse temporal shape. The optimization of temporal pulse shape is accomplished through the manipulation of the pulse spectral phase. A standard method for spectral phase representation employs a Taylor expansion of spectral phase, $\varphi(\omega)$, around the central angular frequency ω_0, of the spectral pulse:

$$\varphi(\omega) = \sum_{=0}^{\infty} \frac{\partial^{(\ )}\phi}{\partial\omega^{(\ )}} \cdot (\omega - \omega_0) \ . \tag{1}$$

The first two terms of the sum of Eq. (1) represent constant phase and group delay and don't have effect on the pulse temporal shape. Third, fourth and fifth term are instead related to the dispersion parameters group delay dispersion (GDD $\sim \varphi'$), third order dispersion (TOD $\sim \varphi''$) and fourth order dispersion (FOD $\sim \varphi'''$) and do have an impact on the temporal profile. In this work, we aim at control these parameters, to which we will collectively refer to as ψ.

* Francesco.Capuano@studenti.polito.it
† Gabriele.Tiboni@polito.it

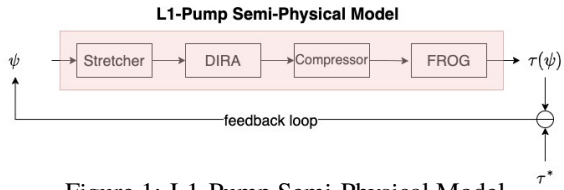

Figure 1: L1-Pump Semi-Physical Model.

Since SHG is an intensity-dependent non-linear optical process, the highest conversion efficiency is obtained for those pulses with temporal shape closest to transform-limited pulses. In particular, since the transform-limited (TL) pulse is the pulse with minimum possible pulse duration given the present spectral bandwidth, it is guaranteed to have the highest possible pulse intensity.

Temporal properties of amplified pulses are controlled through manipulation of parameters of the stretcher by *Teraxion* and monitored using SH FROG 1030 by *Femtoeasy*. The control of pulse temporal shape is done by manipulating three parameters, d_2, d_3 and d_4 on the stretcher controls that relate to parameters GDD, TOD and FOD previously introduced through a determinate system of linear equations, depending on the central wavelength (derived from central frequency) of the spectrum considered.

Related Works

Obtaining short and sharp laser pulses is crucial for running high-efficiency physical systems that would reach really high intensities with limited energy consumption.

Various algorithms were used in ultra-fast laser systems for active feedback optimization of laser parameters or laser-matter interactions [1, 2], but none of these previous approaches optimized the single pulse shape.

In this work we compare three different algorithms for the optimization of the temporal profile of laser pulses. These are: Bayesian Optimization, Differential Evolution, and a custom implementation of gradient descent.

Semi-Physical Model of L1-Pump-System

Optimization of laser pulse shape is designed through feedback control loop between stretcher and FROG. To test performances of different optimization algorithms we designed a theoretical semi-physical model of the L1 amplifier chain containing stretcher, DIRA, compressor and FROG, as presented in Fig. 1.

Each of the elements in the pump chain contribute to the overall spectral phase. Furthermore, while stretcher and compressor introduce only linear change to the spectral phase (GDD, TOD and FOD), DIRA also introduces nonlinear

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THPP6

MONOCHROMATOR CONTROLLER BASED ON ALBA ELECTROMETER Em#

A. Baucells †, G.Agostini, J.Avila-Abellan, C. Escudero, O.Matilla, X.Serra-Gallifa, O.Vallcorba
CELLS - ALBA Synchrotron, Cerdanyola del Vallès, Spain

Abstract

Guaranteeing that the X-Ray beam reaches the experimental station with optimal characteristics is a crucial task in a synchrotron beamline. One of the critical factors which can lead to beam degradation is the thermal drifts and the mechanical inertias present in the optical elements, such as a monochromator. This article shows a new functionality of the ALBA Electrometer (Em#), which ensures that the beamline receives the maximum possible beam intensity during the experiment. From the current reading of an ionization chamber and driving the piezo-actuator pitch of the monochromator, the Em# implements a Perturb and Observe (P&O) algorithm that detects the peak beam intensity while tracking it. This feature has been tested on NOTOS beamline and the preliminary results of the performance are shown in this paper.

INTRODUCTION

Monochromators are essential elements in a beamline to study the behaviour of materials and they actively play a role to perform spectra and scans in 3rd generation synchrotrons. While they are robust optical elements they are still subject to slight beam degradation caused by the mechanics inertia, temperature drifts and the parallelism of the crystals. To deal with this, several beamlines at ALBA have been using the Monochromator Controller from ESRF for many years. As the number of beamlines increases the need to guarantee stock and support of these controllers has led to explore the development of a monochromator controller using the ALBA Electrometer Em# [1]. Another motivation to implement this new functionality to the Em# is to use the monochromator controller in oscillation mode, allowing to operate at the maximum point of the intensity.

NEW FUNCTIONALITY OF THE EM#

The implementation of this new feature of the Em# is based on the usual arrangement of a beamline to monitor the intensity generated by an ion chamber in the end station. In particular, it has been designed considering the structure of the NOTOS beamline at ALBA. In the experimental hutch, an ion chamber is used at the sample environment generating a current that is measured by an Em#. The monochromator controller will apply an analogue voltage signal to the piezo amplifier that drives the piezo motor stacked over the pitch stage of the DMM monochromator in the Optical Hutch (Fig. 1). Due to this mechanical arrangement of the pitch crystal, the motion produced by both the stepper motor and the piezo will affect the same encoder reading on that stage. For that reason, the pitch axis has to operate in open loop to be closed then by the ion chamber, generating the current to be processed by the monochromator controller and applying a voltage to drive the piezo.

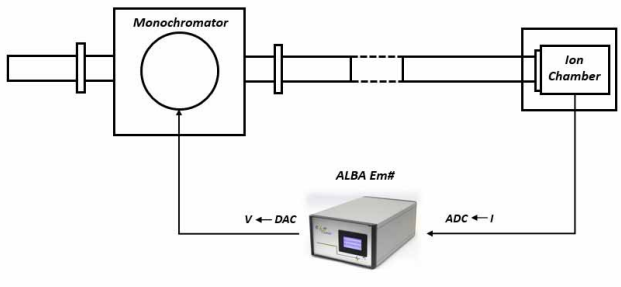

Figure 1: Schema of the Em# as monochromator controller.

The Em# will process the incoming values by executing a Maximum Power Point Tracking (MPPT) algorithm (Fig. 2), based on a Perturb and Observe (P&O) technique widely used for optimization of power management of solar cells [2].

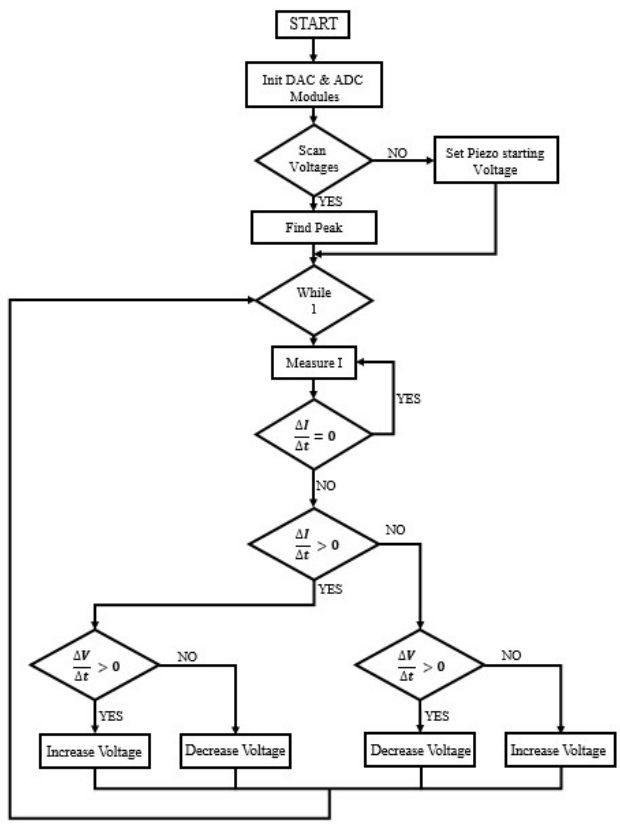

Figure 2: MPPT algorithm of the Em#.

† abaucells@cells.es

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THPP7

OPC UA BASED USER DATA INTERFACE AT ELBE

K. Zenker*, M. Justus, R. Steinbrück
Institute of Radiation Physics, Helmholtz-Zentrum Dresden-Rossendorf, Dresden, Germany

Abstract

This paper presents an OPC UA based data interface implemented at the high-power radiation source ELBE. It provides access to ELBE machine data via dedicated gateway devices.

In addition to the intrinsic security and authentication features included in OPC UA an access control mechanism was implemented at the PLC level. This allows to enable/disable user data access using the SCADA system of ELBE.

INTRODUCTION

The center for high-power radiation sources ELBE delivers different kinds of radiations serving a variety of user groups. The facility is in operation for more than 20 years and over time different machine interfaces were created for different user groups tailored to their needs. On the one hand some user groups provide experiment data like counting rates to the operators. This information is used for machine optimization. On the other hand some user groups have the permission to change certain machine parameters, like undulator gap size or beam repetition rate on their own.

So far different solutions have been used to provide those machine interfaces. Solutions include direct access to the SCADA system via dedicated accounts and OPC interfaces available in the SCADA system.

Here we present an approach of providing a common machine and ELBE data interface based on the OPC UA standard [1]. It is applicable to all use cases at ELBE and can be accessed via a single access point OPC UA server.

In the following, the infrastructure at ELBE is introduced. After the ELBE data interface (EDI) and its architecture is discussed.

ELBE INFRASTRUCTURE

ELBE is operated using the SCADA system WinCC by Siemens. The majority of ELBE systems is connected to WinCC via industrial Ethernet and proprietary S7 communication. At control level SIMATIC S7 300 and S7 400 Programmable Logic Controllers (PLC) are used. Different subsystems, like the machine protection system or the personnel safety system, and different subsets of each subsystem, belonging to e.g. different beam line sections, are implemented on dedicated individual PLCs.

The integration of subsystems based on MicroTCA.4 [2] hardware, which do not provide S7 communication interfaces, into the existing infrastructure is based on OPC UA using the open source C++ software toolkit ChimeraTK [3, 4]. Here, OPC UA allows a direct integration into WinCC using the native OPC UA support of WinCC.

* k.zenker@hzdr.de

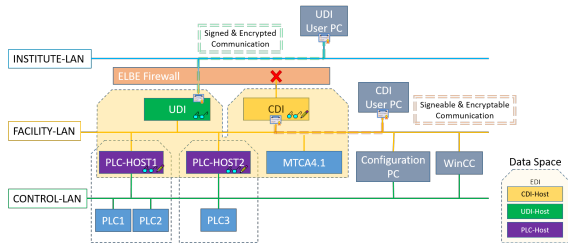

Figure 1: Overview of the ELBE data interface (EDI). The only available access points are the gateways named UDI and CDI. All other components are hidden to EDI users.

Commercially available OPC UA gateways [5] (UA Link) by IBHsoftec are used at ELBE as a data bridge between the Siemens S7-300/400 PLCs and OPC UA based applications as described in [6–8]. These UA Links are the basis of EDI.

ELBE DATA INTERFACE ARCHITECTURE

As introduced above two data sources – direct OPC UA sources like MicroTCA based subsystems and indirect OPC UA sources that aggregate PLC data via UA Links – need to be considered for EDI. Both can be interfaced by UA Links, which allows to implement EDI based on the UA Links.

From a traffic point of view it is desirable to place the UA Links as near as possible next to the PLCs in respect to the control network. Where possible the gateways have been directly connected to the according PLC interfaces. This automatically led to the introduction of several so called PLC-Host gateways throughout the ELBE machine and a multi-layer structure as introduced in the following.

Abstraction Layers

Figure 1 shows an overview of the EDI architecture currently implemented at ELBE. In a first abstraction layer of EDI are the PLC-Hosts, that are shown in the bottom part of Fig. 1. Depending on the location and the address space of the PLCs data of multiple PLCs can be aggregated by a single UA Link (see Fig. 1 PLC1 and PLC2) or a single UA Link per PLC-Host is used (PLC3 in Fig. 1). At this level the OPC UA address space is automatically generated based on the symbols defined in the PLC STEP7 project.

