



October 25th – 28th, 2016
Campinas, Brazil

11th International Workshop on Personal Computers and Particle Accelerator Controls

pages.cnpem.br/pcapac2016/



11th International Workshop on Personal Computers and Particle Accelerator Controls

Dear colleagues,

Welcome to Brazil and to CNPEM!

Thank you all for coming. This year we complete twenty years of the PCaPAC workshop series, and also twenty years of the first electron beam around the accelerator at LNLS (Brazilian Synchrotron Light Laboratory). These days in 2016 are a special time to celebrate!

For the first time, Brazil is the host of a workshop in the accelerators control field. Nevertheless, this country is well acquainted to the matter. The LNLS accelerator complex has produced its first electron beam in 1996. In order to reach this, ten years of preparation under severe conditions were necessary: a decade-long struggle for funding during an era of high inflation rates and of policies that aimed to limit the importation of IT goods. Brazilian young engineers and physicists faced the challenge and reached success. They created a whole system of data acquisition devices, which remain the basis of the LNLS accelerator control system, and which was made available to the local industry, at some extension.

We are here to exchange achievements and experience. But, in addition, we would like to proudly present to you the project and the construction works of SIRIUS, a fourth-generation accelerator, which involves enormous engineering challenges. The LNLS team, with many new-generation young engineers who joined us, will enjoy the opportunity to show and validate some of their ideas around the project.

I would like to thank you all for the contribution to the program, of course, through a talk, a poster or a tutorial session. I leave also my acknowledgment to the invaluable advices from the colleagues in the international Programme Committee. Last but not least, I leave my gratitude to the members of the Local Organisation Committee, from different areas of CNPEM. These months of intense planning and organizing for this PCaPAC workshop brought us even closer.

May we all have a great time!

James Rezende Piton
james@lnls.br

PCaPAC 2016 Programme Committee, chair
Beamline Software Group, LNLS/CNPEM

LNLS - Brazilian Synchrotron Light Laboratory
CNPEM - Brazilian Center for Research on Energy and Materials

ORGANIZERS | SCIENTIFIC COMMITTEE

1. Y.-M. Abiven (SOLEIL)
2. A. Arkilic (BNL)
3. M. Abbott (DIAMOND)
4. R. Bacher (DESY)
5. L. Catani (INFN)
6. J.-M. Chaize (ESRF)
7. M. Clausen (DESY)
8. M. Clift (AS)
9. P. Duval (DESY)
10. R. Farnsworth (APS)
11. P. Jemian (ANL)
12. N. Kamikubota (KEK)
13. T. Kosuge (KEK)
14. R. Lange (ITER)
15. O. Matilla (ALBA)
16. W. Mexner (KIT)
17. T. Mooney (APS)
18. J. Piton (LNLS), chair
19. M. Plesko (Cosylab)
20. T. Straumann (SLAC)
21. D. Tavres (LNLS)
22. J. Weber (LBL)
23. D. Zimoch (PSI)

LOCAL COMMITTEE

1. Daniel de Oliveira Tavares (LNLS/CNPEM)
2. Fábio Reis Fonseca (CNPEM)
3. Gustavo Barbosa Monteiro Bruno (LNLS/CNPEM)
4. Harry Westfahl Jr. (LNLS/CNPEM)
5. Ildéria Maíra dos Santos (CNPEM)
6. James Rezende Piton (LNLS/CNPEM)
7. Lucas Maziero Russo (LNLS/CNPEM)
8. Lucas Sanfelici (LNLS/CNPEM)
9. Luciana Noronha (CNPEM)
10. Pâmela Machado (CNPEM)
11. Renan Picoreti (LNLS/CNPEM)

Contents

Preface	i
Foreword	iii
Committees	iv
Contents	v
Papers	1
WECSPLIO01 – The Sirius Motion Control Report	1
WECSPLCO02 – Control System Evolution and the Importance of Trial and Error	6
WECSPLCO03 – Software Tests and Simulations for Control Applications Based on Virtual Time	10
WECSPLCO04 – Development and Current Status of a Carborne Gamma-Ray Survey System, Kurama-II	13
WEUIPLIO01 – Augmented User Interaction	16
WEUIPLCO04 – A Cython Interface to EPICS Channel Access for High-level Python Applications	21
WEOPRPO03 – Recent Beamline Interlock System and STARS at the Photon Factory	25
WEOPRPO10 – Personal Safety System in Sesame	28
WEOPRPO11 – Recent Improvements to the RIKEN RI Beam Factory Control System	31
WEOPRPO12 – Integration of Standalone Control Systems into EPICS-Based System at RIKEN RIBF	35
WEOPRPO18 – Automated Availability Statistics	38
WEOPRPO20 – Multipurpose Vacuum Chamber - Automation, Interlock-System and Self-Operating Vacuum Routines	41
WEOPRPO21 – Development of a Virtual Accelerator for Sirius	45
WEOPRPO22 – High Level Applications for Sirius	47
WEOPRPO23 – Beamline Supervisory System Using a Low-Cost Single-Board Computer	50
WEOPRPO24 – VDE - Virtual Documentation Environment	53
WEOPRPO25 – Using Tkinter of Python to Create Graphical User Interface (GUI) for Scripts in LNLS	56
WEOPRPO26 – Developments of the 'Cerberus' Laser Interlock and Hazard Display System and Associated Design Tool	59
THKTPLK01 – Open Hardware and Collaboration	61
THHWPLIO01 – PandABox: A Multipurpose Platform Adapted for Multi-technique Scanning and Feedback	67
THHWPLCO02 – New Controls Platform for SLAC High-Performance Systems	72
THHWPLCO04 – Open Hardware Experience on LNLS' Beam Diagnostics	75
THDAPLCO01 – Embedded Control System for Programmable Multi-Purpose Instruments	80
THDAPLCO03 – Gateware and Software Frameworks for Sirius BPM Electronics	84
THDAPLCO06 – A Framework for Development and Test of xTCA Modules With FPGA Based Systems for Particle Detectors	88
THPOPRPO03 – UVX Control System: An Approach with Beaglebone Black	91
THPOPRPO04 – openMMC: An Open Source Modular Firmware for Board Management	94
THPOPRPO05 – Implementation of a Precision Logarithmic Ammeter	97
THPOPRPO09 – Operation Experience and Migration of I/O Controllers for J-PARC Main Ring	101
THPOPRPO10 – Timing and Synchronization at FRIB	105
THPOPRPO11 – Processing SPE Files from Princeton Instruments during Data Acquisition in LNLS	108
THPOPRPO13 – High Level Software for the Commissioning of the European XFEL	110
THPOPRPO14 – Automation of the Magnetic Field Measurements of the Air Coils by Means of the Moving Wire System	114
THPOPRPO16 – Diagnostics at JINR LHEP Photogun Bench	117
THPOPRPO22 – Orbitkorrektur, a Java Client for Transverse Orbit Correction in PETRA-III	120
THPOPRPO23 – Fast Orbit Feedback at DELTA	123
THPOPRPO27 – High-Level Application Development and Production Infrastructure at TRIUMF	126
FRFMPLCO01 – Status of the NSLS-II LLRF System	129
FRFMPLIO03 – Overview of Some Feedback & Control Systems at Synchrotron Soleil	132
FRFMPLCO05 – A Fast, Custom FPGA-Based Signal Processor and Its Applications to Intra-Train Beam Stabilisation	137
FRFMPLCO06 – Harmony: A Generic FPGA Based Solution for Flexible Feedback Systems	141
FRITPLCO01 – Continuous Integration and Continuous Delivery at FRIB	145
FRITPLCO04 – Experience Gained during the Commissioning of the Undulator Control System at the European XFEL	148

Appendices	151
List of Authors	151
Institutes List	155
Participants List	159

THE SIRIUS MOTION CONTROL REPORT

H. D. Almeida, F. P. Figueiredo, M. P. Donadio*, J. R. Piton
LNLS, Campinas, Brazil

Abstract

Sirius is the new 4th generation synchrotron light source being built in Campinas, Brazil. The motion control report was created to describe all the steps taken to choose the set of motors, motor drives, and controllers that the hardware (GAE) and software (SOL) support groups will recommend. The steps include researching motion control systems in other Synchrotron laboratories, talking to the Sirius beamline designers, defining requirements and testing. This presentation describes the report, showing the information gathering process and latest results.

INTRODUCTION

The method to define the motion control solution for Sirius will be based in 7 steps:

1. Description of current motion control systems from the UVX synchrotron at LNLS, Brazil, and also from other laboratories, so that it is possible to have a base reference of what can be done and what are the current established practices related to motion control.
2. Definition of the requirements for the Sirius motion control system. It focuses on deciding, together with the beamline operators and scientists, what will be necessary for the motion control system to provide (precise movements, coordinated movements, fast movements, etc). It will also include requirements suggested by the support groups, either because they are useful (debugging support, installation on standard racks, etc), or necessary (adherence to safety standards, availability of software drivers, etc). Also, it will be considered the current expertise in equipment used in UVX as a requirement.
3. Initial device selection, where the requirements are formalized and the list of possible devices is presented. The result of this step is a filtered and standardized description of everything that beamline operators need from the motion control system and also a list of devices that will be part of the selection process.
4. Definition of tests. A set of tests will be elaborated in a way to identify the devices that comply with the requirements or not. The result of this step is a set of programs, types of equipment and textual descriptions necessary to execute the tests.
5. Execution of tests. All devices that we can acquire from the pre-selected list will be tested as necessary. The result of this step is a list of devices that passed the tests.
6. Reevaluation of tests. This step is basically doing the fourth and fifth steps again, if necessary.

7. Final device selection. The devices will be selected according to tie break rules. The result of this step is the set of recommended motion control devices for Sirius, which complete the report.

CURRENT STATE OF MOTION CONTROL SYSTEMS

The UVX light source at LNLS uses motors in the beamlines to control the beam setup, for example, to place mirrors in position, and also to control the experiments, for example, by selecting the energy from the monochromator.

Controllers

The main controllers used are the Galil DMC-4183 and Parker OEM 750X, which also has an integrated driver. There are a few IMS (integrated controller and motor) models 14A4, 17C4-EQ and 23C7-EQ and a few All-motion, models: EZHR17EN and EZHR23NHC.

Drivers

Basically, there are 3 drivers in use: Galil DMC 4140, Parker OEM750X, and Phytion ZMX+.

Motors and Control System Setup

There are many examples of motors used for beamline setup. For example, there are motors for positioning the first mirrors on the beamline entry point. The mirrors position the beam before entering the monochromator. The monochromator base may also contain motors, which will influence both the entry and exit positions of the beam at the monochromator. After the monochromator, motorized slits will select relevant parts of the beam, for example, the most uniform section of the beam, or some part with certain polarization properties. Another motorized set of slits farther away from the monochromator blocks scattered light. The table that holds all the equipment after the monochromator may rotate starting from the monochromator, to guide the light exiting the monochromator at different angles.

All those motors are used to set up the beamline, for example, to focus the beam at a precise point or to select the energy on fixed energy beamlines. They are set up before the experiments start.

Other motors are used during the experiment. Motors may move inside the monochromator to select the energy while doing scans. Sample holders move to select and position different samples to be tested. The samples or the cameras may be rotated to measure diffraction at different angles. The undulator needs to move synchronized with the monochromator to select different energies.

The motor types used at UVX are stepper motors and piezo. The stepper motor brands are Arsape, Haydon Kerk,

* marcio.donadio@lnls.br

IMS, Kalatec, Lin, Newport, Parker, Phytron, Sanyo Denki, Slo Syn, Syncro, Thorlabs, and Vexta. The piezo motor brands are Newfocus, Newport, and Pi. Some Phytron motors are specially built to operate in the vacuum.

Motion Control Software

The LNLS control system is almost completely standardized on using the EPICS [1] middleware. EPICS exports remote resources representing devices as remote variables (called "process variables") and organized into records. Motors are exported as motor records [2], with each kind of controller having its own server (called "IOC", or Input/Output Controller). Movement commands or automation is achieved by using scripts sending commands to change and read process variables. For example, a scan script, typically written in python, will connect to the process variables representing slits, change the process variable to move the slit motors to the new position and then measuring beam intensity by reading a process variable representing a photo-diode. This allows doing beam alignment.

Beamline operators may either write their own scripts from zero, or they may use the Py4Syn [3–5] library, developed at LNLS, which contain common procedures, like doing scans, plotting, fitting and storing data files. A set of example front-end scripts, that uses Py4Syn, is also available for beamline operators that don't want to program anything.

MOTION CONTROL IN OTHER ORGANIZATIONS

Research articles [6–43] explaining or citing parts of the motion control choices in other synchrotron labs were collected. Other particle accelerators with participation in ICALEPCS 2015 were also researched [44–51]. Also, e-mails were sent to some labs to gather direct information.