In general, the configuration of a UA Link allows to define the access level for each process variable on the OPC UA side. Additionally user accounts can be configured to have read only or read write access. However the available process variables are the same for each user and single process variable access cannot be defined user dependent. Furthermore, it is not possible to change process variable access during runtime.

Since UA Links at this level are not visible to the outside world it is possible to waive encryption and authentication

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THPP8

IC@MS — WEB-BASED ALARM MANAGEMENT SYSTEM

Ł. Żytniak*, M. Gandor, P. Goryl, J. Kowalczyk, M. Nabywaniec, S2Innovation, Cracow, Poland
S. Rubio-Manrique, ALBA Synchrotron Light Source, Cerdanyola del Vallès, Spain

Abstract

The IT world is moving to the web and cloud. IC@MS is a web-based alarm management system. Every control system can face unexpected issues, which demand fast and precise reactions. As the control system starts to grow, it requires the involvement of more engineers to access the alarm list and focus on the most important ones. IC@MS allows the users to access the alarms fast, remotely via a web browser. According to current trends in IT, creating a web application turned out to be the most comfortable solution. IC@MS is the extension and web equivalent to the Panic GUI desktop application. There is no need to install it on the client's computer. The access to the different functionalities can be restricted to the users provided just with appropriate roles. The web-based alarm management system provides a better user-friendly user interface for everyday use with Integration with Active Directory. Alarms can be easily added, edited, and managed from the web browser*. It has a Web API that can be used by 3rd party applications. The instance of IC@MS is available on Amazon Web Services (AWS) and Microsoft Azure clouds.

ALARM SYSTEM IN TANGO

Tango Controls is a free open source device-oriented controls toolkit for controlling any kind of hardware or software and building SCADA (Supervisory Control And Data Acquisition) systems (Fig. 1). Tango Controls is operating system independent and supports C++, Java and Python for all the components. As Tango Controls exists for more than 20 years and becomes more and more popular among facilities, it has proved itself as a reliable toolkit [1]. Engineers at institutes like particle accelerators are daily dealing with hundreds or thousands of signals per second coming from different types of devices. Therefore, there is a need of an application to detect and monitor non typical situations. One of the most popular solution is PANIC and PyAlarm Device Server.

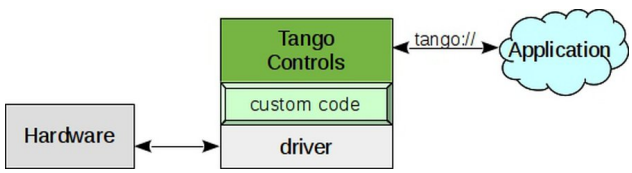

Figure 1: Tango high level overview.

PANIC

PANIC (Package for Alarms and Notification of Incidences from Controls) is an alarm system developed in ALBA Synchrotron. It is a set of of tools that provides:

- Periodic evaluation of a set of conditions.
- Notification (email, sms, pop-up, speakers)
- Keep a log of what happened. (files, Tango Snapshots)
- Taking automated actions (Tango commands / attributes)
- Tools for configuration/visualization

The Panic package contains the python AlarmAPI for managing the PyAlarm device servers from a client application or a python shell. The panic module is used by PyAlarm, Panic Toolbar and Panic GUI [2].

PyAlarm

The key element of PANIC toolkit is PyAlarm device server. This device server is used as a alarm logger, it connects to the list of attributes provided and verifies its values [3]. That features are needed to ensure comfortable alarm management for user. Each alarm is independent in terms of formula and receivers, all alarms within the same PyAlarm device will share a common evaluation environment determined by PyAlarm properties.

Panic GUI

Panic GUI is a application written using Taurus library. It allows to check existing alarms and manipulate them. Panic GUI is an desktop application for controlling and managing alarms. It depends on panic and taurus libraries. It allows the user to visualize existing alarms and adding/editing/deleting alarms. In edit mode user can change name, move alarms to another device, change descriptions and modify formulas. Additional widgets in which the app is equipped allows alarm history viewing, phonebook editing and device settings manipulation.

EPICS

EPICS (Experimental Physics and Industrial Control System) is a set of Open Source software tools, libraries and applications developed collaboratively and used worldwide to create distributed soft real-time control systems for scientific instruments such as a particle accelerators, telescopes and other large scientific experiments. Moreover, it is a set of software tools and applications which provide a software infrastructure for use in building distributed control systems (Fig. 2). Such distributed control systems typically comprise tens or even hundreds of computers, networked together to allow communication between them and to provide control and feedback of the various parts of the device from a central control room, or even remotely over the internet [4].

* lukasz.zytniak@s2innovation.com

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THPP9

SIMPLE PYTHON INTERFACE TO FACILITY-SPECIFIC INFRASTRUCTURE

J. Gethmann*, E. Blomley, P. Schreiber, M. Schuh, W. Mexner, A.-S. Müller
Karlsruhe Institute of Technology, Karlsruhe, Germany
S. Marsching, aquenos GmbH, Baden-Baden, Germany

Abstract

The various particle accelerators hosted at KIT represent a complex infrastructure with a live control system interface, a data archive, measurement routines, and storage and management of metadata, among other aspects. The "KIT Accelerator Python tools" were created to provide a unified interface to all aspects of the accelerator infrastructure for both short-term student projects and basic accelerator operations. Instead of creating another custom framework, these sets of tools focus on bridging the gap between well-established libraries, our facility and accelerator specific needs. External and accelerator specific libraries are glued together to provide an interface in order to minimise the technical knowledge of the accelerator infrastructure needed by the end user. Best practice software engineering workflows of continuous integrationn were implemented to provide automatic testing, packaging, API documentation and release management. This paper discusses the general motivation and approach taken to create and maintain such a set of Python modules.

INTRODUCTION

At KIT, we operate multiple kinds of accelerators (KARA, FLUTE) and further compact accelerators are in the planning and building phase. KARA, a 2.5 GeV storage ring, a part of the KIT Light Source, with a circumference of 110 m. FLUTE [1] includes a linear accelerator. Further accelerators are planned, e. g. a plasma accelerator and a compact storage ring within the project cSTART [2]. This paper first describes the setup of the controls infrastructure, then introduce the requirements for accelerator physics' experiments and the control of the machines. Then, the strategies to solve issues and the decisions made are described. Lastly, we present some lessons that we learned in the process.

SETUP

The controls infrastructure has to integrate the required different operation modes of the different KIT accelerators with a quick and safe access by e. g. non-experienced students via users of synchrotron radiation to experts with very demanding accelerator physics and technology experiments.

The data flow starts by providing live data and control for machine parameters from magnet power supplies to diagnostics instrumentation. Relevant parameters are archived in databases that are flat for performance reasons and are furthermore highly available. Snapshots of machine settings can be saved to and restored from relational databases for selections of groups, e. g. the pre-accelerator synchrotron, in short booster, settings can be restored independently from the storage ring's settings. Furthermore, there exist electronic lab notebooks for routine operation and certain typical experiments. For machine optimisation purposes and typical measurements needed for beam dynamics simulations, MATLAB® routines write their outputs to a central file storage.

Though the controls architecture for the different accelerators is similar, there are logical stand-alone systems for each accelerator requiring different addresses and namespaces.

REQUIREMENTS

For different stakeholders, the requirements differ. Especially for students and visiting researchers, it is necessary to get easy and quick access to certain sub-systems without in-depth knowledge of how exactly these systems are set up or interact with each other. They do not need to know from which control system component the data originates nor which protocol that systems speaks. Another crucial point is a quick setup, possibly with limited permissions granted. Facility scientists need access to a broad variety of data and therefore have to interact with many systems at the same time. However, the knowledge of the specifics of each particular system is typically not required. Finally, there are the developers of such systems who want to have maintainable and flexible code, which is reusable and actually used. This includes to adopt FAIR principles. We need straightforward access to all systems without the domain knowledge of the individual sub-systems or the infrastructure's topology.

In the past, we struggled with these different user groups trying to solve similar issues with their own individual scripts. Such scattered scripts were often written in different languages, with different levels of code quality and focusing on different aspects of the same task. This resulted in making these scripts very hard to share, maintain and re-use, especially across different user groups. The user experience of systems with different interfaces for similar behaviour is not good and can lead to errors in the worst case. Lastly, substantial inside knowledge is required to be aware of such scattered scripts and where they can be found.

APPROACH

Instead of creating another framework or a generic scientific library, our approach bundles existing ones for convenient ways to use them for the KIT accelerators according to our needs.

* gethmann@kit.edu

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THP02

CONTROL SYSTEM FOR HESEB BEAMLINE AT SESAME

A. Abbadi*, A. Al-Dalleh, A. Aljadaa, M. Abugharbiyeh, M. Alzubi,
M. Genisel, R. Khrais, S. Matalgah, Y. Momani
SESAME Synchrotron-light, Allan, Jordan

Abstract

The HElmholtz-SEsame Beamline (HESEB) is a state-of-the-art soft X-ray beamline donated by Helmholtz research centre that was successfully installed and commissioned recently at Synchrotron-light for Experimental Science and Applications in the Middle East (SESAME). The control system design and implementation, which includes controlling low-level up to most sophisticated devices, has been done by SESAME's control engineers. This paper describes the design and implementation of the control system required to deliver the complete functioning of the beamline as well as the safety system including its measures put in place to protect the beamline's equipment and users.

INTRODUCTION

HESEB is the first soft X-ray beamline at SESAME and will significantly expand the research capabilities available to the user community in the Middle East and neighbouring regions. The source of the HESEB beamline is based on an elliptical polarising insertion device of the APPLE-type. The undulator's ability to provide linearly to circularly polarized light makes the beamline very suitable for materials science applications, especially magnetic materials. Its plane grating monochromator uses exchangeable gratings to cover a photon energy range from 70 eV to 2000 eV [1].

SESAME employs the Experimental Physics and Industrial Control System (EPICS) toolkit for controlling both machine and beamlines. EPICS is a control system middleware or framework for acquiring and controlling equipment through standardized communication between distributed components of different subsystems. The goal of beamline control system is to provide the users an easy interface to interact with the beamline components and devices. All beamline devices are connected to central units called EPICS Input Output Controllers (IOCs). These IOCs are hosted on Virtual Machines (VMs), there are three VMs per beamline, all of which reach the same network with the rest of the beamline's devices and hardware. A beamline network is separated from the other beamlines and machine networks. External access to beamline IOCs is provided through EPICS gateway.

CONTROL SYSTEM

Beamline control system consists of several sub-systems in which are illustrated in Fig. 1.

Equipment Protection System (EPS)

Each beamline has its own independent EPS in order to keep the devices and components safe while they are operating. It constantly checks the status of the beamline's parts and collects signals from field sensors to allow or inhibit the operation of beamline. This includes:

- Temperature measurement to protect beamline components where high heat loads are expected.
- Monitoring of coolant flow rates and react in case of insufficiency.
- Control the vacuum valves based on the signals received from the vacuum gauges and ion pumps.
- Watch the shutters' opening and closing time and report it to the control system. Also, it reacts in case of fail to close or limit switch failure.

PLC The EPS of SESAME's beamlines are PLC based. SIEMENS S7-1500 is the main controller used in HESEB. This model replaced the older SIEMENS S7-300, which was used in all beamlines before HESEB. Recently, a decision was made to consider the S7-1500 and S7-1200 in all new projects as the S7-300 model will be discontinued in 2023 [2]. Building of HESEB PLC control racks including PLC wiring, wiring diagrams, PLC coding and system commissioning have been done completely in-house by the control team and trainees.

PLC Programming The process of PLC programming begins with requirements analysis. Next, a matrix of causes and effects is defined, and finally, the PLC code is created and tested using this matrix. The PLC code structure is based on main routine, sub-routines and functions. This structure helps standardize the PLC programs and minimize programming errors. Moreover, it is easy to commission and understand, and flexible to upgrades.