VME Devices

The following organizations use or have used VME controllers:

- CLS, BESSY, LCLS, PETRA, SSRF and SLS: OMS MAXv controller.
- DELTA and SLS: OMS VME58 controller.
- RHIC: OMS VX2 controller.
- APS and BESSY: Delta Tau PMAC2-VME controller.
- Spring-8: Advanet Advme2005 controller.

The combination of the VME motion controller with other VME devices and VxWorks is used to provide an integrated deterministic control system.

In-house Developed Devices

Some organizations have opted to develop devices or parts of devices in-house, the most evident example being the IcePAP system, developed by ALBA and ESRF and expected to be adopted by MAX IV and Solaris. IcePAP is composed of a rack with power supply, the controller, and driver boards. Up to 16 racks can be linked together to control a total of 128

motors. The IcePAP development was motivated by reducing compromises perceived to be present with off-the-shelf controllers. The project goal was to optimize functionality, performance, ease of deployment, a level of standardization and cost.

Another custom device is the YAMS system, developed at Elettra. The system uses a Galil DMC-21x3 controller as the basis of the project. It includes a rack for controller and drivers. The driver boards are custom designed, using the IMS/Schneider Electric IM48xH/IM805H micro stepping drivers. Encoder daughter boards were also designed. The YAMS development was motivated by standardization and costs.

Other in-house custom devices are an RS-485 based motion control system developed at RRCAT using the Texas Instruments LMD18245 driver chip and at Spring-8, a DeviceNet PLC based controller, and the PC-based Blanc4 controller.

Host Communication Protocols

There are many protocols and buses available for communication with the host machines or the rest of the control system. Some of them are listed here:

- Newport ESP7000: GPIB, RS-232 and USB
- Delta Tau PMAC2-VME: VME
- OMS MAXnet: Ethernet+TCP/IP and RS-232
- Galil DMC-21x3: Ethernet+TCP/IP, Ethernet+UDP/IP and RS-232
- OMS MAXv: VME
- Newport XPS: Ethernet+TCP/IP
- Parker 6K: Ethernet+TCP/IP and Ethernet+UDP/IP
- Delta Tau Turbo PMAC2: VME
- Delta Tau GeoBrick LV IMS: Ethernet+TCP/IP, USB and RS-232
- Adlink PCI-8134: PCI
- OMS VME58: VME
- Hytec IP8601: VME+IndustryPack
- IcePAP: Ethernet
- National Instruments PCI-STEP-4CX: PCI
- Attocube ANC300: Ethernet+TCP/IP, USB and RS-232
- Aerotech Ensemble CP10: Ethernet, USB and RS-232
- OMS VX2: VME
- Newport MM4006: RS-232, GPIB and RS-485
- Galil DMC-4183: Ethernet+TCP/IP, Ethernet+UDP/IP, USB and RS-232
- Delta Tau Power PMAC: Ethernet
- Tsujicon PM16C: Ethernet+TCP/IP, RS-232 and GPIB
- Advanet Advme2005: VME
- Interface Corporation PCI-7414M: PCI
- Galil DMC-1000: ISA
- Delta Tau PMAC PCI: PCI
- Galil DMC-40x0: Ethernet and RS-232

High Precision Devices

While there have been well-described efforts to standardize stepper and servo motor controllers in other labs, no

related efforts seem to have been done for piezo motors. For example, in MOCRAF 2015, it has been described the NSLS-II usage of a Delta Tau and Newport controllers for stepper and servo motors, while the list for piezo motors is composed of SmarAct, Attocube, PI, PiezoJena, and nPoint.

High precision may be used when building instruments, like microscopes or interferometers and on beamlines they are typically used to move sample holders or focusing and correcting small-scale positioning errors. A more specific use case exists at BESSY, a Nanomotion HR8 piezo motor is used to rotate the monochromator grating and mirror. Outside beamlines, the European XFEL has described at ICALEPCS 2015 the piezo-based cavity fine tuner, which reduces deformation on the accelerator's walls.

Complex and Coordinated Motion

Motions that happen on an arbitrary nonlinear path over time will be considered complex motions in this paper. Coordinated motion is a motion that simultaneously moves more than one motor.

Examples in complex motions: to change the monochromator energy in fixed energy steps, nonlinear movement with variable speed is required by the monochromator motors between each step. If a slit can move independently its top, bottom, right and left blades, then the operation of increasing the gap in the slit is a linear coordinated movement.

A nonlinear movement may also happen at a lower level, like in piezo devices, which may be nonlinear in nature. With the help of position encoders and closed loop feedback they might be translated and interpreted at higher levels as linear movements, if necessary.

An example of complex motion at SOLEIL comes from the ICALEPCS 2015 report on the REVOLUTION project. A beamline had its motion controller switched to a Delta Tau Power Brick to control the beamline energy selection. Seven equations were implemented to translate the required energy to the positions of each of the seven motors inside the monochromator so that only the high-level request for a specific energy is required. The goals of the implementation were to reduce communication with the host machine and to allow continuous energy scan operations.

At BESSY, a Nanomotion HR8 piezo motor uses a Delta Tau PMAC2-VME controller with a custom nonlinear model and closed loop feedback to overcome the encoder precision limitations when moving the piezo motor in very small steps. The motion also needs to be synchronized with the undulator for correct energy selection.

Another example of complex and coordinated motion is the SLAC LCLS fast wire scanner, used for beamline emittance diagnostics, which uses the S-curve available in the OMS/ProDex MAXv controller.

REQUIREMENTS FOR THE SIRIUS MOTION CONTROL SYSTEM

UVX as the Baseline

The beamline requirements for Sirius might not be well defined. For example, the beamline operators and scientists may not know what they need. The current motion control environment in UVX may be used as a baseline for these cases. Knowledge and expertise on the current solution are also considered requirements with strong importance. This is because it took years of work from LNLS workers to understand, correct and develop hardware and software until the system stabilized. Probably this work needs to be redone when choosing different components.

Standardization of Components

We expect to follow an 80% / 20% rule, considering a cheap and simple solution for 80% of the motors and an expensive, precise and resource-full solution for the remaining 20%, regarding complex and coordinated motion. In particular, we expect that a single selection of components will be useful for the majority of use cases. For the more specific cases either one or more sets of components will be required, the less, the better for administration and maintenance costs.

Requirements for the New Beamlines

Currently, out of the 13 initial beamlines planned for Sirius, 5 of them have an approved preliminary design report. We have talked to the beamline scientists about the motion control system of these 5 lines and also with 1 other scientist for the remaining beamlines. The discussed requirements are detailed below.

Sapucaia The Sapucaia beamline will be an SAXS beamline, with support for doing "nanobeam" experiments. The main beamline elements are:

- the initial slits after the protection wall
- a double crystal monochromator
- a positioning mirror
- two sets of slits for slicing and focusing the beam
- kinoform lenses to focus the nanobeam
- the sample holder stage
- the X-ray detectors, that detect light scattered from the samples.

Cateretê The Cateretê beamline is defined as a coherent and time-resolved X-ray scattering beamline. It will support time-resolved small angle X-ray scattering (SAXS) and ultra-small-angle X-ray scattering (USAXS), which are techniques for structural studies of nanoparticles, coherent diffractive imaging (CDI), a technique for 3D imaging with 30 nm resolution that uses coherent X-ray scattering instead of lenses and X-ray photon correlation spectroscopy (XPCS), for studying dynamics of nanoparticles using fluctuating patterns generated by coherent X-ray scattering.

The main beamline elements are:

- the undulator at the storage ring

- the initial slits
- a double crystal monochromator
- a focusing mirror
- a set of slits for slicing the beam
- the sample stage
- the detector stage

Carnaúba The Carnaúba beamline is defined as a nano focus beamline, providing a beam focused to a diameter of up to 80 nm. It targets fluorescent and absorption techniques, as well as imaging based on X-ray scattering.

The main beamline elements are:

- the undulator at the storage ring
- the initial slits
- a focusing mirror
- a second mirror for positioning and removing harmonics
- a set of slits used as the secondary light source
- a 4-bounce crystal monochromator
- a set of slits after the monochromator to slice the beam
- a Kirkpatrick-Baez mirror for focusing the beam at nanometric scale
- the sample holder station

Ipê The Ipê beamline targets soft X-ray spectroscopy, with an energy range of 100 eV to 2.000 eV and will have two separate experimental end stations for two different techniques: resonant inelastic X-ray scattering (RIXS), which uses emitted photons, and near ambient pressure X-ray photoelectron spectroscopy (NAP-XPS), which uses emitted electrons to perform the spectroscopic measurement. The main beamline elements are:

- the undulator at the storage ring
- a first mirror used as a filter to reduce thermal load
- a grating and mirror (planar grating) monochromator
- a deflecting mirror used to switch between the two end stations
- a set of slits after the monochromator to slice the beam before each end station
- focusing mirrors for each end station
- sample stages on each end station
- detector stages on each end station

RESULTS AND DISCUSSION

Step 2 of the motor selection procedure was concluded and the requirements were partially described in this paper. Next step, the initial device selection will start.

CONCLUSION

An extensive work needs to be done to finish this procedure. When we have the final results a new report will be written.

REFERENCES

- [1] EPICS, <http://www.aps.anl.gov/epics>
- [2] EPICS: Motor Record and Device/Driver support, <http://www.aps.anl.gov/bcda/synApps/motor>
- [3] Py4Syn GitHub, <https://github.com/hhslepicka/py4syn>
- [4] Py4Syn Documentation, <http://py4syn.lnls.br>
- [5] H. H. Slepicka *et al.*, "Py4Syn: Python for Synchrotrons", *J. Synchrotron Rad.*, vol. 22, pp. 1182-1189, 2015.
- [6] D. Hernández-Cruz, A. P. Hitchcock, T. Tyliczszak, M.-E. Rousseau, and M. Pézolet, "In situ azimuthal rotation device for linear dichroism measurements in scanning transmission x-ray microscopy", *Rev. Sci. Instrum.*, vol. 78, p. 033703, 2007.
- [7] E. Gann *et al.*, "Soft x-ray scattering facility at the Advanced Light Source with real-time data processing and analysis", *Rev. Sci. Instrum.*, vol. 83, p. 045110, 2012.
- [8] D. J. Vine *et al.*, "An in-vacuum x-ray diffraction microscope for use in the 0.7-2.9 keV range", *Rev. Sci. Instrum.*, vol. 83, p. 033703, 2012.
- [9] S. Stepanov *et al.*, "JBLuCe-EPICS control system for macromolecular crystallography", *Acta Crystallogr. D Biol. Crystallogr.*, vol. 67, pp. 176-188, 2011.
- [10] D. Zangrando *et al.*, in *Proc. EPAC'08*, pp. 2329-2331.
- [11] N. Janvier, J. Clement, P. Fajardo, and G. Cuní, in *Proc. ICALEPCS'13*, pp. 766-769.
- [12] K. Cerff, D. Haas, D. Jakel, and M. Schmitt, in *Proc. PCaPAC'14*, pp. 198-200.
- [13] M. Clift, R. Farnsworth, A. Starritt, and L. Corvetti, "Motion control using EPICS and Galil controllers", presented at ICALEPCS'09, Kobe, Japan, Oct. 2009, paper WEP056, unpublished.
- [14] D. G. Hawthorn *et al.*, "An in-vacuum diffractometer for resonant elastic soft x-ray scattering", *Rev. Sci. Instrum.*, vol. 82, p. 073104, 2011.
- [15] R. Louis *et al.*, "Synchrotron powder x-ray diffractometer beamline at J. Bennett Johnston, Sr., Center for Advanced Microstructures and Devices", *Nucl. Instr. Meth. Phys. Res. Sect A*, vol. 582, pp. 84-86, 2007.
- [16] M. D. Miller, G. N. Phillips, Jr., M. A. White, R. O. Fox, and B. C. Craft, III, "The development of the GPCPC protein crystallography beamline at CAMD", *Application of Accelerators in Research and Industry - Sixteenth Int'l Conf.*, pp.734-740, 2001.
- [17] C. Conolly, "Facility upgrades, networking and computing", *CHESS News Magazine*, pp. 13-16, 2009.
- [18] B. Nutter, in *MOCRAF'13*, http://www.synchrotron-soleil.fr/images/File/Informatique/Workshop-Motion/Talks/TT01_Diamond-Nutter-technical-Mocraff-2013.pdf
- [19] U. Berges and S. Döring, in *Proc. ICALEPCS'07*, pp. 232-234.
- [20] M. Lonza *et al.*, in *Proc. ICALEPCS'11*, pp. 589-592.
- [21] A. Balzer *et al.*, in *Proc. ICALEPCS'05*, paper MO4B.2-20.
- [22] K. Horiba *et al.*, "A high-resolution synchrotron-radiation angle-resolved photoemission spectrometer with in situ oxide thin film growth capability", *Rev. Sci. Instrum.*, vol. 74, pp. 3406-3412, 2003.