EPICS Driver To connect the S7-1500 PLCs with EPICS, we used s7-nodave device support which is based on Asynchronous EPICS driver (ASYN) and libnodave library to communicate with S7 PLCs. s7-nodave does not require any special programming on the PLC side. Instead, the EPICS records just specify the memory address in the PLC to read or write the data [4].

Vacuum System

The control of vacuum system in HESEB consists of gate valves, TPG366 and TPG500 gauge controllers, IPCMini ion-pump controllers from Agilent and VAT fast valve controller. Remote monitoring and control over these devices is essential during operation of HESEB. The control interface

* anas.abbadi@sesame.org.jo

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THP04

EPICS TANGO BRIDGE

T. Madej, P. Goryl, M. Nabywaniec, Ł. Żytniak, S2Innovation, Kraków, Poland

Abstract

EPICS (Experimental Physics and Industrial Control System) [1] and Tango Controls [2] are the two most popular control systems for scientific facilities. They both have advantages and disadvantages. Sometimes there is existing software driver supporting integration only with EPICS or Tango. EPICS Tango Bridge is the perfect solution for accessing Tango devices using EPICS Channel Access protocol. Using our bridge, the cost of integration is significantly lower than providing dedicated integration of specific hardware in EPICS.

The bridge provides high reliability and robustness. Test cases created during the development process verify their limitations like the response time of reading one attribute in the bridge with a different number of Process Variables (PVs) or parallel access for multiple EPICS clients and many others.

OVERVIEW

Architecture

EPICS Tango Bridge provides access to Tango servers via the EPICS Access Channel. The architecture of the tool is shown in Fig. 1. We used the PyTango [3] library as the interface of Tango devices servers.

The project is based on the PCASpy library, which provides biding to EPICS Channel Access. The main components of EPICS Tango Bridge are pcaspy's SimpleServer and TangoDriver which are inherited from pcaspy's Driver class.

SimpleServer creates PVs based on the PVDB file and runs IOC (Input Output Controller). The server encapsulates transactions performed by the channel access server.

Driver Class is the base class that connects channel access requests to a real-world data source. The base class implementation simply stores a user-written value and retrieves it on request. TangoDriver, extending Driver class allows to read and write attributes of Tango Device Server.

TangoDriver during the initialization creates multiple Manager objects, which provide access to attributes and commands of specific Tango Device Server using the PyTango DeviceProxy class. There are multiple implementations of the Manager interface according to the specific usage (e.g., polled attributes, commands).

Each Manager object creates its DriverAdapter thanks to whom, write to PV (Process Variables) causes an update of value in Tango Device Server's attribute.

Usage

To run the bridge, we need to pass the corresponding PVDB file, providing a valid mapping between EPICS PV and Tango attributes and commands. That file contains a simple Python dictionary where the keys are the base names of the PVs, and the values are their configurations. TangoDriver creates multiple managers providing access to Tango Device attributes. Each manager implements the read and writes method. The write method is realized using the write thread. In that thread, the new value is assigned to the Tango Device Server attribute and then to the corresponding PV using the DriverAdapter object. If the Tango attribute is a two-dimensional list, it will be flattened during reading from the PV, i.e. [[1,2], [3,4]] will be read as [1,2,3,4]. During writing to the PV, the Bridge IOC will attempt to restore the correct structure before passing the data to Tango.

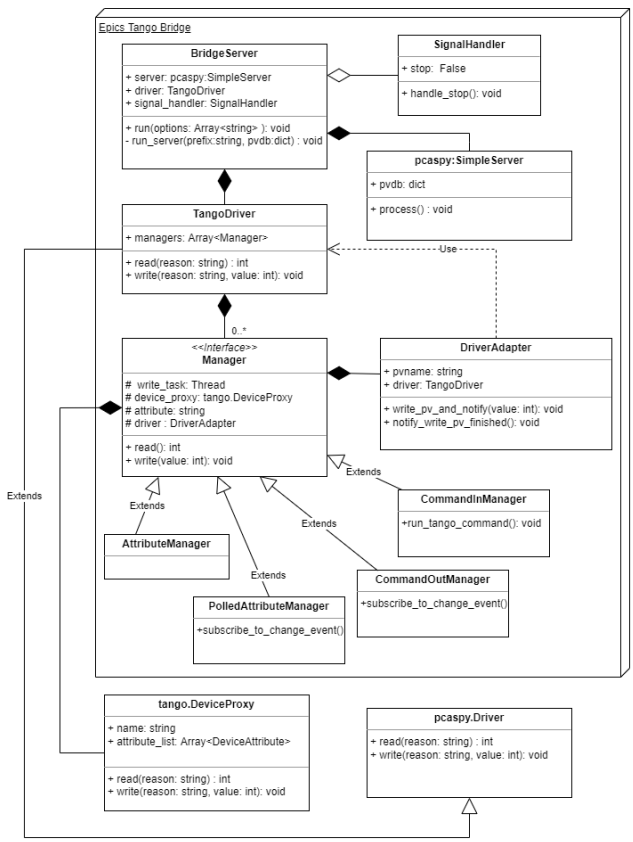

Figure 1: EPICS Tango Bridge Architecture.

STRESS TEST

To recognize EPICS Tango Bridge's capabilities and limitations, we prepared a set of stress tests.

In this section, we will describe the test scenarios as well as the test setup.

Test Scenarios

To test the performance of EPICS Tango Bridge, we used the PyTest [4] library. We prepared the following test scenarios:

1. Key Performance Indicators (KPIs) test – checking the maximum size of PVDB databases due to the type of attribute (double, spectrum double, string). Tests

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THP05

INTEGRATION OF QUENCH DETECTION SOLUTION INTO FAIR'S FESA CONTROL SYSTEM

Matic Marn, Andrej Debenjak, Cosylab d.d., Ljubljana, Slovenia
Michal Dziewiecki, GSI Helmholtzzentrum für Schwerionenforschung, Darmstadt, Germany

Abstract

Facility for Antiproton and Ion Research (FAIR) is going to make wide use of superconducting magnets for its components: the SIS100 synchrotron, the Superconducting Fragment Separator (SFRS) and Atomic, Plasma Physics and Applications (APPA) experiments. For all these magnets, uniform quench detection (QuD) electronics have been developed to protect them in case of uncontrolled loss of superconductivity. The QuD system will contain ca. 1500 electronic units, each having an Ethernet interface for controls, monitoring, data acquisition, and time synchronization. The units will be grouped into sub-networks of ca. 100 units and interfaced via dedicated control computers to the accelerator network. The interfacing software used to expose QuD functions to the FAIR controls framework is implemented as a Front-End Software Architecture (FESA) class. The software provides a solution for the constant collection of the data and monitoring of the system, storing the complete snapshot in the case a quench event is detected, and prompt notification of a quench to other components of the FAIR facility. The software is developed with special attention to robustness and reliability.

INTRODUCTION

Quench (uncontrolled superconductivity loss) events are considered a major threat in superconductor technology. During a quench of a superconducting magnet, the energy stored in the magnetic field is rapidly released in the quench area, which may cause severe damage. However, damage can be avoided by quench protection devices like quench heaters or dumping resistors. To activate them right in time, QuD systems are used.

FAIR's SIS100 synchrotron, the SFRS and APPA will employ a number of superconducting magnets for their operation [1]. For these magnets, voltage-measurement-based QuD methods are utilized: direct voltage measurement for current leads, bridge measurement for high-current magnets and pickup coils for low-current magnets [2]. Voltages over all superconducting components are monitored, mostly redundantly, by assigned QuD units. Each unit consists of a custom analog front-end (specific for each monitored part type), a set of comparators (quench detection relies on comparing acquired signals against pre-programmed thresholds), and circuitry for device control and data acquisition.

The data acquisition feature is not involved in quench detection itself, but collecting signals allows in-depth analysis of magnet operation and 'post-mortem' data are crucial for understanding magnet failures once they occur.

FAIR's control system is FESA-based, and the integration of the QuD units is done via a FESA class. The QuD FESA class was designed to control up to 96 QuD units by a single equipment class, however, there is no technical limitation to extend this number further as far as enough processing and memory resources are allocated to the hosting computer.

FESA is a C++ framework used for developing real-time (RT) software for devices that have to be integrated into a control system – software that controls and monitors accelerator equipment and performs data acquisition. It comes with a rich environment for designing, developing, deploying, and testing. It offers wide possibilities in data acquisition, data handling, and standardized generic interfaces called properties, which can be specified for each device being integrated. An acquisition property allows for acquired data to be published (i.e., notified) to the users, while command and setting properties facilitate the user to provide input to a FESA class. In addition, FESA provides tight and seamless integration of the FAIR's White Rabbit [3] timing system and focuses on real-time execution.

FAIR QUENCH DETECTION UNITS

Each QuD unit can be divided into two parts: the analog QuD circuitry (which is virtually independent of the control system or any other high-level electronics) and a module for device control and data acquisition. The latter is built around a Cyclone IV FPGA and runs a NIOS-II soft-core with FreeRTOS-based software. Its functions range from simple operations (like remote forcing or resetting of the quench signal) through controlling quench thresholds to continuous 10kHz signal acquisition. Further, it provides numerous diagnostics features and remote updates.

Digital Interface

All units are controlled via a fast Ethernet interface running multiple protocols on top of IP v.4. For generic device control, the Simplified Universal Serial Interface (SUSI) protocol [4] has been developed, which utilizes a variable size register model for device programming. For data download, a custom real-time data transfer protocol has been implemented directly in FPGA hardware to offload the softcore. Further, some generic protocols (DHCP, mDNS) are used to support IP connectivity.

It's worth noting that the quench signal is transmitted over a dedicated digital line and it's independent of the network interface.

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THP07

A MODERN C++ MULTIPOCESSING DOOCS CLIENT LIBRARY IMPLEMENTATION

S. Meykopff†, Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

Abstract

At the DESY site in Hamburg/Germany the linear accelerators FLASH and European XFEL are successful operated by the control system DOOCS. DOOCS based on the client–server model and communicates with the matured SUN-RPC. The servers are built with a framework which consists of several C++ libraries. The clients use a DOOCS client library implementation in C++ or Java. In the past years the public interface (API) of the C++ client library was refined. But modern C++ features like futures are not provided in the API. Massive multi-processing, parallel communication, and optimized names resolution could improve the overall communication latency. The usage of the standard C++ library, the limit of external dependencies to ONC-RPC (former SUN-RPC) and OpenLDAP, and the reduction of the code size, increases the maintainability of the code. This contribution presents an experimental new client C++ library which achieves these goals.

BIRDS EYE ON DOOCS

The DOOCS control system uses the proven ONC-RPC (formerly SUN-RPC) for data transmission. To establish a connection to an ONC-RPC server, you need a host name and a port number. A DOOCS address must therefore first be resolved into a combination of host name and port number. This resolution is started via an LDAP query. If the server locations are served via multiple servers, all required information is available in the LDAP response. If all locations are answered via a single server, the existing locations can be queried via an RPC call to these servers.

Address Resolving with LDAP

The LDAP connection is established with OpenLDAP. First a connection to the LDAP server must be bound via ldap_simple_bind_s(). The single queries are then made with ldap_search_ext_s(). Depending on which entries are searched for, the distinguished name (DN) must be created at runtime. The DN consists of the following Relative Distinguished Names (RDN) "location", "device", "facility", "dc". The filter consists of a string with a "server-mask" and partly "device" as logical link. The last parameter to be specified is the scope in which to search. This also depends on which DOOCS address part is searched for. The connection to the LDAP server can be reused until all address resolutions are finished. At the latest at the end of the program the connection must be closed with ldap_unbind(). A sequence diagram is shown in Figure 1.

LDAP Answer

The LDAP query response contains these fields: "facility", "device", "location", "property", "server", "host", "channel", "protocol", "lib-prog", "server-mask", "status". The first four are part of the DOOCS address. The fields "host" and "lib-prog" are needed to connect to a DOOCS server. The two fields contain the above mentioned connect information for the RPC call. When searching for locations, the "property" field remains empty.

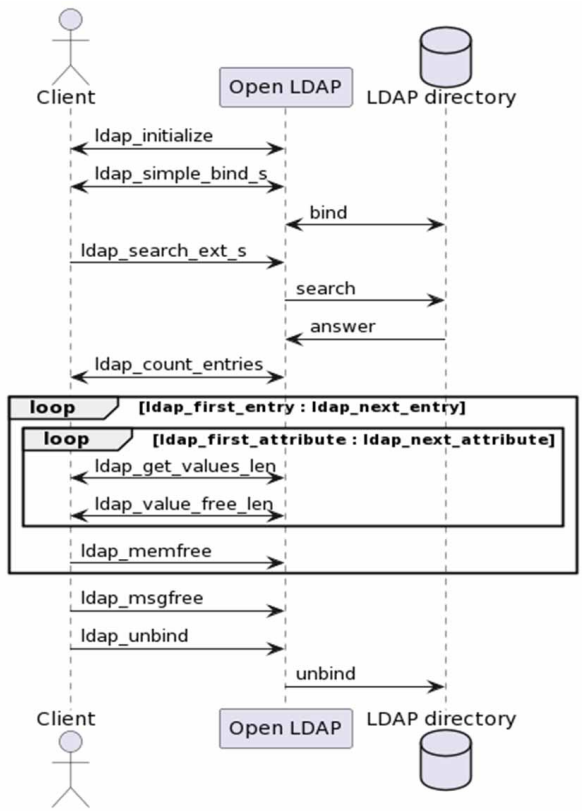

Figure 1: Address Resolving.

Authorization

To establish an RPC connection, authorization must first be performed. The authorization is provided by the authunix_create() function. The function requires the target host name, the (Unix) user ID and the group ID of the person to be logged in as parameters. The authorization is valid for all threads of a task until the auth_destroy() function deletes it again.

RPC Bind

A connection to the target server will be opened with clnt_create(). The function needs the host name as parameter and the port number determined above as parameter. The choice whether UDP or TCP is used as protocol must now be made. However, UDP is not used in DOOCS because the RPC messages may otherwise have a maximum of 8 kbytes of coded size. The function expects the protocol type as string. After the connection has been established,

† sascha.meykpoff@desy.de

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THP09

SMART VIDEO PLUG-IN SYSTEM FOR BEAMLINE OPERATION AT EMBL HAMBURG

E. Galikeev†, U. Ristau, C. Blanchet, D. v. Stetten, V. Palnati, S. Fiedler
all European Molecular Biology Laboratory (EMBL), Hamburg Unit
c/o DESY, Hamburg, Germany

Abstract

Fast data collection, image processing, and analysis of video signals are required by an increasing number of applications at the EMBL beamlines for structural biology at the PETRA III synchrotron in Hamburg, Germany.

Consequently, a new Smart Video Plug-in system has been designed in-house to meet the needs by combining video capture, machine learning, and computer vision with online feedback for motion control. The new system is fully integrated into TINE: data acquisition, and experiment control system.

In this paper, the architecture of the new video system is described and use cases relevant to beamline operations are presented.

INTRODUCTION

Recent projects carried out at EMBL Hamburg synchrotron beamlines for structural biology showed the necessity for a flexible tool that combines modern image processing libraries, video codecs, and motion control support for such experimental workflows like:

- Sample positioning using piezo stage control
- Control of the focus of the sample observation optics
- Image processing (e.g. filtering, denoising)
- Object detection and image recognition using machine learning techniques.
- Fast video recording and compression for offline and online sample motion analysis

The usage of the OpenCV library for image detection and computer vision at synchrotron beamlines was studied by Gofron and Watson [1] who showed that it can be applied for sample detection and centering robotic mounting, and evaluation of x-ray beam properties.

TINE at PETRA III [2] is a cross-platform, scalable distributed control system covering the majority of beamline operations and can be used with various OS like Windows, MAC/OS, and Linux. One of the core concepts of the TINE system is the "TINE Server" which represents a unique entity in the control network. It exports properties as endpoints used by the other TINE servers and clients to communicate with each other.

The new video system uses the full potential of the TINE API to provide means for the building and deployment of a new server as well as utilizing the centralized TINE alarming and archiving functionalities.

ARCHITECTURE

There are several reasons for choosing the Microsoft .NET framework [3] as a platform for integrating 3rd party functionality and user algorithms into the TINE system:

- .NET is a free and open-source, cross-platform software development framework.
- It enables interoperability between native C/C++ to preserve and take advantage of existing investments in unmanaged code.
- There exists wide official API support of the .NET platform among motion control and video solutions manufacturers.
- TINE .NET API allows getting full access to the EMBL beamline control system [4].
- C# is a powerful, modern, object-oriented type-safe programming language for rapid application development.
- Bindings for the NumPy and OpenCV libraries as well as such C# language features like LINQ provide extensive data processing toolsets for the experiment scripts.

The plug-in architecture [5], as shown in Fig.1, was chosen to meet the needs of the flexibility of adding new functionality (e.g. new camera or motion controller API support). The plug-in modules are meant to be separate containers for the additional features or user code that is designed to enhance the main functionality.

The core system provides interfaces to which plugins can connect, creating TINE server instances and routing data exchange between the plug-ins, and offering external communication with clients using standard TINE data transfer mechanisms. Also, the core system provides logging, native 3rd library support, and system configuration functionality.

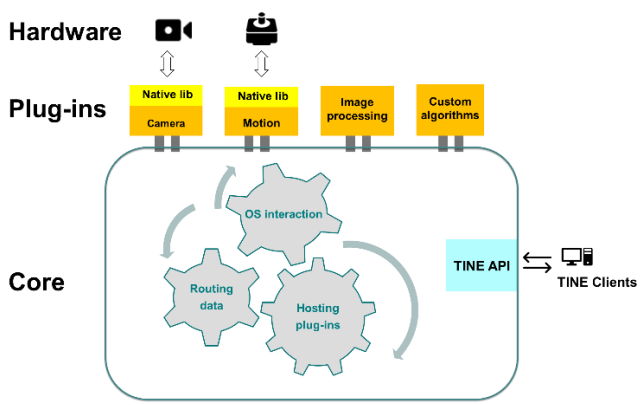

Figure 1: Architecture of the Smart Video System.

†emil.galikeev@embl-hamburg.de

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THP12

PLC OPERATED PLUG AND PLAY VACUUM GAUGE FUNCTIONALITY AT THE ARGONNE TANDEM LINEAR ACCELERATOR SYSTEM*

K. Bunnell†, C. Dickerson, B. Nardi, D. Novak, D. Stanton
Argonne National Labs, Chicago, IL, USA

Abstract

ATLAS (Argonne Tandem Linear Accelerating System) accelerator at Argonne National Laboratory is upgrading the vacuum control system from hardware-based, embedded controllers to modern flexible PLC-based controllers. This PLC (Programmable Logic Controller) system includes additional fail safes and a new remote operation feature. As part of this upgrade, enabling easy vacuum equipment replacement became apparent, specifically the vacuum gauges which are interfaced using serial communications. We developed a solution to remotely program the gauges, which offers more options for gauge control than the hardware-based controllers. Expanding on this, we added a process to initialize and restore the configuration for replaced gauges. This simplifies the process and eliminates the need for a system expert for these tasks.

INTRODUCTION

The ATLAS accelerator is located at the United States Department of Energy's Argonne National Laboratory in the suburbs of Chicago, Illinois. It is a National User Facility capable of delivering ions from hydrogen to uranium [1] for low energy nuclear physics research to perform analysis of the fundamental properties of the nucleus.

The accelerator requires pumps to create a vacuum in the beamline to prevent collision of the beam with air particles. Most of the vacuum control system uses custom built hardware-based chassis (Figure 1) that provide static vacuum interlock logic. The interlock logic largely relies on the set points of vacuum gauges to determine when safe operating conditions have been exceeded so that pressure-sensitive pumps can be isolated or stopped. The set point values can be locally adjusted from the gauge controllers. These systems are connected to a CAMAC (Computer Automated Measurement and Control) Serial Highway [2] to provide basic vacuum information to the control system such as: absolute vacuum pressures, pump status, and valve status.

Since 2019, ATLAS has been replacing the hardware-based vacuum monitoring system with flexible Schneider Electric PLC-based systems. The new systems enable remote vacuum control (Figure 2), flexible logic and vacuum hardware additions, and fully automated vacuum systems; all of which were cumbersome with the hardware-based system [3].

A goal of the ATLAS vacuum upgrade is to integrate vacuum automation. Since safety checkpoints are programmed in automated vacuum systems, there is less knowledge required to operate the system. At the same time, the new vacuum upgrade should have all the features of the hardware-based system. In this paper we will focus on seamless gauge replacement and set point configuration.

HARDWARE DESCRIPTION

The existing ATLAS vacuum system includes many obsolete vacuum gauges which feature manual set point adjustments from a gauge controller. The replacement devices largely include faster responding vacuum gauges with larger vacuum ranges, more features, and remote communication interfaces. To minimize the upgrade efforts and to keep all the former features in place, the obsolete vacuum gauges are being replaced by modern Granville-Phillips (GP) 275 and 390 gauges as the sectioned PLC upgrades take place. The modern gauges are modular and have integrated controllers. Therefore, when a gauge is replaced, all programmed information from the old gauge must be sent to the new gauge.

SOFTWARE DESCRIPTION

The PLC used for this vacuum upgrade features RS232 and RS485 serial communication natively. This is important since the majority of the new GP vacuum gauges require RS485 serial communication as their only interface to program the gauge, and read vacuum pressures. The default factory serial communication settings for all gauges are the same and will not be changed for use with the PLC.

Figure 1: Hardware-based vacuum Control Chassis with adjustable vacuum set point switches.

Gauge Addressing

Each GP vacuum gauge has a manual, 16-position adjustment of 0-F for device addressing. In addition, there is a programmable address offset that can be changed to a value of 0, 10, 20, or 30 (HEX). For PLCs with more than 16 gauges, the address offsets will be required to prevent duplicate addresses. With address offsets, a maximum of

* 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.
† kbunnell@anl.gov

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THP13

CONTROL AND TIMING SYSTEM OF A SYNCHROTRON X-RAY CHOPPER FOR TIME RESOLVED EXPERIMENTS

U. Ristau, V. Palnati, J. Meyer, D. Jahn, S. Fiedler
European Molecular Biology Laboratory, EMBL, Hamburg, Germany

Abstract

A mechanical X-ray chopper for the Small Angle Scattering (SAXS) beamline P12 operated by the EMBL Hamburg Unit at the PETRA III synchrotron on the DESY campus has been developed. In this paper we will describe how control and timing for time resolved experiments have been implemented.

INTRODUCTION

In the time resolved SAXS experiments carried out at P12, the structural changes of proteins in solution are studied. The scattering of the sample is recorded with a fast area detector as a function of time after a reaction has been triggered. As the samples are sensitive to radiation damage, it is desirable to minimize the exposure to the X-ray beam between two frames. This is achieved with the chopper that can be adjusted to variable duty cycles.

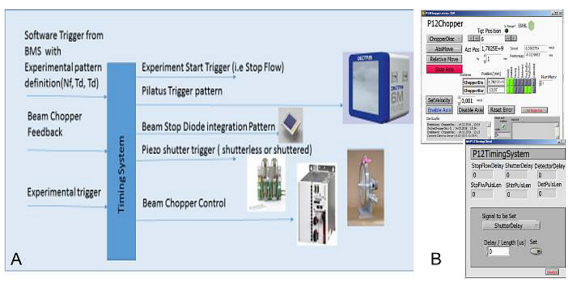

Figure 1: (A) Schematics of instruments to be synchronized for an experiment. (B) Clients for the timing control of the experiment and of the chopper [1].

Also pump and probe measurements can be carried out with chopper systems where the sample is probed with a short X-ray pulse after a laser excitation [2]. For such experiments several systems or processes must be synchronized like storage ring bunch pattern, sample injection, excitation laser pulse, intensity monitor signal, shutter opening, detector trigger, and the chopper revolution frequency that must be kept very constant (Fig. 1A). The solution chosen for the timing and synchronization system is based on EtherCAT electronics [3]. Servers and clients are integrated into the TINE control system [4] with LabView [5] (Fig. 1B) and TwinCAT [6].

CHOPPER DESIGN & IMPLEMENTATION

Different designs for X-ray beam choppers used at synchrotrons have been suggested ranging from the milli- to the sub-microsecond range (see for example [2, 7]).