- [23] Y. Nagatani and T. Kosuge, in *Proc. PCaPAC'14*, pp. 78-80.
- [24] N. Inami, Y. Takeichi, and K. Ono, "Real-time motion control and data acquisition system for scanning x-ray microscopy using programmable hardware", *J. Phys.: Conf. Ser.*, vol. 502, p. 012011, 2014.
- [25] T. Kracht, in *MOCRAF'11*, <http://www.synchrotron-soleil.fr/images/File/soleil/ToutesActualites/Workshops/2011/MotionControl/WS-MoCRaf-TL.04-MotionControlSoleilMay2011.pdf>
- [26] M. R. Jathar, in *Proc. ICECT'11*, vol. 2, pp. 104-106.
- [27] D. M. Gassner *et al.*, in *Proc. IPAC'11*, pp. 462-464.
- [28] K. Takemoto *et al.*, "Development of an auto-focusing imaging system in the soft x-ray microscope beamline of the SR center in Ritsumeikan University", *J. Phys.: Conf. Ser.*, vol. 186, p. 012019, 2009.
- [29] Z. H. Zhang, W. H. Jia, P. Liu, and L. F. Zheng, in *Proc. IPAC'13*, pp. 2998-3000.
- [30] X. Lan *et al.*, "SPEC application for achieving inelastic x-ray scattering experiment in the SSRF", arXiv:1508.06726, 2015.
- [31] P. Liu, in *EPICS Collaboration Meeting'11*, http://www.aps.anl.gov/epics/meetings/2011-06/sys/data/16/LIU_Ping_BL_Control.ppt
- [32] Q.-S. Wang *et al.*, "The macromolecular crystallography beamline of SSRF", *Nuclear Science and Techniques*, vol. 26, p. 010102, 2015.
- [33] W. Klysubun *et al.*, "X-ray absorption spectroscopy beamline at the Siam Photon Laboratory", *AIP Conf. Proc.*, vol. 879, pp. 860-863, 2007.
- [34] D. Corruble *et al.*, in *Proc. ICALEPCS'13*, pp. 81-84.
- [35] S. Z. Zhang *et al.*, in *Proc. ICALEPCS'15*, pp. 201-204.
- [36] M. Ishii and T. Ohata, in *Proc. ICALEPCS'09*, pp. 465-467.
- [37] T. M. McPhillips *et al.*, "Blu-ice and the distributed control system: software for data acquisition and instrument control at macromolecular crystallography beamlines", *J. Synchrotron Radiat.*, vol. 9, pp. 401-406, 2002.
- [38] C. L. Li, A. M. Kiss, and W. J. Zhang, in *Proc. IPAC'15*, pp. 1243-1245.
- [39] J. Krempaský *et al.*, in *Proc. ICALEPCS'13*, pp. 729-732.
- [40] V. N. Strocov *et al.*, "High-resolution soft X-ray beamline ADRESS at the Swiss Light Source for resonant inelastic X-ray scattering and angle-resolved photoelectron spectroscopies", *J. Synchrotron Radiat.*, vol. 17, pp. 631-643, 2010.
- [41] J. Raabe *et al.*, "PolLux: a new facility for soft x-ray spectro-microscopy at the Swiss Light Source", *Rev. Sci. Instrum.*, vol. 79, p. 113704, 2008.
- [42] I. Saleh and A. Ismail. ftp://ftp.sesame.org.jo/SESAME-Uploads/SAC-TAC/SAC-TAC-2013/SAC2013/SAC-2013-Motion_Control.pptx
- [43] K. Cole, R. R. Vallance, and T. Lucatorto, in *Proc. Precision Mechanical Design and Mechatronics for Sub-50nm Semiconductor Equipment'08*, pp. 117-122.
- [44] R. Walton *et al.*, in *Proc. ICALEPCS'15*, pp.1-4.
- [45] E. Suljoti, in *MOCRAF'15*, <http://www.synchrotron-soleil.fr/images/File/Informatique/Workshop-Motion/MOCRAF-2015/MOCRAF2015-AfternoonSession2-03-StatusAndNewDvpInMotionControlAtBESSY-II.pdf>
- [46] J. M. D'Ewart, M. Campell, P. Krejcik, H. Loos, and K. Luchini, in *Proc. ICALEPCS'15*, pp. 114-116.
- [47] M. Linberg *et al.*, in *Proc. ICALEPCS'15*, pp. 240-243.
- [48] P. Goryl *et al.*, in *Proc. ICALEPCS'15*, pp. 510-512.
- [49] R. A. Kadyrov, J. H. De Long, K. Ha, S. So, and E. Stavitski, in *Proc. ICALEPCS'15*, pp. 881-884.
- [50] W. Lewis, in *MOCRAF'15*, <http://www.synchrotron-soleil.fr/images/File/Informatique/Workshop-Motion/MOCRAF-2015/MOCRAF2015-MorningSession2-02-NSLS-II.pdf>
- [51] C.Y. Liao *et al.*, in *Proc. ICALEPCS'15*, pp. 1173-1176.

CONTROL SYSTEM EVOLUTION AND THE IMPORTANCE OF TRIAL AND ERROR

P. Duval, M. Lomperski, DESY, Hamburg, Germany
J. Bobnar, Cosylab, Ljubljana, Slovenia

Abstract

In this paper we address the importance and benefits of trial and error in control system evolution. Here we refer to the control systems of particle accelerators and large machines, whose control systems, although complex, will not lead to catastrophe in case of failure. We likewise focus on the evolution of control system software, although the issues under discussion will apply to and are often driven by control system hardware. We shall contrast classical Darwinian evolution via natural selection with control system evolution, which proceeds rather via artificial selection, although there are numerous software memes which tend to replicate according to their 'fitness'. The importance of general trial and error, i.e. making mistakes and learning from them, in advancing the capabilities of a control system will be explored, particularly as concerns decision making and overcoming *Einstellung*.

INTRODUCTION

A mature accelerator control system will be able to address a wide variety of problems which might arise throughout the controlled facility's natural lifecycle. Solving new problems or a push to provide better solutions to old problems will generally lead to control system evolution, even if this amounts to little more than keeping up with industrial or commercial components. The way one goes about problem solving will in turn have a marked influence on the pace of this evolution. We will discuss many of these aspects below, finishing with a few concrete examples of control system evolution at play.

GOALS AND PROBLEM SOLVING

The God Complex and Einstellung

When we are well-versed in our control system and at the same time faced with a new problem or challenge we are apt to fall prey to the God Complex, i.e. that "no matter how complicated the problem, you believe that your solution is correct." [1, 2] This is furthermore often compounded by what psychologists refer to as the *Einstellung* effect, or the "predisposition to solve a given problem in a specific manner even though better or more appropriate methods of solving the problem exist". [3,4]

The danger is not that our problem won't get solved. It most likely will. The danger is that we might not only miss an opportunity to explore new ideas, we might also end up wasting resources, and/or missing the big picture entirely due to our rush to implement a known solution.

Priming and Anchoring

Indeed our choices in problem solving and decision making are often due to an implicit memory effect known as priming [5], where exposure or familiarity with one stimulus (or solution paradigm) can influence our response to another. The classic trivial example: "How many animals did Moses take on the ark?" (answer: 0) might appear to have little to do with our decision making until we realize that our *expectations* can be easily primed, a case of priming known as *anchoring* [5]. For example, imposing an artificial deadline of one week to try some solution automatically suggests a level of difficulty. Worse, refusing to consider a new solution because "everyone else does it differently" suggests a knowledgeable rejection of the new solution. Unfounded expressions such as "one week" or "everyone else" often serve only to anchor our expectations at some level.

Accumulated Advantage

The previous example of anchoring ("everyone else does it differently") is also an example of the effect of accumulated advantage, often referred to as the *Matthew Effect*, (from Matthew 25:29 in the King James version of the Bible) [6]. In point of fact, our opinions are strongly related to and often dependent on those of others. The crowded restaurant *must* serve better food than the empty one next door! In an experiment by Duncan Watts [6], two sets of college students could download garage band music from two web sites. The sites were identical except that in one case the students could see the *likes* and downloads of everyone else. It's not surprising that in one case there were a handful of hit songs and in the other the *likes* and downloads showed a flat distribution.

Trial and Error

We should in any case be aware of the aforementioned challenges to our problem solving abilities. Whether we admit that we already know the solution to a new problem or not, the practice of trial-and-error cannot be avoided. The basic algorithm of trial-and-error can be described as:

- 1) Define what constitutes a solution to our problem.
- 2) Try *something*.
- 3) Check to see if the problem is solved. If not:
- 4) Modify *something* into a more promising direction and repeat step 3). Or, if the problem is solved:
- 5) Quit.

The psychological effects we have just discussed will of course influence the *something* that we initially try in step 2). In fact, if there is any kind of time pressure the best bet is indeed to go with our best over-all hunch. If on the other hand there is a time-window for bold experimentation, trying several *somethings* might lead to remarkable improvements.

EVOLUTION

Evolution, Darwinian or otherwise, naturally progresses via trial-and-error. The incremental evolutionary steps might occur by chance, as in the case of the natural world, or by design as in the case of control system evolution. Either way, the measure of success is the ability to replicate. Thus there is an implicit drive to improve.

Darwinian Evolution

In Darwinian evolution [7] the replicating unit is the gene. Any mutation which leads to a greater chance of survival will in turn lead to an organism's genetic material replicating itself more often, which is what we mean by improvement. Evolution by natural selection is of course slow and has a direction. That this occurs by chance means that evolutionary changes cannot be reengineered in order to improve performance. Any improvement comes entirely by trial-and-error.

An oft cited *proof of intelligent design* by creationists is the eye, which is so complex that it couldn't possibly have happened by chance and *must* have had a designer. A billion years, though, is ample time for cells initially able to only distinguish dark from light to successively evolve into such a remarkable organ. More to the point, the design is actually rather clumsy, as recognized already in the 19th century by Hermann von Helmholtz. No engineer would route the wiring leading from a camera's photo cells back into the path of the light source and then bundle it all into a thick cable near the center of the collecting surface, thereby creating a blind spot!

There is no chance to reengineer this into a more sensible solution. Nor is there any chance that a random mutation will fix the design flaws.

It is nonetheless instructive to recognize how incredibly well the eye does work (in tandem with our visual cortex). The design flaws are practically irrelevant, a point which should be remembered when the temptation to refactor complicated, yet well-working, software arises.

Software Evolution

Software evolution is driven by design decisions from the very beginning. Although survival might still be of the fittest, the agent of change is artificial- rather than natural-selection, and the replicating unit will be the *meme*, the smallest idea that gets transferred within a culture [8] (e.g. the idea of *sockets* or *threads*, but not necessarily the implementation of them).

Manny Lehman identified three categories of software, S- (*specific*) programs, P- (*procedural*) programs, and E- (*evolutionary*) programs [9]. S-programs are written once for a specific purpose. P-programs implement a set of

procedures only (e.g. play chess). E-programs perform some real-world activity and adapt to the environment and circumstances in which they run.

As much as we might wish particle accelerator control systems to be P-programs, they are in fact E-programs and necessarily evolve. In fact, as the environment in which a control system operates does indeed evolve, one of Lehman's Laws [9] asserts that the quality of the control system will decline unless it also evolves.

Here, however, we do have and often utilize the ability to refactor *bad* or *clumsy* design decisions. Of course, the question remains as to what a bad design is and (like the eye) as to whether it is in the end worth the risk of changing a (well-running and complex) running system merely for the sake of improving the design.

Complex software might also contain vestige routines (analogous to the appendix) which have no practical purpose but continue to be accessed by vintage application programs and are therefore required to exist. Thus, API breaks in reusable software such as control system libraries should be avoided when possible, including the disposal of deprecated API routines or class methods, unless the consequences of doing so are understood beforehand.

In addition to keeping pace with an evolving hardware environment, control system software will evolve on its own accord in order to improve or introduce functionality. The pace of evolution here will be strongly dependent on the developer's susceptibility to the psychological effects mentioned in the previous section.

Regardless of pace, any real evolutionary change will occur via trial-and-error. Typically, coding modifications will be run through various unit tests (a tight trial-and-error loop) until there is a new release candidate. The next trial might occur in the field when the software is deployed. After deployment, however, the cost of error will be much higher. In the case of accelerator control there is fortunately little or no chance of catastrophic error (as there is in airplanes or nuclear power). Nonetheless an error can lead to downtime or damage to equipment. Thus the cost of error should be examined along with rollback strategies prior to any new deployment.