Our beam chopper design (Figs. 2A, 2B) is based on a rotating disk that has a 10-fold symmetry cut-out pattern with stepped slots of different length in angular direction (Fig. 2C). When the disk is rotating the beam is chopped as a function of

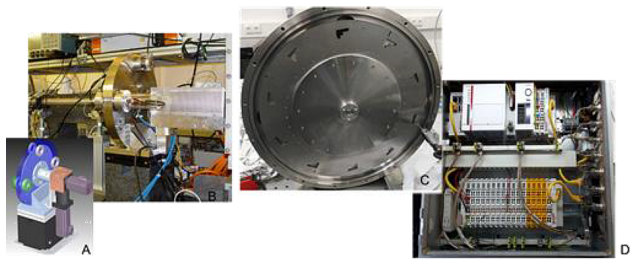

Figure 2: (A) Mechanical design of the chopper, (B) Photograph of installation at P12 beamline, (C) Photograph of the chopper disk, (D) Fieldbus electronics for chopper control.

the length of the slots. The chopper can be translated laterally so that the X-ray beam can be transmitted at positions with different slot lengths. In this manner variable duty cycles can be adjusted. The chopper can also generate short X-ray pulses. For this, a narrow slot at the outer diameter of the disk is used. The pulse length can be varied by adjusting the angular velocity.

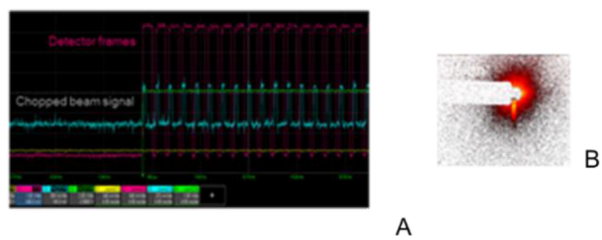

Figure 3: (A) Chopped beam signal at 750 fps measured with an intensity monitor at the detector with ¼ duty cycle. (B) Residual parasitic scattering pattern in the SAXS region of the chopped beam at 750 Hz.

Examples of the system in operation are depicted in Fig. 3. The chopper can run with a beam chopping frequency of up to 1500 Hz. A frequency stability of $<10^{-5}$ can be achieved and the jitter is below 2 µs. The shortest X-ray pulse that can be produced is 10 µs.

CHOPPER CONTROLS

The TINE Control system is used as transport layer at the EMBL Hamburg beamlines at PETRA III [1, 8–10]. All server and client communication are based on the TINE protocol. The TINE CDI (Common Device Interface) [11] has

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THP15

NEXT GENERATION GSI/FAIR SCALABLE CONTROL UNIT: LESSONS LEARNED FROM 10 YEARS IN THE FIELD

K. Lüghausen, M. Dziewiecki, K. Kaiser, G. May, S. Rauch, M. Thieme, GSI, Darmstadt, Germany

Abstract

The end-of-life of many components brought the need for a redesign of the main Control System Front-End for GSI accelerators - the SCU (Scaleable Control Unit). It was a chance to make improvements and use more powerful state-of-the-art core components. This included a new Arria 10 FPGA and a completely redesigned housekeeping circuit based on an AVR micro controller. Further, the project was cleaned by removing unused components and features. Main frame conditions stay fixed for backward compatibility, like the mechanical form factor or the 16-bit parallel bus. Majority of gateware and firmware could be reused and just some adaptations for the new FPGA were needed. Nevertheless, providing continuous compatibility with legacy peripherals needed a substantial effort.

INTRODUCTION

Scaleable control units (SCUs) have been designed as a uniform, intelligent, realtime interface between various electronic components of accelerators and the control network. Internally, they contain an off-the-shelf computing module (main CPU, main memory) running Linux and a custom baseboard (I/Os, White-Rabbit timing receiver) with an Intel Arria FPGA as summarized in Table 1. The integrated timing receiver allows executing various tasks with nanosecond-level precision while the main CPU covers higher-level computing and data management. On the network side, they offer a Gigabit Ethernet interface for communications and a White-Rabbit slave for timing and triggering. On the device side, they are equipped with a parallel bus and some further interfaces as shown in Fig. 1. The device is disc-less: all data, software and the boot image are delivered via the network.

The new generation, the SCU 4, keeps compatibility with previous versions while providing significantly more resources.

SCU4 HARDWARE DESIGN

Table 1: FPGA Comparison

Parameter	SCU3	SCU4
FPGA Type	Arria II	Arria 10
Technology	40 nm	20 nm
Logic Elements	100 k	270 k
BlockRAM	8121 Kb	17 452 Kb

The general design was inherited after the previous SCU generation [1]. The basic parameters like mechanical dimensions and electrical connection to the back-plane remained unchanged.

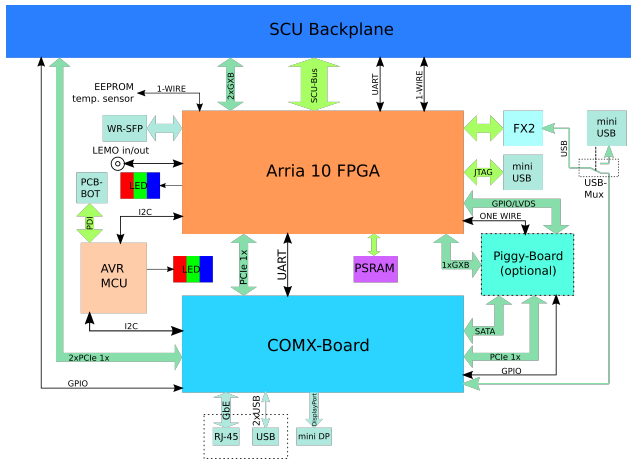

Figure 1: Block diagram of SCU4.

Major changes took place regarding the front panel. An OLED-display was removed. It turned out to be of limited use while it needed significant effort in assembly and occupied valuable space. Instead, a DisplayPort socket was added which allows connecting an external screen and a RGB-LED indicates the system status.

An important feedback from field operation was the lack of User-I/Os. As a result we added four additional user I/O lines. A dual mini D-Sub 9 debug connector was replaced by a mini USB device connector. A summary of these changes is shown in Table 2.

Table 2: SCU IOs

IOs	SCU3	SCU4
SCU-Bus	1	1
SFP	2	1
GbE Ethernet	1	1
USB 2.0 Host	2	2
USB 2.0 Device	–	1
Serial	2	–
Video	–	miniDP
LEMO 5V-TTL	2 In/Out	2 In, 4 Out
Logic-Analyser	1	–

The form-factor of the computing module has been changed from COMX Type 2 to Type 10 [2]. With a smaller footprint, it allowed us to rotate the module by 90 degrees, giving better airflow to cool the FPGA. The new module brings also advantages in processing power, as shown in Table 3.

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THP16

OCELOT INTEGRATION INTO KARA'S CONTROL SYSTEM

P. Schreiber*, E. Blomley, J. Gethmann, W. Mexner, M. Schuh, A.-S. Müller
Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract

The Karlsruhe Research Accelerator (KARA) at the Karlsruhe Institute of Technology (KIT) is an electron storage ring and synchrotron radiation facility. The operation at KARA can be very flexible in terms of beam energy, optics, intensity, filling structure, and operation duration. For different aspects of the operation of the accelerator separate and individual simulation models are in place using different simulation tools, custom lattice data and varying levels of maintenance. In a general push at the accelerator to provide unified access via Python, a new framework was implemented using Ocelot with a much closer integration to the accelerator control system and supplementary tools. This allows a better integration and lowers the effort necessary for simulations and predictions of actual changes to the beam properties based on live data. It also provides a good entry point for the various Python based machine learning activities at the accelerator and the goal to obtain an easier to maintain and test accelerator model. This paper presents the taken approach and current status of this project.

INTRODUCTION

The storage ring KARA (Karlsruhe Research Accelerator) is operated as an accelerator test facility and serves as the KIT light source. It is a ramping electron synchrotron from 0.5 GeV to 2.5 GeV with a 4 fold symmetry and a total of 8 double bend achromatic cells. The circumference of 110.4 m and frequency of the RF system of 500 MHz results in a bunch revolution time of 368 ns and a bunch spacing of 2 ns. Furthermore, a wide range of beam diagnostics tools is available. A lot of beam time is dedicated to machine physics including experiments performed by scientists from external institutions. KARA is available as R&D facility for TNA (TransNational Access) via Eurolabs [1]. In order to make best use of the beam time simulations beforehand are necessary. Therefore easy access and ideally pre-set-up simulations are required.

REQUIREMENTS

Performing simulations for an accelerator, especially for new students or external scientists working at KARA can be hard to set up as many details such as exact lattice parameters as well as machine specific conversion coefficients are mandatory. Therefore the integrated simulation should be easy to use and have reasonable defaults. It should furthermore allow a reasonable amount of customisations to adapt the simulations to the various needs.

Furthermore, future systems might rely on e.g. the optics functions for the currently used settings at KARA. Such systems could consist of machine learning applications for various tasks such as operation optimisation. For these systems the simulation results should be readily available via network connections. The simulations for such systems should also be run periodically to ensure recent values.

A third requirement for the integration is easy maintain- and extendability. Additional simulation features, such as tracking or new elements should be easily implemented in order to encourage extension of the system.

SIMULATION SETUP

The simulation setup consists of a lattice defined with LatticeJSON [2], an epics-IOC [3] with records for each element or family, ocelot [4] as simulation software and a control panel in CSS [5]. An overview of the involved objects and their connections is shown in Fig. 1.

From the LatticeJSON definition an intermediate object is created that acts as uniform interface between the lattice definition, the IOC and the actual magnetic lattice for ocelot. This approach was chosen in order to allow potential other simulation tools to make use of the same lattice input. For each element in the lattice, the intermediate object creates an element type specific object. These element objects are responsible for transforming the user facing values of current flowing through the magnets to magnet strengths and also for creating the elements used by ocelot.

The intermediate lattice object is used by the automatic creation of an epics-IOC. A record in the IOC is created for each specified PV of each element in the LatticeJSON definition (this allows for multiple PVs per element if necessary).

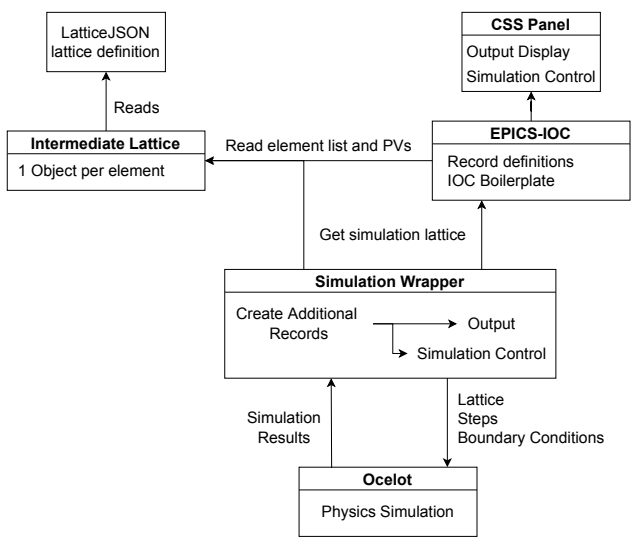

Figure 1: Structure and connections between different objects involved in the simulation integration system.

* patrick.schreiber@kit.edu

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THP18

STATUS, RECENT DEVELOPMENTS AND PERSPECTIVE OF AVINE VIDEO SYSTEM

Stefan Weisse*, David Melkumyan, DESY, Zeuthen, Germany
Philip Duval, DESY, Hamburg, Germany

Abstract

DESY's TINE-powered Video System, originally released in 2002, was last presented in 2011 at ICALEPCS [1], at this time not yet known under the name Advanced Video and Imaging Network Environment (AVINE). AVINE provides a framework and toolkit for operators, physicists and technicians related to, but not limited to, Ethernet-based imaging at accelerator facilities. Over the past decade, the major emphasis was put on extended support, incorporating user requests, migrating to the latest Windows and Linux operating systems and the latest Java Virtual Machine, all while replacing legacy GigE Vision APIs in order to support past, current and future camera hardware. In this contribution, the current status, layout, recent developments and perspective of AVINE is described. The focus will be on experience migrating to future-oriented (still under vendor support) GigE Vision APIs, the recently upgraded image (sequence) file format, and first experiences on Windows 11.