EXAMPLES

Control System Protocol

One of the problems we sometimes have to deal with is an unacceptably high load (CPU and/or network) on a control system server. If this load is primarily due to information transfer from server to client then we have an issue with the control system protocol.

The TINE [10] control system makes use of the device server paradigm, where a server exposes control system elements as instances of *devices* and offers access to their attributes and actions through *properties*. It also offers *publish-subscribe* data acquisition, which in itself goes a long way in reducing unnecessary load on a server due to data transfer to multiple clients. However if client applications obtain data via repetitive non-persistent

transactions (*polling*) then there is no load reduction from *publish-subscribe*. The situation can be compounded many-fold if a server with many device instances (e.g. a vacuum pump, beam position monitor, or power supply controller - PSC) is requested to deliver information from all elements one-at-a-time. And precisely this is an all too common occurrence with simple client panel applications.

When faced with this situation in 2009 with multiple *ddd* [11] clients accessing the FLASH PSCs we decided to eschew the traditional split-the-load-among-multiple-servers approach and add a new feature to the control system protocol called contract coercion [12].

As the PSC server is not only prepared to, but prefers to send property information for all PSC instances for a given property as a multi-channel array, we addressed the question “Can we coerce a client’s synchronous request for the value of a property into a monitor for all values of that property?” We answered yes and then introduced contract coercion. The initial results were more than encouraging as the load on the server was effectively decimated without modifying a single line of client code.

To be sure, there was a significant amount of tight-loop trial-and-error with unit tests, etc. prior to deployment, but we were nevertheless aware of the costs of unforeseen errors beforehand. We assumed, based on prior testing, that the likelihood of a serious error on *most* clients was extremely small. If an exotic client did have an error we based the decision to rollback or not on how critical to operations the exotic client was and whether the error appeared immediately or some significant time later.

In the end, no rollbacks were ever necessary. Several errors were nonetheless encountered and repaired in the months following initial deployment. Likewise, the ensuing years saw several quality-of-service additions to the initial contract coercion implementation, each with its own trial-and-error process.

Control System Services

The TINE control system offers many central services, among them a plug-and-play system concerning name resolution.

The TINE Equipment Name Server (ENS) maintains a device server database and provides address information when a client needs to contact a control system element. The ENS database can of course be modified by an administrator, but in general it is updated automatically. The plug-and-play mechanism will add a new server to the database or update a server’s meta-information with every server start.

This level of automation requires a good deal of trial-and-error whenever new features are added. The ENS must not only guarantee a unique entry for a control system server it must also inform any server trying to usurp an existing name that its request was denied. The requested names and meta- information of any new server must also be validated, etc.

When we make modifications here, however we are modifying a central service rather than the control system protocol itself. What are the costs of error in this case?

The ENS would appear to offer a critical service, in contrast to, say, central archive or alarm servers. In fact it is *semi*-critical. All clients already have a fallback mechanism (using the last locally cached address in the event of address resolution failure) when the ENS is not operational. The worst that can happen is either 1) a new server will not be able to plug itself into the system, or 2) an existing server starting on a new host will not be able to modify its address information.

The other TINE central services are even less critical in that operations are never threatened in the event of error.

Applications

A good example of a specific application with on-going trial-and-error is the Operation History Viewer in TINE Studio [13, 14]. This application shows the machine state information (including problems) over any selected time range. The *problems* state indicates non-availability of the machine and can be divided into sub-systems, where one has the ability to browse through the fatal alarms responsible for the downtime. The goal is to have a fully automatic calculation of operation and availability history. As blame for non-availability is assigned to fatal alarms, we see that the application consists of more than a mere presentation of data, and involves, among other services, the central alarm system in a vital way. We may not break free from the trial-and-error loop here for some time to come and have added the ability to post-correct both the state information and the availability information (by a machine coordinator) over any time interval.

The costs of error here, as for most applications, apply almost exclusively to the application itself, and will have no impact on operations unless the application is critical to operations (which this isn’t). This is not to say that we can make errors with impunity. Once an application is regularly used, any degradation in quality of service will of course result in unhappy customers. Ensuring that the most-used features continue to work properly is generally sufficient to allow deployment of a new version. Should an error be discovered, a rollback can easily be made while the error is dealt with.

CONCLUSIONS

Control system evolution will occur if for no other reason than the necessity of keeping pace with the commercial and industrial world. Real innovation in control system software will involve a trial-and-error period. This period can be extensive or even continual, but primarily constitutes what is meant by control system evolution. Dramatic improvement most often occurs if we resist the psychological pressures to solve any new problems in a tried-and-true manner and admit that we perhaps don’t already know the best course of action. Often enough, deadlines will require us to play it safe, but if we have a large enough time window for development such that we can test several solutions to the same problem then we can often make great strides in the advancement of our control systems.

REFERENCES

- [1] A. Cochran,
https://en.wikipedia.org/wiki/Archie_Cochran
e
- [2] T. Harford, "Trial, Error and the God Complex";
<https://www.ted.com/>.
- [3] A. Luchins, "Mechanization in problem solving: The effect of Einstellung". Psychological Monographs, 1942.
- [4] M. Bilali and P. McLeod, "Why Your First Idea Can Blind You to a Better One", Scientific American, March 2014.
- [5] D. Kahneman, "Thinking Fast and Slow", Penguin Books, 2011.
- [6] S. Pinker, "The Better Angels of Our Nature", Viking Press, 2011.
- [7] R. Dawkins, "The Blind Watchmaker" (and references therein), W.W. Norton & Co., 1986.
- [8] R. Dawkins, "The Selfish Gene", Oxford Press, 1976.
- [9] M. Lehman,
https://en.wikipedia.org/wiki/Software_evolution
- [10] TINE, <http://tine.desy.de>
- [11] jddd, <http://jddd.desy.de>
- [12] P. Duval and S. Herb, "The TINE Control System Protocol: How to Achieve High Scalability and Performance", in *Proc. PCaPAC'10*, paper WECOAA02.
- [13] P. Duval, M. Lomperski, and J. Bobnar, "TINE Studio, Making Life Easy for Administrators, Operators and Developers", in *Proc. ICALEPCS'15*, paper WEPGF133.
- [14] P. Duval, M. Lomperski, H. Ehrlichmann, and J. Bobnar, "Automated Availability Statistics", presented at PCaPAC'16, Campinas Brazil, Oct. 2016, paper WEPOPRPO18, this conference.

SOFTWARE TESTS AND SIMULATIONS FOR CONTROL APPLICATIONS BASED ON VIRTUAL TIME

M. Hierholzer*, M. Killenberg, T. Kozak, N. Shehzad, G. Varghese,
M. Viti, DESY, Hamburg, Germany

Abstract

Ensuring software quality is important, especially for control system applications. Writing tests for such applications requires replacing the real hardware with a virtual implementation in software. Also the rest of the control system which interacts with the application must be replaced with a mock. In addition, time must be controlled precisely. We present the VirtualLab framework as part of the Chimera Tool Kit (formerly named MTCA4U). It has been designed to help implementing such tests by introducing the concept of virtual time, and combining it with an implementation basis for virtual devices and plant models. The virtual devices are transparently plugged into the application in place of real devices. Also tools are provided to simplify the simulated interaction with other parts of the control system. The framework is designed modularly so that virtual devices and model components can be reused to test different parts of the control system software. It interacts seamlessly with the other libraries of the Chimera Tool Kit such as DeviceAccess and the control system adapter.

INTRODUCTION

To test software automatically, a virtual test environment has to be provided. With the VirtualLab framework the necessary tools for this tasks are available. The framework is part of the Chimera Tool Kit and is available as open-source software [1].

This paper explains the procedures of writing tests for control applications at the example of a low-level RF controller server for FLASH-like machines. The low-level RF system used at FLASH [2] and XFEL [3] uses ADC and DAC boards based on MicroTCA.4 [4], connected to down converters and vector modulators for 1.3 GHz controls. The machine is pulsed with 10 Hz. The control loop for the fast phase and amplitude stabilisation is running on FPGAs on the ADC/DAC boards. Trigger pulses are sent to the FPGA and with delay to the low-level RF controller server. The server is based on the DOOCS middleware and running on the frontend CPUs. It presents the interface to the control system and performs several slow tasks, like generating tables for setpoint, feed-forward, gain etc. based on input parameters provided by the operator, and executes slow control loops for drift compensation and adaptive feed-forward. This controller server is a critical element required for the operation of the machine. Therefore thorough tests are crucial to prevent machine failures and unnecessary down time.

To avert regression failures of the software staying undetected and being included in the production system during

the next software update, automated continuous integration tests should be implemented. This requires full automation of the tests.

VIRTUAL DEVICES

Virtual devices can be used to achieve a full automation of tests. In contrast to testing on real hardware devices, tests based on virtual devices can fully govern the function of the device. Faults can easily be injected to test exception handling.

Figure 1 shows the layout of the test example. The test routines take control over the low-level RF controller server to be tested. The server is connected through a dummy register set with a state machine and a control loop algorithm reflecting the relevant behaviour of the FPGA firmware. This algorithm is connected through signal sinks and signal sources with a simple cavity model. These signal sinks and sources help creating the modularity needed for reusing parts of the virtual components for different tests.

A simple example for a test routine is shown in Figure 2. The test routine first sends the command to ramp up the gradient through the control system. Next it waits until the procedure is completed and finally it tests the result by comparing the actual current gradient of the cavity model with the nominal set point.

The virtual devices used for the test may be an imperfect approximation of the real devices. This presents no issue if the approximation is good enough for the application to function normally. In this particular case the actual control loop is not part of the test, which strongly relaxes the requirements on the cavity model. The control loop implementation and the cavity model can be tuned to each other to minimise the effort. Faults (like quenches of super-conducting cavities) don't need to be properly simulated, as long as there is no sensitivity in the tested software to those details. Simply switching off the measured signal might be enough to simulate such condition.

USE VIRTUAL TIME TO AVOID RACE CONDITIONS

The example test routine shown in Figure 2 uses a system time-based sleep function to wait until the rampup procedure is completed. This approach severely suffers from potential race conditions: A fault shall be injected at a particular point of the rampup procedure. The test routine is running asynchronously to the server in a separate thread. To inject the fault in the right moment, the sleep time between starting the rampup procedure and the fault injection has to be tuned precisely. Otherwise, the fault may be injected in a different

* martin.hierholzer@desy.de

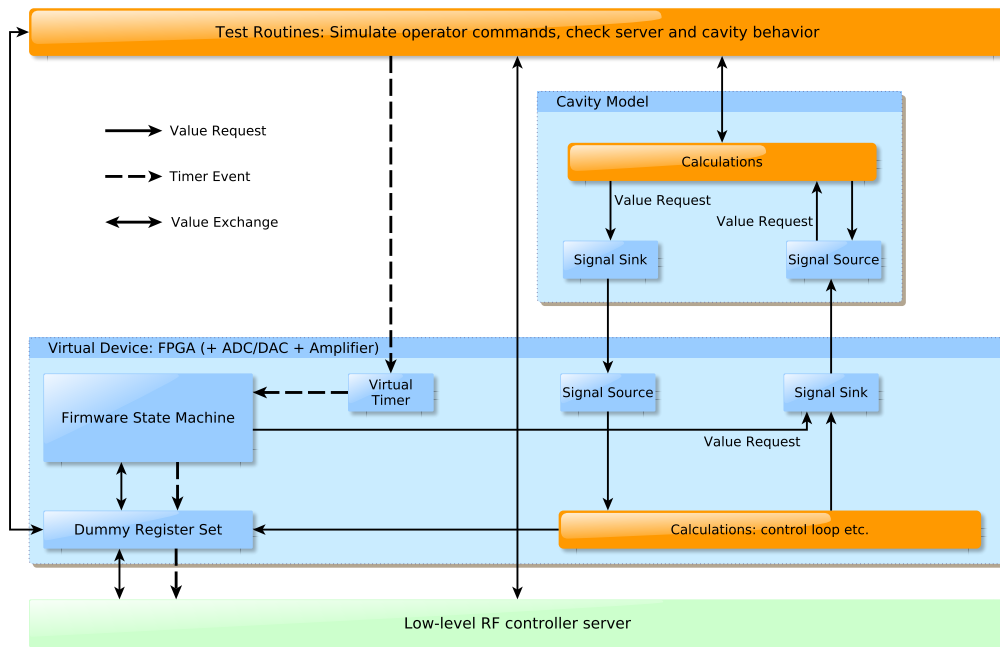


Figure 1: The layout of the test of the low-level RF controller server based on VirtualLab. The box at the bottom represents the low-level RF controller server to be tested.

```
// tell the server to ramp up the RF
doocsSet("//AUTOMATION/START_RAMPU", 1);

// wait until shortly before some safety check
sleep( (numberOfRfPulsesUntilCheck-1)*100ms + someExtraTime );

// inject a fault
cavityModel.injectFault();

// wait one RF pulse to perform the check
sleep(100ms);

// check if drive signal was switched off
BOOST_CHECK( cavityModel.driveSignal == 0.0 );
```

Figure 2: Pseudo-code of a simple test routine, which is not based on virtual time and thus subject to race conditions.

```
// tell the server to ramp up the RF
doocsSet("//AUTOMATION/START_RAMPU", 1);

// synchronously wait until before the safety check
virtualTimer.advanceTime(numberOfRfPulsesUntilCheck*100ms);

// inject a fault
cavityModel.injectFault();

// synchronously wait for one RF pulse
virtualTimer.advanceTime(100ms);

// check if drive signal was switched off
BOOST_CHECK( cavityModel.driveSignal == 0.0 );
```

Figure 3: Pseudo-code of a simple test routine based on virtual time to avoid race conditions.

RF pulse than intended, which might trigger some other, unexpected error handling. If the system is busy with other tasks, the wakeup might be delayed, or the server might take slightly longer than usual and the wakeup might relatively be early. Spurious false failures of the tests might occur as well as occasionally passing the test successfully despite of bugs in the code.

To eliminate these race conditions, the threads need to be synchronised properly. This can be done by introducing the concept of virtual time. Figure 3 shows a test routine, where the system time-based sleep has been replaced with a command controlling the virtual time. This command will instruct the virtual timer shown in Figure 1 to send as many timer events as necessary to move forward by the requested amount of virtual time. This will trigger the generation of data by the model. Also the appropriate number of RF pulse

interrupts will be sent to the control server to trigger its processing.

To make sure no race conditions can take place, it also has to be made sure that the server has finished processing before the command returns. The exact implementation for this may depend on the control system middleware and/or the application.

ACCOUNT FOR SLOW MODEL COMPUTATIONS

A typical signal sampling frequency for the RF control application is around 9 MHz. Today's CPUs are not capable of computing the simple cavity model at this frequency by far, especially if multi-cavity setups have to be taken into account. As shown in Figure 4, there are no samples recorded during the gap between the RF pulses. Typical pulse lengths are in

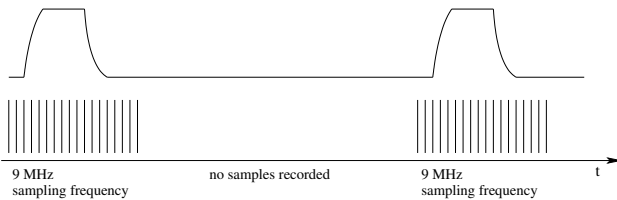


Figure 4: Typical signal sampling for pulsed mode operation. The ratio of pulse lengths and gap time between the pulses is not to scale.

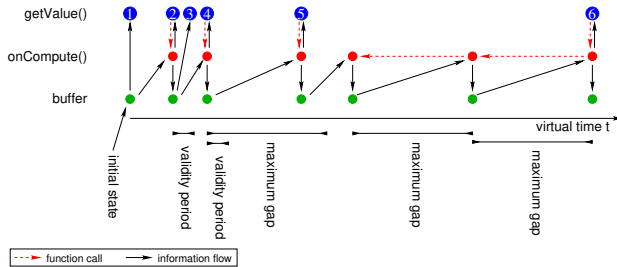


Figure 5: Sketch of the signal sources' efficient handling of request for new values. The blue circles on the top represent the requests via a call to `getValue()` incoming in the sequence of the numbers inside each circle. The red circles in the middle represent the calls to the callback function `onCompute()` which triggers computing a new model state. The green circles at the bottom represent the entries in the buffer storing the computed states for later re-use. New states are computed according to the configured validity period and maximum gap time. Request 3 does not require a computation, since it falls within the validity period of the previously computed value. Request 6 exceeds the maximum gap time, which triggers the computation of additional intermediate states.

the order of around 2 ms at a repetition rate of 10 Hz, thus 98 % of the computations can be saved.

Even a preliminary continuous wave setup of the low-level RF controls used for the ELBE accelerator [5] uses this sampling scheme in the interface between firmware and software, while the control loop is constantly running also in between the “pseudo pulses”. Since the performance of the RF control in the virtual test setup is unimportant for the software tests, the sampling frequency of the control loop can be strongly reduced in between the pseudo pulses. In more complex test setups, other model computations may be required at a totally different rate or potentially even only triggered by some kind of event.

To achieve the required flexibility for this and similar use cases, VirtualLab will not sample all components with the same, constant sampling frequency. Instead, a component will request a sample from its connected components as needed. This mechanism is built into the signal sinks and sources shown in Figure 1. The arrows between the sinks, sources and calculation blocks represent requests for new values. Since control loops and models often depend on the

history, each signal source must be able to return a value for any given time stamp within certain limits.

Each signal source has a smart, internal buffer to efficiently handle the incoming requests with minimal effort. Figure 5 shows how a signal source handles incoming requests. The signal source has a number of tuning parameters which allow to reduce the number of samples which need to be computed. Most important are the validity period, within which an already computed sample can just be reused, and the maximum gap time which governs the insertion of additional intermediate computations needed to keep the model precision within acceptable limits. The maximum gap time is especially important for the continuous wave setup, which makes sure that the model is computed also between the pseudo pulses with reduced sampling rate.

CONCLUSION AND OUTLOOK

The VirtualLab framework has been used successfully to develop tests for the low-level RF controller server for the ELBE accelerator at HZDR. All hardware has been replaced by virtual implementations, so the tests can be run on a standard PC. Its execution is triggered by commits to the source code management system or completed tests of dependencies used by the controller server. This presents a full continuous integration test chain.

By introducing virtual time, the problem of race conditions between the test routine and the tested software has been eliminated. To achieve an execution speed close to a real setup with actual hardware, model samples are computed only when needed. This mechanism is implemented in the signal sinks and sources which connect the different components of the virtual setup.

To further reduce the CPU load, an interpolation between the samples will be implemented in future. In addition, more complex test environments will be made possible by allowing to share virtual devices and model components across server executables and thus testing the interplay of multiple control servers.

REFERENCES

- [1] ChimeraTK, <http://github.com/ChimeraTK>
- [2] C. Schmidt *et al.*, "Real time control of RF fields using a MicroTCA.4 based LLRF system at FLASH", 19th IEEE Real-Time Conference, Nara, Japan, 2014.
- [3] M. Altarelli *et al.*, "XFEL: The European X-Ray Free-Electron Laser: Technical Design Report", DESY-2006-097, DESY, Hamburg, 2007.
- [4] PICMG®, "MicroTCA® Enhancements for Rear I/O and Precision Timing, MicroTCA.4 R1.0", 2011/2012.
- [5] M. Kuntzsch *et al.*, "First experience using a MicroTCA.4-based LLRF-controller driving the SSPA-based high power RF system at ELBE", Ninth CW and High Average Power RF Workshop, Grenoble, France, 2016.

DEVELOPMENT AND CURRENT STATUS OF A CARBORNE GAMMA-RAY SURVEY SYSTEM, KURAMA-II

M. Tanigaki*, R. Okumura, K. Takamiya, N. Sato, H. Yoshino, H. Yoshinaga, Y. Kobayashi
Research Reactor Institute, Kyoto University, Kumatori, Osaka 590-0494, Japan

Abstract

A carborne gamma-ray survey system, named KURAMA (Kyoto University RAdiation MApping system), was developed as a response to the nuclear accident at TEPCO Fukushima Daiichi Nuclear Power Plant in 2011. A CompactRIO-based system KURAMA-II has been developed as the successor of KURAMA, and served for various activities on the radiation monitoring in Eastern Japan. We continue developing KURAMA-II as a tool not only for the current monitoring activities, but also for the immediate responses in nuclear incidents in future. The current status and on-going developments of KURAMA-II will be introduced along with the recent status of the east Japan.

INTRODUCTION

The magnitude-9 earthquake in Eastern Japan and the following massive tsunami caused a serious nuclear disaster of Fukushima Daiichi nuclear power plant. Serious contamination was caused by radioactive isotopes in Fukushima and surrounding prefectures. KURAMA [1] was developed to overcome the difficulties in radiation surveys and to establish air dose-rate maps during and after the present incident. KURAMA enabled operations of a large number of in-vehicle units for large-scale surveys owing to its high flexibility in the configuration of data-processing hubs or monitoring cars. KURAMA has been successfully applied to various activities in the radiation measurements and the compilation of radiation maps in Fukushima and surrounding areas.

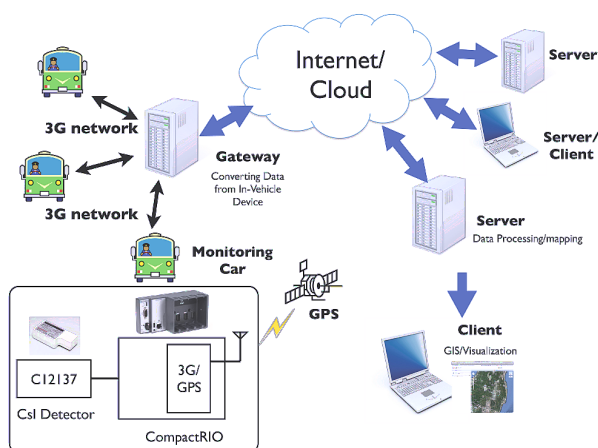


Figure 1: System outline of KURAMA-II.

As the situation becomes stabilized, the main interest in measurements moves to the tracking of the radioactive ma-

terials that have already been released into the environment surrounding the residential areas. Such monitorings can be realized efficiently if vehicles that periodically move around the residential areas, such as local buses, delivery vans or postal motorcycles, have compact and full-automated KURAMAs onboard. KURAMA-II [2] is designed for such purposes, characterized by its compactness, autonomous operation, and additional functions such as the measurement of pulse height spectrum. In this paper, the system outline of KURAMA-II as well as the results of continuous monitoring using KURAMA-II will be introduced.

SYSTEM OUTLINE OF KURAMA-II

Long term (in the order of tens of years) and detailed surveillances of radiation are required in the residential areas exposed to the radioactive materials by the nuclear accident. Such monitoring can be realized by implementing radiation monitoring units into moving vehicles in residential areas such as local buses, delivery vans or postal motorcycles. KURAMA-II is developed for such usages.

System outline of KURAMA-II is shown in Fig. 1. KURAMA-II stands on the architecture of KURAMA, but the in-vehicle part is totally re-designed for the autonomous, continuous operations in vehicles [3]. The platform is replaced from a conventional laptop PC to a CompactRIO controller of National Instruments to obtain better toughness, stability and compactness. The radiation detection part is replaced from the conventional NaI survey meter to a Hamamatsu C12137 [4], a CsI detector characterized as its compactness, high efficiency, direct ADC output and USB bus power operation. The mobile network and GPS functions are handled by a Gxxx module for CompactRIO by



Figure 2: In-vehicle unit of KURAMA-II. A CsI detector and a CompactRIO controller are compactly placed in a tool box with the size of 34.5 cm × 17.5 cm × 19.5 cm.

* e-mail: tanigaki@rri.kyoto-u.ac.jp

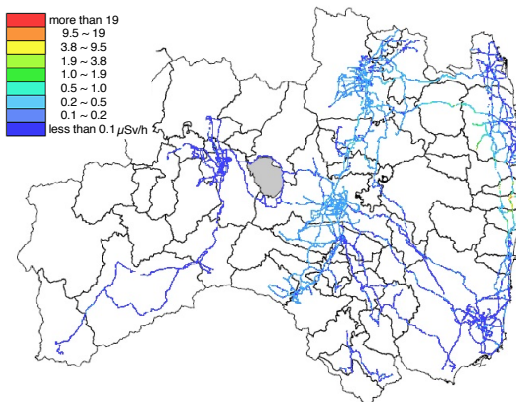


Figure 3: The result of continuous monitoring by local buses etc. in Fukushima as of the fourth week of September 2016. This kind of maps are released both in picture and in data files (csv and kmz) through the web on weekly-basis.

SEA. All components of the in-vehicle unit are placed in a small tool box for the better handling (Fig. 2).

The software for KURAMA-II is basically the same code as that of original KURAMA, with additional developments for newly introduced C12137 detector and Gxxx module, for the start up and initialize sequences fautonomous operation, and for a RESTful-based file transfer protocol.