OVERVIEW

The Video System explained further on originates at the Photo Injector Test Facility at DESY in Zeuthen (PITZ) [2]. Started in 2002 as a test facility at DESY for research and development on laser-driven electron sources for Free Electron Lasers (FEL) and linear colliders, PITZ has been extended over the last years in order to study different applications of photo injectors, e.g. for the generation of THz SASE light [3]. A significant milestone, first emission of THz light from the SASE FEL, has been reached in August 2022 [4].

Despite its to some extent different requirements, AVINE has been exported over the years to accelerator setups on DESY campus in Hamburg, namely PETRA III (e.g. machine diagnostics [5], user beamlines and so-called beam paths E-Weg and L-Weg), REGAE, SALOME and User Beamlines of Petra III at EMBL Hamburg. In addition, small installations have been realized over the years as well, e.g. monitoring and positioning of an electron beam at an electron welding machine in mechanics workshop and standalone AVINE video installations on notebooks for live video, data taking and analysis, which can also be used offline/off-site.

Ethernet-based industrial vision cameras which are conforming to standards GigE Vision and GenICam can easily be connected and used. In general, image acquisition is done with a repetition rate of 1 to 20 Hz, either initiated by external trigger signal (TTL pulse) or by an internal clock of the camera. Most cameras are greyscale (non-colour) with 8 to 12 bits per pixel. AVINE is integrated into the TINE control system [6]. Network transport of video image stream is done via TINE, either via TCP or multicast. To use available network bandwidth more effectively, JPEG compression can be applied on network transport. As a trade-off, greyscale pixel bit depth is limited to 8 bits when using JPEG.

Image size varies from hundreds of kilo- to several megapixels. Colour images (RGB 24 bits) are supported, however client GUIs and applications for measurements and analysis are mainly designed for greyscale data.

The Universal Slow Control System (USC) is designed to control various types of cameras used in the AVINE Video System, providing a common approach to configuring, displaying and controlling camera slow control settings required for PITZ operation, e.g. exposure time and analogue video signal gain. The USC system consists of a USC client, a USC server, and a USC Server Configurator implemented in Java. The USC Client is a TINE-based GUI client application that includes the functionality required by an operator to monitor and control the settings of all available cameras in the Video System. The USC Server is a TINE-based server that provides slow control of various types of cameras using various communication protocols (e.g. RS-232 for analogue cameras, TINE-based for controlling Gigabit Ethernet cameras via AVINE SGP servers). The USC Server Configurator (USCSC) is a GUI application for easy configuration of the USC Server, allowing server maintenance personnel to use predefined templates, modify them, or create a new template from scratch. It also checks server configurations and displays appropriate errors and troubleshooting tips to avoid unexpected USC server behaviour at runtime.

A versatile easy-to-use albeit aged Windows client application called Video Client 3 can be used for live view, data taking, measurements and on/off-line analysis.

With regards to platform independence, a JAVA component called TINE ACOP Video bean is available, which acts as a basis to create client-side GUIs with special functionality based on the experimental setup. The Video bean is also integrated into the TINE Instant Client, a basic Java GUI to browse, read and write TINE properties.

LAYOUT

The general layout of AVINE is shown in Fig. 1. AVINE is following a component (object-oriented) based approach. The idea is to hide implementation details and only expose what is necessary, so that changes in implementation are mainly kept inside a single component. For example, migration to a different API to interface cameras is only reflected in the front-end server (so-called Small Grabber Part (SGP)).

* stefan.weisse@desy.de

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THP20

PYTHON BASED INTERFACE TO THE KARA LLRF SYSTEM

E. Blomley*, J. Gethmann, P. Schreiber, M. Schuh, W. Mexner, A. Mochihashi, A.-S. Müller
Karlsruhe Institute of Technology, Karlsruhe, Germany
S. Marsching, aquenos GmbH, Baden-Baden, Germany
D. Teytelmann, Dimtel Inc., San Jose, California, USA

Abstract

The Karlsruhe Research Accelerator (KARA) at the Karlsruhe Institute of Technology (KIT) is an electron storage ring and synchrotron radiation facility. The operation at KARA can be very flexible in terms of beam energy, optics, intensity, filling structure, and operation duration. Multiple digital low-level radio-frequency (LLRF) systems are in place to control the complex dynamics of the RF cavities required to keep the electron beam stable. Each LLRF system represents a well established closed system with its own set of control logic, state machine and feedback loops. This requires additional control logic to operate all stations together. In addition, during special operation modes at KARA, extra features such as well defined beam excitation are needed. This paper presents the implementation of a Python layer created to accommodate the complex set of options as well as an easy to use interface for the operator and the general control system.

RF SETUP OF KARA

KARA makes use of two RF stations controlled by one LLRF system each. Each RF station consist of a klystron which feeds two cavities utilizing a magic T with a fixed 3dB splitting ratio. In addition, the smaller booster synchrotron is powered by one cavity with an additional LLRF system. The electron beam in KARA is accumulated at 0.5 GeV with a 1 Hz injection cycle in the booster synchrotron. The accumulation period can typically last up to one hour. After the injection is finished, the beam energy of the storage ring is slowly increased to its final operating energy. Depending on the final energy, the energy ramp lasts up to three minutes. Therefore, in our common operation scheme, the demands of the RF system differ substantially between the booster synchrotron and the storage ring. The first operates on a fast, pre-defined and synchronized voltage ramp, whereas the approach for the storage ring is to explicitly set the RF voltage level to accommodate for the increase in beam energy as the energy slowly increases. Table 1 shows a summary of the RF system characteristics.

Table 1: RF System Characteristics

Parameter	Booster	Storage Ring
f_{RF} (MHz)	500	500
Voltage Range (kV)	5 - 25	300 - 1500
Energy Range (GeV)	0.05 - 0.5	0.5 - 2.5
Ramp Duration (s)	0.8	170
LLRF Stations	1	2
Cavities	1	4

Digital LLRF System

Modern, digital LLRF systems take care of operating the RF cavities by controlling the amplitude and phase of the RF waves, while also providing access to data points to read out signals, configure the relevant parameters and operate the RF station in general. At KARA the LLRF9 [1] from Dimtel was installed in 2016 with the booster synchrotron following in 2017. Each LLRF system has up to 9 RF inputs as well as the option for additional slow inputs to, for example, monitor the cavity vacuum levels. The main internal signal processing takes place on a field-programmable gate array (FPGA) surrounded by a Linux operating system providing slow access via the EPICS control system [2], which is also the main control system of all accelerators at IBPT. The EPICS interface allows access to around 600 process variables. This includes access to a stored ramping curve, captured waveforms of the fast inputs and an integrated network analyser, among other features. The system comes with a full set of graphical user-interface (GUI) panels (see Fig. 1).

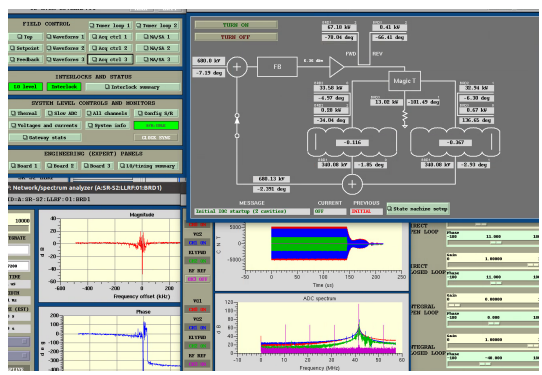

Figure 1: Native LLRF GUI. Visible is the top level control, the network analyser and the waveform acquisition tools.

USE CASES

The embedded EPICS controls and panels do an excellent job for what they were designed for, namely commissioning, configuring and monitoring the LLRF system. Still, there are several reasons as to why an additional software layer surrounding the core LLRF system might be useful.

Daily Operations

How the LLRF system is used in daily operations might be quite different from accelerator to accelerator or in this case even between the storage ring and the booster synchrotron.

* edmund.blomley@kit.edu

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-THP22

USING REACT FOR WEB-BASED GRAPHICAL USER APPLICATIONS FOR ACCELERATOR CONTROLS

R. Bacher, J. Szczesny
Deutsches Elektronen-Synchrotron DESY, Germany

Abstract

Today, control applications need to run on a variety of different operating systems, including Windows, Linux, and Mac OS, but also Android and iOS. Programming languages like Java have tried to solve this problem in the past by providing a common runtime environment. However, this approach is insufficient or even unavailable for mobile devices such as tablets and smartphones. Another problem is the different form factors of mobile and desktop devices, which makes it difficult to develop portable applications. One way out of this dilemma is to use standard web technologies (HTML5, CSS3, and JavaScript) to implement applications that run in the browser, which is available for all platforms. Modern JavaScript web application frameworks combined with JavaScript graphics libraries such as D3 are suitable for building both very simple and very complex web-based graphical user applications. This paper reports on the status and issues that we encountered in our current developments with React.

LEGACY: WEB2CTOOLKIT

At DESY, the development of web-based user applications for accelerator operation started 2005 during the conversion of the high-energy booster synchrotron PETRAII into the synchrotron light source PETRAIII. In the following years, the so-called Web2cToolkit framework [1] was further implemented and its functionalities were extended step by step.

The Web2cToolkit is a collection of web services including

(1) *Web2c Synoptic Display Viewer*: Interactive synoptic live display to visualize and control accelerator or beam line equipment,

(2) *Web2c Archive Viewer*: Web form to request data from a control system archive storage and to display the retrieved data as a chart or table,

(3) *Web2c Messenger*: Interface to E-Mail, SMS and Twitter,

(4) *Web2c Logbook*: electronic logbook with auto-reporting capability,

(5) *Web2c Manager*: administrator's interface to configure and manage the toolkit,

(6) *Web2c Editor*: graphical editor to generate and configure synoptic displays, and

(7) *Web2c Gateway*: application programmer interface (HTTP-gateway) to all implemented control system interfaces.

Web2cToolkit is a framework for Web-based Rich Client Control System Applications. It 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 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. Registered and so-called privileged users have more rights compared to ordinary users (read-only permission).

The Web 2.0 paradigms and technologies used include a Web server, a Web browser, HTML (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 Web browser. The interface is compatible with 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 [2], DOOCS [3], EPICS [4] and TANGO [5] as well as STARS [6]. The toolkit is capable of receiving and processing JPEG-type video streams.

The Web2cToolkit is a proprietary framework that is not built on widely used JavaScript toolkits such as Dojo [7]. Due to progressing standardization efforts and the impressive development speed of modern web technologies, the Web2cToolkit is outdated and hardly maintainable in terms of performance and usability. Therefore, the development has recently been discontinued and only bug fixes are being made.

Figure 1: Hybrid app architecture [8].

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR011

Taskomat & Taskolib: A VERSATILE, PROGRAMMABLE SEQUENCER FOR PROCESS AUTOMATION

L. Fröhlich*, O. Hensler, U. Jastrow, M. Walla, J. Wilgen
Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

Abstract

This contribution introduces the Taskolib library, a powerful framework for automating processes. Users can easily assemble sequences out of process steps, execute these sequences, and follow their progress. Individual steps are fully programmable in the lightweight Lua language. If desired, sequences can be enhanced with flow control via well-known constructs such as IF, WHILE, or TRY. The library is written in platform-independent C++ 17 and carries no dependency on any specific control system or communication framework. Instead, such dependencies are injected by client code; as an example, the integration with a DOOCS server and a graphical user interface is demonstrated.

INTRODUCTION

Like other scientific and industrial facilities, particle accelerators consist of many subsystems that need to be coordinated. The operation of any such complex system is inevitably governed by *processes* of various kinds. These processes can be implicit ("operators typically do it like this"), explicit (checklists and step-by-step instructions), or automated (implemented in hard- or software). Automated processes have several advantages over their manual counterparts:

- Faster execution
- Better reproducibility
- Reduced work load for operators
- Improved documentation of what happened when

The particle accelerator community has traditionally been in a good position to profit from automation in software because of its early adoption of distributed control systems. Tools to execute process steps automatically, so-called *sequencers*, have a long history [1–7]. Even today, this field keeps spawning new developments due to changing requirements and user expectations [8–10].