All the data collected by in-vehicle units are shared over Dropbox, a cloud storage service. The gateway server is prepared as the interface between in-vehicle units and Drop-box because it does not currently support CompactRIO. A feasibility study of ownCloud with the implementation of data transfer protocol of KURAMA-II is on the way as the alternative cloud storage for KURAMA-II system, not only to eliminate the gateway server, but also to obtain more flexibility in the cooperative operations with other databases, such as GIS servers.

OPERATIONS OF KURAMA-II IN EASTERN JAPAN

Continuous Monitoring by Local Buses

One of the major application of KURAMA-II is the continuous monitoring in residential areas by local buses. A series of field tests on local buses started in December 2011 have finally evolved into the official project organized by Fukushima prefectural government under the collaboration of Kyoto University and JAEA [5]. As of October 2016, more than fifty KURAMA-II units are continuously operated throughout Fukushima prefecture. In-vehicle units are deployed not only to local buses, but also to official vehicles operated by Fukushima prefectural government for purposes other than radiation monitoring. In the case of local buses, the bus routes in each operation areas are completed over three to five days typically, based on the transportation plan determined by its respective bus operator. The air dose rate at 1 m above on the road is determined by multiplying the shielding factor of bus body determined by the comparison with the results of periodical carborne surveys by Japanese

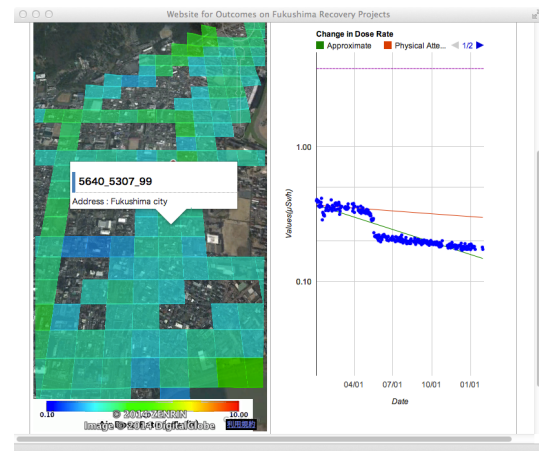


Figure 4: Typical example of the decontamination effect observed through monitoring by KURAMA-II on a local bus. In this figure, a drastic reduction of air dose rate along with the decontamination was observed.

government. The results from this project is released to the public from the web site on a weekly basis (Fig. 3). The changes of radiations in residential areas are successfully observed (Fig. 4).

Periodical Surveys by Japanese Government

The Nuclear Regulation Authority (NSR) in Japan conducts the periodic carborne surveys in Eastern Japan since 2011 [6, 7]. This project has been taken over to Nuclear Regulation Authority (NSR) because of the reorganization of atomic energy administration in Japan. In this project,

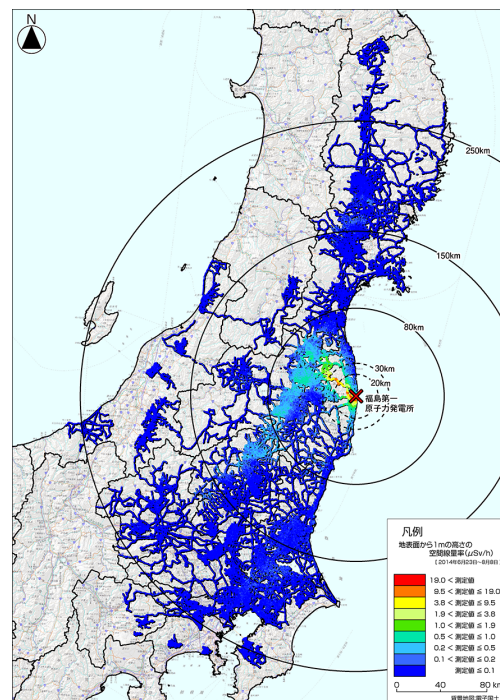


Figure 5: A typical map of air dose rates on roads measured by KURAMA-II from June to August in 2014 [7].

around one hundred KURAMA-II are deployed to the local municipalities in Eastern Japan twice a year, and the staff members in each municipality drive around their own municipalities with in-vehicle units (see Fig. 5). The summarized data is released through the websites [8, 9] as well as served for various analyses including the evaluation of ecological half-lives of radioactive cesium in environment and the long-term predictions of air dose rates in Fukushima [10] (Fig. 6).

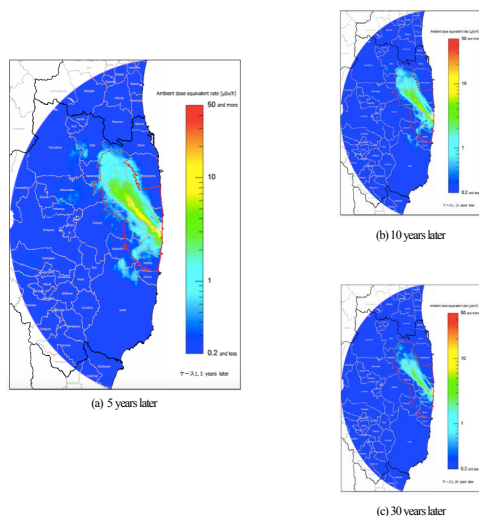


Figure 6: Prediction of air dose rate in the area within 80 km from the Fukushima Daiichi Nuclear Power Plant [10]. The parameters used for these predictions are determined by using the data of carborne surveys by KURAMA-II.

ON-GOING DEVELOPMENT

One important lesson we learned from the experience in the Fukushima accident is that the quality of data in a monitoring activity under emergency situation can be easily and severely suffered by human errors. For example, almost half of data was suffered by human errors in one of the large-scale monitoring activities performed by several hundreds of scientists, who were considered to be the specialists on this kind of activities. Efforts and costs to recover such errors sometimes become sufficient to terminate the activity itself.

Such human errors can be eliminated by implementing an autonomous data-handling function like KURAMA-II to any measurement devices used in emergency situations. We are working for KURAMA-mini, a survey meter with KURAMA-II function, as an device for such purpose. Feasibility studies on the monitoring scheme with KURAMA-mini in emergency situations are on the way (Fig. 7).

ACKNOWLEDGMENTS

Authors are grateful to Dr. Mizuno, Mr. Abe, Mr. Koyama and staff members of the KURAMA operation team at the Fukushima prefectural government for the continuous support to field tests of KURAMA in Fukushima. The development of KURAMA-II is adopted by “Japan recovery grant program” from National Instruments Japan. Authors are

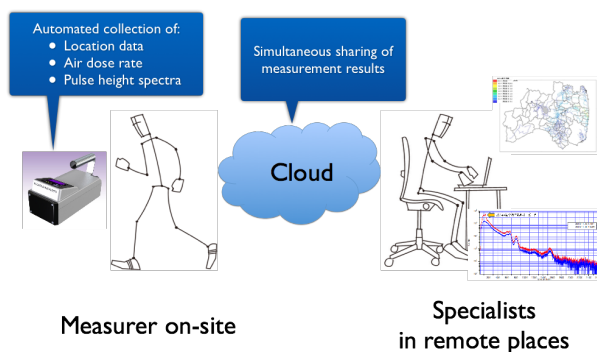


Figure 7: Expected usage of KURAMA-mini in emergency situations. Measurer just go to the site with KURAMA-mini, and the data autonomously collected by KURAMA-mini is shared with specialists in remote sites for proper analysis or decision making etc.

indebted to Mr. and Mrs. Takahashi and the staff members at Matsushimaya Inn, Fukushima, for their continuing support regardless of their severe circumstances due to the earthquake and the following nuclear accident.

REFERENCES

- [1] M. Tanigaki, R. Okumura, K. Takamiya, *et al.*, "Development of a car-borne γ -ray survey system, KURAMA", *Nucl. Instr. Meth. A* vol. 726, pp. 162-168, Oct., 2013.
- [2] M. Tanigaki, R. Okumura, K. Takamiya, *et al.*, "Development of KURAMA-II and its operation in Fukushima", *Nucl. Instr. Meth. A* vol. 781, pp. 57-64, May, 2015.
- [3] M. Tanigaki, Y. Kobayashi, R. Okumura, *et al.*, "Development of the Car-borne Survey System KURAMA", in *Proc. PCaPAC'12*, Kolkata, India, pp. 248-250, Dec. 2012.
- [4] C12137, Hamamatsu Photonics K.K., <https://www.hamamatsu.com/jp/en/C12137.html>
- [5] "Carborne survey in Fukushima prefecture" (in Japanese), <http://www.pref.fukushima.lg.jp/sec/16025d/soukou.html>
- [6] Press Release, "Results of Continuous Measurement of Air Dose Rates through a Vehicle-borne Survey by MEXT (as of December 2011)", March 21, 2012, MEXT, <http://radioactivity.mext.go.jp/en/contents/5000/4688/view.html>
- [7] Press Release, "Results of Continuous Measurement of Air Dose Rates through a Vehicle-borne Survey by MEXT (as of March 2012)", September 12, 2012, MEXT, <http://radioactivity.mext.go.jp/en/contents/6000/5637/view.html>
- [8] Vehicle-borne Survey in the Distribution Survey of Radioactive Substances (KURAMA) (in Japanese) <http://emdb.jaea.go.jp/emdb/portals/b131/>
- [9] As part of "Extension Site of Distribution Map of Radiation Dose, etc." <http://ramap.jmc.or.jp/map/eng/>
- [10] S. Kinase, T. Takahashi, S. Sato, *et al.*, in *Proc. of International Symposium on Radiological Issues for Fukushima's Revitalized Future*, Fukushima, Japan, pp. 40-43, May, 2015.

AUGMENTED USER INTERACTION

R. Bacher, DESY, Hamburg, Germany

Abstract

The advent of advanced mobile, gaming and augmented reality devices provides users with novel interaction modalities. Speech, finger and hand gesture recognition or even gaze detection are commonly used technologies, often enriched with data from embedded gyroscope-like motion sensors. This paper discusses potential use cases of those technologies in the field of accelerator controls and maintenance. It describes the conceptual design of an intuitive, single-user, multi-modal human-machine interface which seamlessly incorporates actions based on various modalities in a single API. It discusses the present implementation status of this interface (Web2cHMI) within the Web2cToolkit framework. Finally, an outlook to future developments and ideas is presented.

MOTIVATION

Zooming applications by performing a pinch gesture at touch-sensitive displays, talking with the personal assistant to retrieve information from the internet, or controlling video games through a gaming console recognizing arm and body motions are all popular and intuitive interface features currently in common use. These technologies, well known in the consumer market, have extremely enriched the way in which people interact with their devices. Even in the rather conservative market of industrial applications, novel interaction technologies are gaining in importance, e.g. to simplify quality assurance of manufacturing processes in the car industry or to improve the efficiency of warehouse logistics.

In addition, a novel concept known as “App” and optimized for these unique technological features has been introduced which has revolutionised the conceptual design, look-and-feel and handiness of graphical user applications.

Hardware commissioning and maintenance use cases might profit from such novel interaction capabilities (modalities). For instance the alignment of mirrors mounted on an optical table to adjust a laser beam spot often requires a “third hand”. Interacting with control applications via spoken commands could be an appropriate alternative. Likewise wearing rough and dirty working gloves during cooling water maintenance work is not adequate for touch sensitive devices. Interacting via hand or arm gestures might be a better choice. Accessing on-line documentation is often indispensable for efficient inspection work. Wearing see-through augmented reality glasses controlled by head movements displaying routing schemes alongside with control applications could substantially improve measurement operations in the field.

Even control room work provides use cases for novel interaction modalities. Remote-controlling an overhead mounted screen showing some overview or control

application panels might be considerably simplified by recognizing spatial gestures such as clenching a fist or snapping fingers. Beam steering requires uninterrupted eye contact with trend charts or other display updates. Controlling a virtual knob by recognizing hand rotation could eliminate the risk of losing device focus which often happens with mouse-based operations.

Today’s youth are more than familiar with these novel interaction capabilities and app-like user applications. Providing up-to-date tools for future control room operators and maintenance technicians appears to be a must.

A PARADIGM CHANGE

Today’s users of accelerator control applications have developed intuitions based on click-like interactions. In an accelerator control room the mouse is still the standard user input device to interact with graphical applications. Being well accepted by the operators it provides a very accurate pointing capability and standardized user actions normally associated with graphical widgets. Mouse-based interactions are highly reliable unambiguous single-user actions. They are best suited for complex applications containing a wealth of graphical widgets.

Thus the introduction of any new interaction capabilities will be accompanied by a serious paradigm change regarding how software programmers design graphical operations and maintenance applications and how operators or maintenance personnel interact with them.

Gesture-Based Interaction

Spatial hand- and arm gestures provide only a rough pointing capability, and experience shows that the user’s arm tends to fatigue quickly, a phenomenon known as “gorilla arm”. In addition head gestures such as turning or nodding might also be considered.

In practice only a very limited number of gesture types are available which are partially standardized and not a priori associated with graphical widgets. Consequently the design and look-and-feel of applications must accommodate these restrictions. A multi-page application design consistent with the app concept, where each page provides a well-confined and standardized functionality with an unambiguous gesture-to-action mapping, appears to be best suited.

In general hand- and arm gestures are less reliable and require a specific arming / disarming procedure to prevent the user from unwanted interaction.

Spatial gestures are not limited to single-user interaction only. It depends on the technology of the gesture recognition device used how many gestures from different persons can be tracked individually. Devices with embedded infrared stereo cameras, multi-axis gyro

sensors or muscle activity sensors are commercially available.

Speech-Based Interaction

In contrast to other interaction modalities the recognition of spoken commands does not provide any pointing capability at all.

On the one hand the huge word pool of human languages is a clear plus factor. On the other hand the context-dependent ambiguity and the language- or dialect-dependence of the vocabulary pose a big challenge for the recognition algorithms involved.

From the audio technical point of view the recognition of spoken commands might suffer from ambient noise and the interference of multi-user inputs.

To achieve a reliable speech recognition ability limiting the allowed vocabulary and ensuring an unambiguous word-to-action mapping is preferable. Similar to gesture recognition a specific arming / disarming procedure is capable to prevent the user from unwanted interaction.

A PRACTICAL EXAMPLE

This paper reports ongoing R&D work and describes a common single-user human-machine interface which seamlessly combines actions based on various modalities provided by input devices commonly available from the consumer market. It presents and discusses a platform-neutral Web-based interface implementation (Web2cHMI [1]) for accelerator operation and maintenance applications in the context of the Web2cToolkit Web service collection [2].

Web2cHMI defines a set of common user interactions comprising all actions needed to control Web2cToolkit-compliant Web applications such as application browsing, display zooming or executing commands associated with interactive graphical widgets. In general it can be considered as a prototype implementation exploring the advantages and disadvantages of these novel interaction modalities for accelerator control and maintenance applications.

In particular the Web2cToGo Web service [3] implements Web2cHMI and provides a test environment for identifying intuitive and handy user actions as well as investigating the proper structure, design and operability of multi-modal accelerator operation and maintenance applications. Web2cToGo embeds both Web2cViewer and

Web2cArchiveViewer Web application which are also members of Web2cToolkit Web service collection.

Supported Modalities

Web2cHMI supports various modalities which can be used simultaneously including

- 1D/2D flat gestures including single-finger actions (mouse) and single- or multi-finger gestures (touch-sensitive display)
- 2D/3D spatial gestures including hand-gestures (LEAP Motion controller (Figure 1) [4]), hand- or arm-gestures (Myo gesture control armband (Figure 2) [5]) and 3-axis (yaw, pitch roll) head movements (smart glasses)
- English spoken commands (Sphinx speech recognition [6]).

The LEAP Motion controller and the Myo gesture control armband are connected locally with their corresponding host device (desktop, notebook or tablet computers) through USB and Bluetooth, respectively. Currently supported smart glasses with gyroscope-based head tracking capability include Epson Moverio BT-200 [7] and Vuzix M100 (Figure 3) [8].

Supported Gesture Types

Web2cHMI recognizes various primitive, i.e. native or input device-specific gestures including

- Mouse: *Click, Move*
- Touch-sensitive display: *Tap, Move / Swipe, Pinch* (two fingers)
- LEAP Motion controller: *Key-Tap, Swipe, Open-Hand, Closed-Hand, Circle*
- Myo gesture control armband: *Double-Tap, Wave-Out / Wave-In, Fingers-Spread, Fist*
- Smart glass: *Move-Fast / Move-Slow, Roll*

In addition, enriched gestures formed by primitive gestures followed by moves or rotations etc. are supported.

All gesture-capable devices provide orientation data being used to position a virtual cursor label at an application window. Unlike mice or touch-sensitive displays gesture recognition devices such as LEAP, Myo or smart glasses with head tracking capability do not allow an accurate positioning of the cursor label.



Figure 1: LEAP Motion sensor.



Figure 2: Myo gesture control armband.



Figure 3: VUZIX M100 with gyroscope-based head tracking capability.

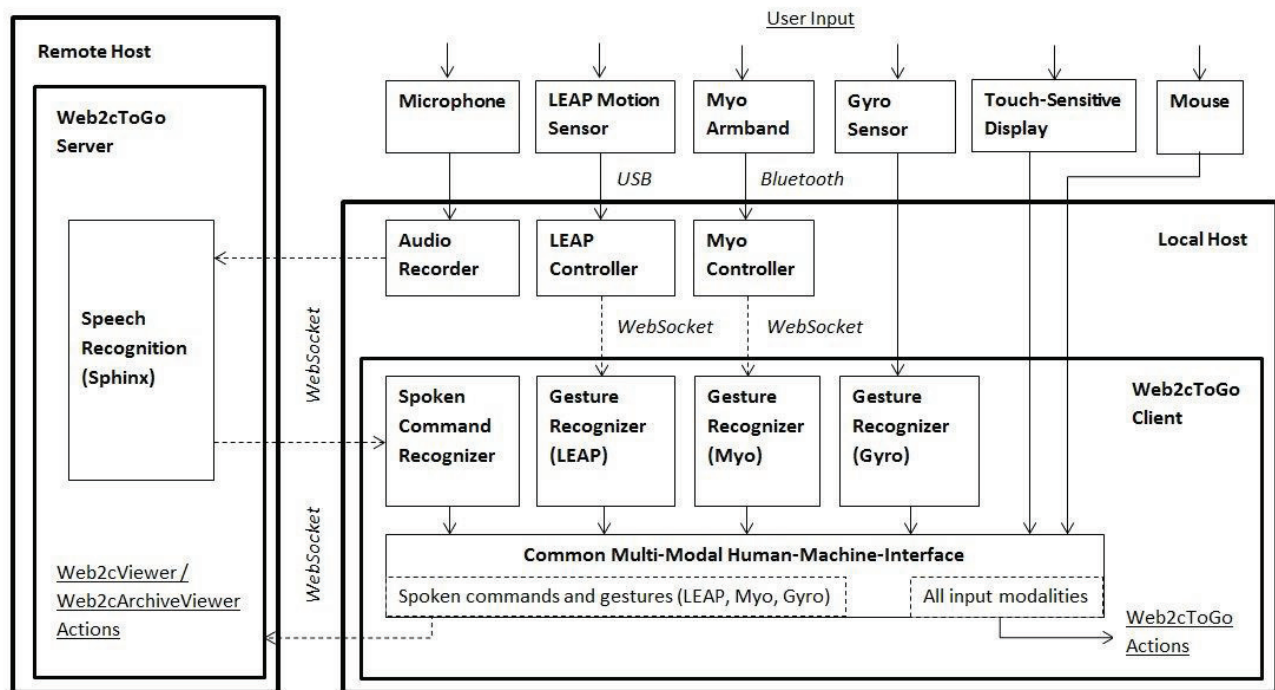


Figure 4: Web2cToGo / Web2cHMI user input data flow.

If applicable or required by ergonomics principles, different gestures may be applied by right or left handed individuals. If a gesture has been successfully recognized the next gesture recognition is momentarily inhibited while the cursor label is fixed to the center of the application window to notify the user.

In addition to avoid unwanted responses to unintentional gestures recognition ability must be armed / disarmed explicitly by the user:

- LEAP Motion controller: *Key-Tap / Key-Tap*
- Myo gesture control armband: *Double-Tap / Double-Tap*
- Smart glass: *Clockwise-Roll - Counter-Clockwise-Roll / Counter-Clockwise-Roll - Clockwise-Roll*

Common Human-Machine-Interface

Web2cHMI analyses user actions recorded by any modality-specific input device attached and maps the recognized gestures, spoken commands, head movements or even mouse clicks etc. to unambiguous commands. The recognized commands are used to control both the Web2cToGo framework application itself and the embedded Web2cToolkit-compliant Web applications. Figure 4 sketches the user input data-flow within Web2cToGo. Besides speech recognition which is performed by the Web2cToGo servlet (Java) at the Web server all recognition algorithms are implemented as client-side JavaScript modules being executed by an HTML5-compliant Web browser. Commands dedicated to an embedded application are redirected to the corresponding Web application.

Web2cHMI recognizes, for instance, among other user input the following corresponding Web2cViewer actions to increase a set value of an attached controls device in small steps using the slider widget (Figure 5):

- Mouse: *Click “>”-button and Click “Set Value”-button of the slider widget*
- Touch-sensitive display: *Tap “>”-button and Tap “Set Value”-button of the slider widget (right or left hand)*
- LEAP Motion Controller: *Clockwise Circle (right or left hand)*
- Myo gesture control armband: *Fist & Clockwise Rotation (right or left arm)*
- Gyro Sensor: *Upward Left-Tilted Move-Slow*
- Speech Recognition: *“More”*

Similarly, in order to choose an item above the currently chosen item in a list box of the Web2cArchiveViewer application the following actions can be performed (Figure 6):

- Mouse: *Click on item in list box*
- Touch-sensitive display: *Tap on item in list box (right or left hand)*
- LEAP Motion Controller: *Upward Long Swipe (right or left hand)*
- Myo gesture control armband: *Wave-Out & Upward Move (right or left arm)*
- Gyro Sensor: *Upward Move-Slow*
- Speech Recognition: *“Browse Up”*

A compilation of all commands implemented by Web2cHMI is given in the Appendix.

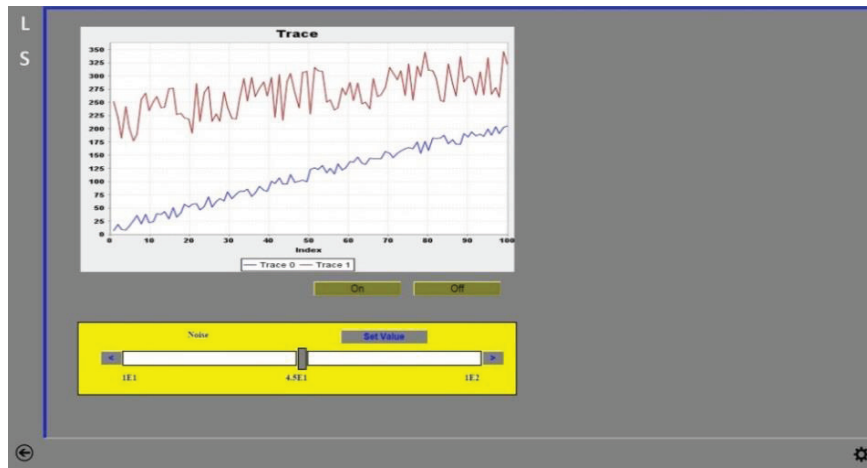


Figure 5: Web2cViewer embedded in Web2cToGo application (Operation View).

Similarly, in order to choose an item above the currently chosen item in a list box of the Web2cArchiveViewer application the following actions can be performed (Figure 6):

- Mouse: *Click* on item in list box
- Touch-sensitive display: *Tap* on item in list box (right or left hand)
- LEAP Motion Controller: *Upward Long Swipe* (right or left hand)
- Myo gesture control armband: *Wave-Out & Upward Move* (right or left arm)
- Gyro Sensor: *Upward Move-Slow*
- Speech Recognition: *“Browse Up”*

A compilation of all commands implemented by Web2cHMI is given in the Appendix.

Standardized Multi-Page Application Design

Due to the limited number of available gestures embedded accelerator controls and maintenance applications have to be split into individual pages which

provide well-known, standardized functionality in order to preserve an unambiguous gesture-to-command mapping.

Following this concept each Web2cViewer application page might contain a single widget instance of each of the following interactive widget types (Figure 5). According to their type, the interactive widgets are capable of performing a specific, predefined user action such as opening a vacuum valve or changing a set value of a power supply:

- On-type Button (user action = “On”)
- Off-type Button (user action = “Off”)
- Slider (user action = “Set Value”)
- Chart (user action = “Zoom Data”)

In addition a page might contain an unlimited number of passive Web2cViewer widgets such as labels or value fields.

Similarly, sets of interactive widgets have been defined for Web2cArchiveViewer application pages.