At DESY, the main tool for the automation of processes for more than a decade has been the aptly named *Sequencer/File Operator* [11]. It can execute linear sequences of steps with limited support for branches and jumps between the steps. Each step can write values to the control system or read them until certain conditions are fulfilled. Apart from this, the tool also provides save&restore functionalities for control system properties.

The *Sequencer* consists of a Java client with a graphical user interface (Fig. 1) and a Java server component that mainly provides central data storage and version control. Although the tool has proven its usefulness across practically all of DESY's accelerator facilities, several flaws have become apparent over the years:

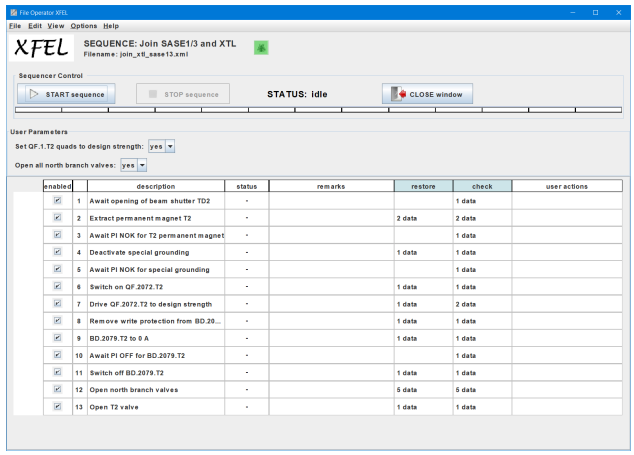

Figure 1: A procedure in the legacy DESY sequencer.

- The sequences are written as XML files with a poorly documented set of tags and attributes. Users typically have to interact with a Subversion repository to edit these files. This presents a high entry bar for creating or maintaining sequences, so that this task is left to very few experts and a lot of potential for automation is wasted.
- The kinds of sequences that can be expressed are limited: Control flow can only be directed through conditional *goto* statements, and there is no support for variables or arithmetic or string operations.
- The client-server communication is based on the TINE protocol [12] which has reached end-of-life and is slowly phased out.
- Sequences are executed on the client and require the graphical user interface. This prohibits their use from other components in the control system.

In combination, these shortcomings would be hard to overcome with incremental improvements to the existing code. Therefore, we have decided to develop a new sequencer from scratch.

Taskomat: DESIGN

The new Taskomat sequencer models processes as sequences of individual steps like its predecessor. In contrast to it, it is designed around the following goals:

- **Integration** into the control system: Sequences can be started, controlled, and manipulated from the control system.

* lars.froehlich@desy.de

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR012

AUTOPARAM, A GENERIC ASYN PORT DRIVER WITH DYNAMIC PARAMETERS

Jure Varlec*, Cosylab d.d., Ljubljana, Slovenia

Abstract

Implementing EPICS device support for a specific device can be tricky; implementing generic device support that can integrate different kinds of devices sharing a common interface is trickier still. Yet such a driver can save a lot of time down the road. A well-known example is the Modbus EPICS module: the same support module can be used to integrate any device that speaks the Modbus protocol. It is up to the EPICS database to map device registers to EPICS records. Because no changes to the driver code are needed to integrate a device, a lot of effort is saved. At Cosylab, we often encounter device controllers that speak bespoke protocols. To facilitate development of generic drivers, we wrote the Autoparam EPICS module. It is a base class derived from `asynPortDriver` that handles low-level details that are common to all generic drivers: it creates handles for device data based on information provided in EPICS records and provides facilities for handling hardware interrupts. Moreover, it strives to provide a more ergonomic API for handling device functions than vanilla `asynPortDriver`.

INTRODUCTION

When it comes to putting together a control system, one of the advantages of EPICS is that, given existing device support, integrating devices requires little or no programming: data that devices provide or consume is mapped to process variables *declaratively* through *records* in an EPICS database [1]. Because EPICS is not merely software, but a vibrant community, lots of existing device support modules are readily available [2]. Chances are, then, that integrating a widely-used type of device into your control system requires very little effort.

Of course, not all devices are widely used. Some are brand new, some are niche, some may even be developed specifically for a particular machine. In such a case, developing a new device support layer cannot be avoided. This is an arduous task: device support represents a mapping from device functionality to records. This means that a device support layer is required for *each record type* of interest [3]. Much of this work is repetitive and results in pretty much the same record-specific code across many different device types. For this reason, the device support layer of EPICS lends itself well to encapsulating the repetitive common code in a reusable module.

`asynDriver` [4] is such a module. It has become the go-to module for integrating new devices, and even whole data processing pipelines [5]. Instead of implementing EPICS per-record device support directly, one instead implements one or more `asyn` *interfaces* to wrap the low-level device driver or communication protocol. These interfaces are then used by `asyn`'s *generic* per-record device support layer which covers pretty much all record types that are meant to get data into or out of hardware devices. By using `asyn`, one is thus saved from a fair amount of repetitive and error-prone work: write one driver, support all records automatically. The most straightforward way to create a device support module based on `asyn` is to create a C++ class derived from the `asynPortDriver` base class.

One can go a step further and recognize that some devices are themselves "generic" in the sense that they are merely front-ends to other devices. One example are programmable logic controllers (PLC) which often serve as interfaces to sensors and actuators. It is thus natural to desire an EPICS module that would communicate with a particular type of PLC in a generic enough manner to facilitate integrating it into the control system regardless of which peripherals are connected to it. There are, in fact, several such EPICS modules available, the Modbus module [6] being a very nice example. It allows integration of any PLC (or other device) that speaks the Modbus protocol. With a device support as generic as this, the "no programming" ideal mentioned in the beginning is truly possible: the EPICS database defines the mapping between Modbus registers and records, interpreting and giving meaning to the raw data values contained in the registers. Regardless of which peripherals are connected to the PLC, no changes to the drivers are needed.

But the Modbus protocol is just one possibility, chosen as an example because it is so widely known and used. As with all devices and protocols, there are many generic ones that are niche. Yet even for niche protocols, it makes sense to implement a module as generic as the Modbus module. A protocol may only be used at a single facility, yet that may still mean that it is used in many different setups (e.g. with different peripherals).

At Cosylab, we have helped with EPICS device support for such generic devices many times now. We felt the pain of doing the same kind of work several times. We have thus decided to develop a C++ base class which encapsulates the common functionality that needs to be implemented by any generic device support code; most importantly, this includes generating per-record handles dynamically based on information provided in EPICS records. This C++ base class builds on top of `asynPortDriver`, allowing us to reuse the basic functionality provided by this `asyn` class, and, crucially, to reuse the `asyn` per-record device support layer. We found this module, called `autoparamDriver` [7], to be generally useful, and are releasing it to the community.

* jure.varlec@cosylab.com

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR013

PROGRESSION TOWARDS ADAPTABILITY IN THE PLC LIBRARY AT THE EuXFEL

T. Freyermuth*, S. T. Huynh†, B. Baranasic, M. Bueno, N. Coppola, G. Giovanetti, S. Hauf, N. Jardón Bueno, N. Mashayekh, A. Silenzi, M. Stupar, M. Teichmann, J. Tolkiehn, L. Feltrin Zanellatto, P. Gessler, EuXFEL, Schenefeld, Germany

Abstract

In 2011, the European X-Ray Free Electron Laser (EuXFEL) commenced parallel developments of their control system (Karabo) and the Programmable Logic Controller (PLC) library. The PLC library was designed to control basic beamline components and under as set of initial assumptions, the automation component was deferred to the control system layer. After five years of operation, it can be seen that not all initial assumptions scaled well to the operational needs of the facility resulting in limitations hindering progress. Having identified the issues, the PLC development is now focused on providing a more cohesive and adaptable solution. In utilizing the IEC61131-3 (3rd edition) features, the PLC library has been restructured towards a layered architecture with loose coupling between function blocks. The ultimate goal is to achieve a PLC library which is not only test driven and capable of quickly integrating in new devices, but can achieve dynamic linking not only between hardware and software, but also across software devices, aiding the rapid development of more complex hardware integration and higher-level automation.

INTRODUCTION

Programmable Logic Controllers (PLCs) have been historically used to provide control and automation of larger systems, and this is no different at the European XFEL (EuXFEL)[1]. To further optimise the generation of PLC projects, a library has been developed to provide buildings blocks, which are pieced together to form the backbone of a PLC project. While initially developed to reduce development time for PLC projects, the PLC library also incorporates a bespoke communication protocol and provides functionality which over time became less efficient and harder to expand upon. As such, a new version of the library is underway, addressing some of the short-comings of the original:

- A strictly layered architecture that is testable
- Dynamic linking is provided by the TwinCAT Hardware Abstraction Layer (TcHAL)
- An enhanced communication protocol
- Simpler inter-device and inter-plc communication
- Additional tools

Paving the way for dynamic, adjustable and configurable complex automation processes, which can easily be developed and tested, whilst leaving room for future changes in a completely disentangled manner.

* Tobias.Freyermuth@xfel.eu
† Sylvia.Huynh@xfel.eu

CURRENT LIBRARY

The current PLC library is built up of a collection of software representations of hardware devices, which are known as softdevices. During early development, it was a priority to ensure that the rapid deployment milestones of PLC projects were met. The softdevice building blocks aided quick delivery through device instantiation, ensuring conformity amongst hardware devices performing a similar function set, and creating a simplified way with how these devices interact with the Supervisory Control and Data Acquisition (SCADA) system known as Karabo [2]. Using a TCP/IP communication protocol developed in-house, a connection header entailing basic device class information known as a self-description provided the SCADA system with a means to obtain information to aid User Interaction (UI). As the number of PLC projects grew in size, it became relevant to ensure a means for peer-to-peer communication between softdevices, in addition to basic equipment protection. This was achieved through shared variables and interlocks - a set of conditions which when met, would trigger a set of actions. Lastly, an additional tool called the PLC Management System (PLCMS)[3] was developed to aid the automatic generation of PLC projects with linking between the hardware Input and Output (I/O) and the previously defined softdevices. The structure of the current library can be seen in Fig. 1.

FUTURE LIBRARY

Building on from the lessons learned thus far in regards to the existing PLC library, this section will explore the changes planned for the next version of the EuXFEL PLC library, and how they work and will be integrated.

PLC Library Goals

Whilst the current library is functional, it is hoped that the many limitations that are currently imposed due to historical design decisions can be overcome in the next version. As many of the previous requirements remain valid, this section details on how they can be enhanced or evolved. Additionally, new requirements, which will facilitate to provide better building blocks for future PLC projects are introduced.

A Complete Self-description of the PLC A highly important feature of the current library is the self descriptive nature of the PLC. When a connection is made to the PLC (via a TCP client), the PLC will send a description of all the devices currently on it. While incredibly useful, the self-description is still lacking some features for true com-

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR021

EPICS IOC AND PVs INFORMATION MANAGEMENT SYSTEM FOR SHINE

Huihui Lv, Yingbing Yan†, Qingru Mi, Guanghua Chen
Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai, P.R. China

Abstract

For Shanghai HIgh repetition rate XFEL aNd Extreme light facility (SHINE), EPICS is adopted as standard to build the complex control system which involves hundreds of IOCs and millions of records. Most of IOCs run in industrial control computers as soft IOC. However, with the large amount of PVs, the need has emerged to develop remote tools to monitor all the channels and IOCs. One application backed by MySQL is designed, running periodically to interact with EPICS system where to take data from run-time databases via Channel Access and pvAccess. We embed scripting codes into the IOC startup script to realize that as soon as the IOC starts up, the information of the server's address, the IOC installation path, and all the records maintained by the IOC will be automatically pushed to the application and then stored in the database. With this necessary information, we could access all the active IOCs and PVs. Another web application is designed to render servers, IOCs and PVs data in MySQL on the web to give us an overall running status of the control system. We also fully consider the modularity and portability for the applications to apply and extend in none-SHINE environments.

INTRODUCTION

Motivated by the successful operation of X-ray FEL facilities worldwide and the great breakthroughs in atomic, molecular, and optical physics, condensed matter physics, matter in extreme conditions, chemistry and biology, the first hard X-ray FEL light source in China, the so called Shanghai HIgh repetition rate XFEL aNd Extreme light facility (SHINE), is under construction. SHINE will utilize a photocathode electron gun combined with the superconducting Linac to produce 8 GeV FEL quality electron beams up to 1 MHz repetition rate [1]. Hundreds of IOCs, located in Shaft #1, Shaft #2 and Shaft #3, will directly or indirectly control almost every equipment distributed in tunnels of the injector, the superconducting linac, and three FEL undulator lines, with the total length of approximately 3.1 kilometers long(as shown in Figure 1). Shaft #1 is a building at the end of the injector, Shaft #2 is at the end of linac, and Shaft #3 is at the end of undulator lines.

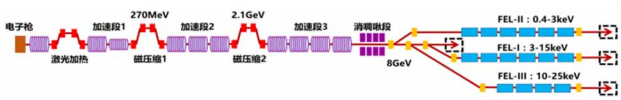

Figure 1: SHINE Accelerator Layout.

The control system involves heterogeneous equipment and interfaces with a number of different subsystems. Several IOCs will be deployed in one server, in which hundreds of PVs perform real-world I/O and local control tasks. But we suffer from a lack of high level tools to centrally manage all the IOCs and PVs and help us to verify which PVs are running on a specified IOC, and how many IOCs are running on a specific server. For an IOC, we also need to know which PVs are active. In this case, we develop the application to obtain all the PVs from EPICS system and store them in the MySQL database. In order to search through data easily, we also develop the web application to represent the information of the database. Details are described in the following sections.

ARCHITECTURE OVERVIEW

The data flow view of the architecture as shown in Figure 2 can be divided into four functional modules, namely database, import service, PV service and web service. The database is a general storage container for the properties and real-time values of servers, IOCs, EPICS records and PVs. The import service aims to import the initial information such as record types, field name lists and field types into the relational database from a spreadsheet. PV service fetches PVs from EPICS run-time databases via Channel Access and pvAccess. At the same time, it retrieves the running information of servers such as the available free memory, CPU load, etc. Then it saves all the data into the database. Web service is mainly used to render the data on the web to give us an overall running status of EPICS control system for SHINE.

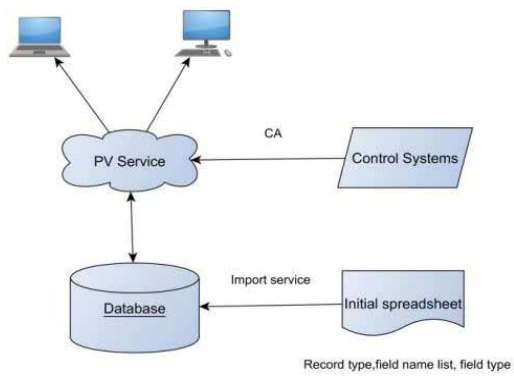

Figure 2: System Architecture.

Database Schema

The database saves the properties as Name/Value pairs. Details are illustrated in Figure 3. More specifically, it is needed to store the information of server, IOC, record and PV. For the server, the database is not only to store static attributes, like environment variables related to EPICS, but also to store the runtime information, for instance, when the server is started, when the server is scanned, CPU load,

yanyingbing@zjlab.org.cn

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR022

DATA ACQUISITION SOFTWARE PIPELINE FOR THE COMMISSIONING OF THE LoKI SMALL ANGLE NEUTRON SCATTERING INSTRUMENT

J. Walker*, S. Alcock, M.J. Christensen, J.E. Houston, K. Muric, W. Potrzebowski, T.S. Richter
European Spallation Source ERIC, Lund, Sweden
Davide Raspino, UKRI-STFC, ISIS Neutron and Muon Source,
Harwell Science and Innovation Campus, Chilton, Oxfordshire, UK

Abstract

The LoKI Small-Angle Neutron Scattering (SANS) instrument will be one of the first instruments to be commissioned at the European Spallation Source (ESS), and will contribute to the early science programme. The detector for the instrument was tested at the ISIS neutron source facility in March of 2022, and this paper outlines the data acquisition software pipeline. It consists of a readout master, an Event Formation Unit (EFU), an instance of Kafka, a NeXus file writer, and, the data reduction software, Scipp. The readout master is responsible for synchronising detector readout timestamps with an external reference, and aggregating those readouts into UDP packets sent to the EFU. The EFU processes the readout data to produce event messages containing a location and time of arrival for each neutron event detected, and acts as a Kafka producer sending event messages to Kafka. The NeXus file writer consists of a Kafka consumer that compiles event messages and other experiment data from a given time interval into a single NeXus file for further analysis. Finally, Scipp is a data reduction python library used to visualise and analyse the experiment data after an experiment is completed.

INTRODUCTION

The European Spallation Source (ESS), co-hosted by Sweden and Denmark, is currently under construction in Lund (Sweden), with the first neutrons expected in 2024. A number of detectors are being developed for the facility, one of which is for the LoKI Small-Angle Neutron Scattering instrument [1, 2], which is being developed in colaboration with Science and Technology Facilities Council (STFC) in the U.K. LoKI will make use of ^{10}Boron-Coated Straws (BCS) from Proportional Technologies Inc. (USA). In order to obtain acceptable efficiencies, these detectors consist of 7 boron-coated copper straws packed within 1 inch aluminium tubes, with each copper straw wired as a position sensitive detector. On LoKI, these 1-shell-escape inch aluminium detector tubes will then be packed into arrays to make detector panels, which will be placed in 4-panel banks around the beam at ~1.3 m and ~4 m from the sample, and a single panel bank on a carriage, which will move between ~5 m and 10 m from the sample positions. Given the number of detector straws in this design, in order to minimise detector signal cables, the multiplexing method will be used. The multiplexing method consists of a resistive chain between the 7 straws at each end of the aluminium tube, resulting in four signals from each each tube rather than 14. When a neutron is detected, 4 amplitude values are then produced, from which the location of the neutron event can be calculated. A smaller subset of this detector consisting of 128 tubes was used for the tests outlined in this paper. A diagram of this detector is shown in Fig. 1, showing the layout of the tubes, and the layout of the straws within each of those tubes is shown in Fig. 2.

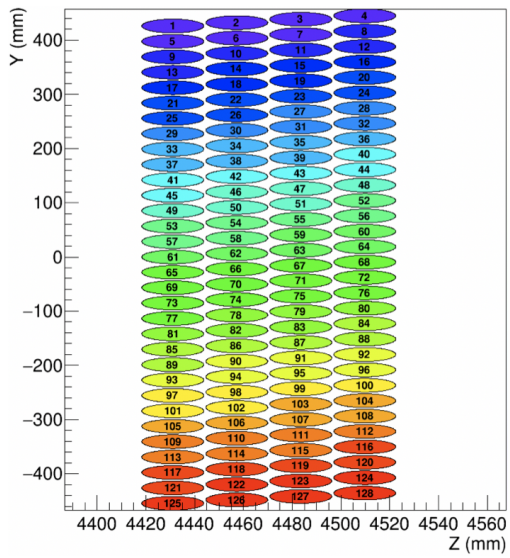

Figure 1: A diagram depicting the tubes of the subset of the LoKI detector tested at ISIS in March 2022.

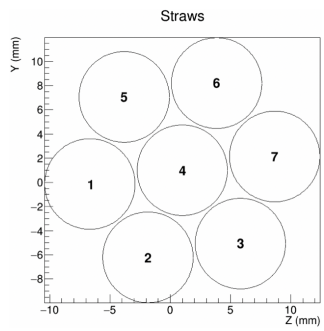

Figure 2: A diagram depicting the straws within each tube of the LoKI detector.

The detector was tested on the LARMOR instrument at the ISIS facility with the full data acquisition pipeline [3]. This pipeline will be used in production at the ESS facility. It consists of the following key components, developed by

* jennifer.walker@ess.eu

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR023

EXPERIMENTAL DATA COLLECTION STANDARDS AT SESAME SYNCHROTRON

M. Alzubi*, A. Abbadi, A. Al-Dalleh, A. Aljadaa, A. Lausi, A. Mohammad, B. Aljamal, G. Iori, G. Kamel, M. Abdellatief, M. Genisel, M. Harfouche, R. khrais, S. Matalgah, Y. Momani
SESAME Synchrotron, Allan, Jordan

Abstract

Experimental data collection is the essential process of acquiring experimental raw data along with its associated metadata from SESAME beamlines. For data collection and processing; scanning modes, data and metadata formats, and data visualisation are only a few aspects in which individual beamlines differ from each other. In addition, the volume of experimental datasets every experimental day might range from a few gigabytes to many terabytes. Herein, the effectiveness of the experiments being conducted at SESAME depends heavily on the efficiency and reliability with which experimental data are collected. Each beamline at SESAME has its own Data Acquisition (DAQ) system that is intended and being developed primarily to meet beamline users' and scientists' expectations. It also ensures that experimental raw data and metadata are not randomly generated and are stored together in a stander and well-defined file formats in compliance with SESAME Experimental Data Management Policy. In this paper, we present the standards and features employed in SESAME's DAQ systems, as well as the experimental data creation, curation, storage, and accessibility pipeline currently being built for SESAME beamlines.

INTRODUCTION

SESAME [1] is a third-generation synchrotron light that has currently three beamlines in operation and serves the SESAME users' community; the XAFS/XRF (X-ray Absorption Fine Structure/X-ray Fluorescence) spectroscopy [2], MS (Materials Science) and IR (Infrared Spectromicroscopy) beamlines [3]. Two more beamlines are being installed and will be commissioned soon, namely, HESEB (HElmholtz-SEsame Beamline) [4] and BEATS (BEAmline for Tomography at SESAME) [5, 6].

At SESAME, the first monochromatic beam was obtained in November of 2017 at the XAFS/XRF beamline. In August of 2018, the first user experiment was conducted on the same beamline. In June of 2020, the SESAME Council adopted its "Experimental Data Management Policy," which is a deliverable of the H2020 BEATS project that is a part of the EU research and innovation funding programme. This policy is harmonised based on the European Synchrotron Radiation Facility (ESRF) and PaNData data policy frameworks [7]. Late in the same year, the organisational structure of SESAME was reformulated and a new dedicated team was created, under the supervision of the scientific director, to obtain the experimental data from the beamlines in compliance with data policy. In January of 2021, the Data Collection and Analysis (DCA) team was founded with these objectives: i) enabling the beamlines' DAQ systems to generate experimental data aligned with SESAME's Data Management Policy, ii) increasing the productivity of experiments, iii) enhancing the scanning time and quality, iv) considering the maximum integration levels with control, motion, and scientific computing systems, v) minimising user experience gaps, vi) automating raw data and metadata collection, and vii) adhering to software engineering standards developing systems inasmuch as possible.

DATA POLICY AND DAQ STANDARDIZATION

The implementation of the data policy not only aids standardising DAQ systems and the data sets generated across beamlines, but also enhances the integration between DAQ and other systems in which they serve as a source of raw data and metadata, as well as the computing infrastructure used to store data and provide access and analysis services on it.

In nutshell, the data policy constitutes and describes the ownership, storage, access and management of the experimental data generated from SESAME beamlines [8]. It is applied on the experimental data for both users awarded beamtime proposals and in-house experiments. There is a three year embargo period after an experiment is completed, during which only the principal investigator of the beamtime proposal and the experimental team have access to the data. Passing the three years, the experimental data becomes openly accessible to SESAME registered users. There is a possibility to exceptionally extend this period by submitting a request to SESAME officials. The experimental data is kept at SESAME archiving for at most ten years on best efforts basis; the exact number of years will be defined for each beamline in reference to the data type and volume considering the financial and technical limitations. The policy implies sorting and archiving the experimental data in well-defined format where such data should be acquired from SESAME software and associated with unique key identifier. In contrast, this data policy does not apply to industrial research and collaborative research with SESAME scientists where other policies will be specifically developed.

EXPERIMENTAL DATA COLLECTION

The majority of SESAME's beamlines use locally developed and maintained DAQ systems. They are the main

* mostafa.zoubi@sesame.org.jo