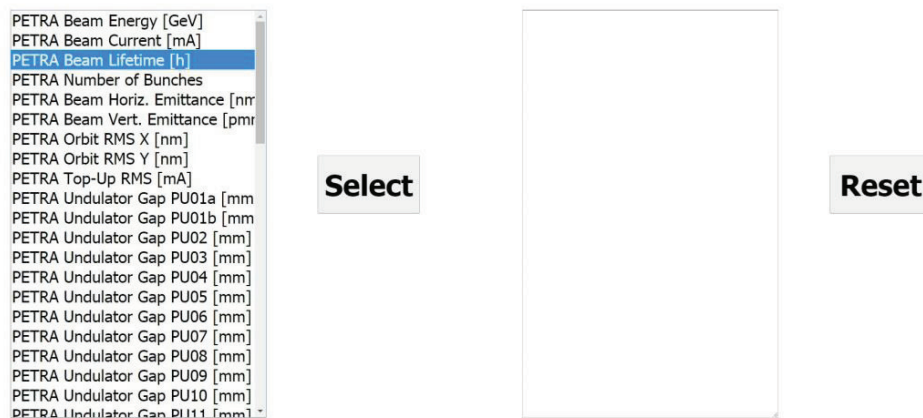


Figure 6: Web2cArchiveViewer embedded in Web2cToGo application (Operation View).

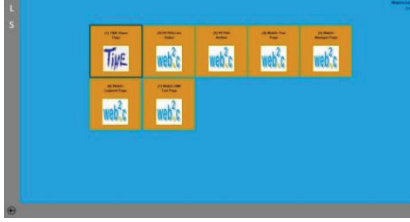


Figure 7: Web2cToGo (Explorer View).



Figure 8: Web2cToGo (Navigation View).

OPEN ISSUES

The work described in this paper is still an R&D project. It is inspired and abetted by a growing number of consumer and industrial use cases. It is expected that the acceptance level of modern user interaction technologies will steadily increase in the future.

To compete successfully with standard mouse-based interactions the quality and reliability of gesture and speech recognition procedures has to be improved. The applicability for accelerator controls and maintenance has to be studied in detail. The most intuitive, most common and best matching sets of gestures and spoken commands have to be defined and the usefulness and potential predominance of this approach have to be explored and proved in real field tests.

In addition, it is an open issue how a future control room preserving the unambiguity of operator's interactions might look. Will it be a camera supervised collective multi-user attentive environment or rather an environment for operators wearing individual single-user augmented reality glasses?

Finally, the next steps to introduce these novel interaction technologies for accelerator operations and maintenance have to be discussed. Is an intermediate step preferable providing, for instance, user applications adapted for touch pads or interactive tables yet keeping the traditional application look-and-feel? Or should a larger step be taken, where mouse-click interactions are omitted entirely, thereby skipping the familiar mouse-centric application design pattern?

APPENDIX

Implemented Web2cHMI commands include:

- *Web2cToGo (Explorer View)*: launch / display selected application, select application icon above below / right / left (Figure 7)
- *Web2cToGo (Navigation View)*: switch to operation view, switch to explorer view, browse to previous / next application, browse to previous / next application page, close current application, zoom in / zoom out current application page, fit current application page to window size, scroll down / scroll up / scroll right / scroll left current application page (Figure 8)

- *Web2cToGo (Operation View)*: Switch to navigation view (Figure 5)
- *Web2cViewer*: activate on-button / off-button, change set-value (any value / small positive step / small negative step, big positive step, big negative step), zoom in chart (left data area / center data area / right data area), reset chart zoom,
- *Web2cArchiveViewer (Single Entry / Composite Entry Data Selector)*: choose channel above / below chosen channel, select chosen channel, clear all entries in list of selected channels (Figure 6)
- *Web2cArchiveViewer (Date and Time Span Selector)*: choose next / previous month of year, select day, select day above / below / right of / left of selected day, select time span above / below selected time span, select default time span
- *Web2cArchiveViewer (Single Data Chart)*: retrieve data, retrieve data from next / previous time interval, zoom in chart (left data area / center data area / right data area), reset chart zoom
- *Web2cArchiveViewer (Multiple Data Chart)*: retrieve data, retrieve data from next / previous time interval, zoom in chart (left data area / center data area / right data area), reset chart zoom, scroll down / scroll up visible pane
- *Web2cArchiveViewer (Data Table)*: retrieve data, retrieve data from next / previous time interval

REFERENCES

- [1] R. Bacher, "A Multi-Modal Human-Machine-Interface for Accelerator Operation and Maintenance Applications", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, paper WEM308, pp 677.
- [2] Web2cToolkit, <http://web2ctookit.desy.de>
- [3] R. Bacher, "Web2cToGo: Bringing the Web2cToolkit to Mobile Devices", in *Proc. PCaPAC'12*, Kolkata, India, Dec. 2012, paper WEIC01, pp. 4.
- [4] LEAP, <https://www.LEAPmotion.com>
- [5] Myo, <https://www.myo.com>
- [6] Sphinx-4, <http://cmusphinx.sourceforge.net/sphinx>
- [7] Epson Moverio BT-200, <https://www.vuzix.com>
- [8] Vuzix M100, <https://www.epson.com>

A CYTHON INTERFACE TO EPICS CHANNEL ACCESS FOR HIGH-LEVEL PYTHON APPLICATIONS

J. Chrin, Paul Scherrer Institut, 5232 Villigen PSI, Switzerland

Abstract

Through the capabilities of Cython (a Python-like programming language with the performance of C/C++), a Pythonic interface to an in-house C++ Channel Access (CA) library, CAFE, has been developed, thereby exposing CAFE's numerous multifaceted and user-friendly methods to Python application developers. A number of particularities of the PyCafe extension module are revealed. These include support for (i) memoryview and other data types that implement the new Python buffer protocol (allowing data to be shared without copying), (ii) native thread parallelism, and (iii) pointers to callback functions from wherein CAFE methods may be effortlessly executed in asynchronous interactions. A significant performance improvement is achieved when compared with conventional Pythonic CA libraries. The PyCafe interface has been realized within the context of high-level application development at SwissFEL, Switzerland's X-ray Free-Electron Laser facility.

INTRODUCTION

The Python programming language is enjoying an increasing profile within accelerator facilities, and is, furthermore, the topic of a number of tutorials and contributions to this workshop. It is both an interpreted and object oriented programming language, with dynamic typing and binding. Its straight-forward syntax is advocated for promoting conciseness and readability, thus enhancing rapid code development. The availability of third-party computational modules such as NumPy [1] and SciPy [2], coupled with modules for the visualization of multi-dimensional data [3], add to its appeal as a platform for beam dynamics applications at SwissFEL, Switzerland's X-ray Free-Electron Laser [4], and elsewhere, e.g., [5]. The very nature of Python's dynamic semantics, however, inevitably results in an adverse effect on performance, especially with respect to low-level computation involving mathematical operations and looping constructs. Faster processing time may be achieved by using Python modules written largely in C/C++ as is the case for NumPy and SciPy, else by wrapping pre-existing C/C++ code or libraries into Python. The latter may be accomplished in a number of ways. The more traditional approach is to use Python's C Application Programming Interface (Python/C API), which necessitates a greater expertise in C than in Python. Other methods include the use of specialized tools such as SWIG, SIP, CFFI, the Boost.Python C++ library or the ctypes package from Python's own standard library. While all of these possibilities have their own specific flavour and target, the focus of this work is to explore the emerging Cython [6] technology to provide a high-performing Python interface to EPICS (Experimental Physics and Indus-

trial Control System) [7, 8], through a well-tested, in-house, C++ Channel Access (CA) client library, CAFE [9–12]. In this way, a full complement of CAFE's multifaceted CA methods are readily made available to application developers in Python.

THE CASE FOR CYTHON

Cython is a high-level, object-oriented, and dynamic programming language, whose principle merit is to provide a Python-like style of coding while maintaining the performance level of C. Cython is equipped with the cython compiler that translates Cython source code into optimised C/C++ code, which in turn is compiled into a Python extension module. Performance enhancement is achieved through Cython's declaration of static C (and certain Python) type variables and with its ability to interface to C/C++ libraries, thus bypassing execution of bytecodes by the Python Virtual Machine (VM). Its primary use cases can thus be anticipated. Speed critical, CPU-bound, segments of Python code may be shifted into the Cython domain, wherein the dynamic Python runtime semantics are replaced by the static C semantics, and existing code in external C/C++ libraries may be natively accessed [13].

Cython is a fully-fledged language that is a super-set of Python, making it enticingly simple to customize, simplify or otherwise Pythonize interfaces as they are wrapped. Cython keeps abreast with developments in the Python programming language, accommodating new features as they appear, as exemplified by its support for the new Python protocol buffer and the *typed memoryview* object. Cython's ability to generate highly optimized code, when compared with those of other wrapping tools, also places it in good stead. It is thus well suited for the task of exposing the many, well-tested, methods of the C++ CAFE library to Python.

THE PyCafe PYTHON MODULE

The PyCafe extension module exposes the Cython interface to the CAFE C++ library (CyCafe). CAFE provides a concise, complete, and clean interface with minimal details of the low-level CA implementation propagating to the user, and an abstract layer tailored for beam dynamics applications. The underlying code base ensures that CAFE clients are entirely decoupled from the CA servers, allowing, e.g., data aggregator daemons and Graphical User Interfaces (GUIs) to function in every eventuality, i.e., irrespective of changes to the connection state of the EPICS channel. CAFE's multifaceted interface further provides the flexibility that eases CAFE's use as CA host to C/C++ based scripting and domain-specific languages [11, 12]. In CyCafe, for instance, methods

with Python lists as arguments map directly onto CAFE's C++ vector counterparts.

The near-full complement of the Python language, with its standard and third-party libraries, together with the given Cython constructs, are at the developers disposal for wrapping C libraries in Cython. This includes support for data types that implement the new Python buffer protocol which allows for their data to be shared without the need for copying, vastly improving performance in large data transfers. In creating the CyCafe wrapper class, a proper handling of CPython's Global Interpreter Lock (GIL) is critical to achieving the thread based parallelism that is essential for establishing callbacks from within Python in asynchronous operations.

Memoryview and Typed Memoryview Objects

The introduction of a new buffer protocol in Python allows a number of Python built-in types, e.g., bytes, bytearray, and extension types, such as the array.array and numpy.ndarray arrays, and other objects that implement the protocol, to share their data without copying. The C level buffer interface may further be exposed as a Python memoryview object that, among other attributes, supports slicing and indexing to expose a subview of the underlying data. They are hence well suited to data arising from, and destined to, EPICS waveform and subArray record types.

Cython can similarly extend the buffer protocol to work with data arising from external libraries through the C-level *typed memoryview* object, which projects, and expands on, the Python memoryview interface. CyCafe consequently provides interfaces that recognize such memoryview and memoryviewslice data types.

Thread Based Parallelism

Cython uses the Python/C API to access C-level code. CPython's memory management is not, however, thread-safe, necessitating a dedicated mutex, the GIL, to ensure that only one native thread executes Python bytecodes at any given time. External C code that does not interact with Python objects can, however, be executed without the GIL in effect, thus achieving thread-based parallelism. All methods that access the low-level hardware through channel access are done so in a non-GIL context, i.e., with the GIL released. Without taking this necessary step of releasing the GIL, PyCafe will otherwise hang, particularly in cases where callbacks are involved.

C Function Pointers and Callbacks

Cython supports C function pointers allowing C functions, that take function pointer callbacks as input arguments, to be wrapped [14]. This feature allows users to pass a Python function, created at runtime, to control the behaviour of the underlying C function. Using this methodology, a Python callback function may be easily supplied for any asynchronous CA interaction. The requirement for the external CAFE library to call into the CyCafe code necessitate the inclusion of a Cython generated header file into the

C++ CAFE library. The header contains the functions and methods purposely declared by Cython's `api` keyword. The Python import mechanism is then invoked within CAFE to dynamically expose the `api`-declared functions.

THE PyCafe INTERFACE

Listing 1 (lines 1-3) shows how the PyCafe extension module is imported from within a Python application and how the Cython CyCafe object is instantiated. The CyCa class exposes various CA and CAFE enumerated types and status codes. The notion to establish CA connections to PVs ahead of time, and in combination (lines 5-12), is optional but represents good practise, not least as network traffic and total wait time are reduced. Subsequent CA operations on PVs may be undertaken by making reference to the PV name or its handle (as returned by `cafe.open`), which is an object reference to CAFE's allotted cache for the associated PV. If the CA connection methods are not explicitly invoked, they will otherwise be made autonomously at the time of the first PV data operation. Helper methods (not shown) provide information on the association between a given PV and its handle. CyCafe defined exceptions (`PyCafe.CyCafeException`) are, by necessity, thrown under the guise of the generic Python `RuntimeError` exception, but may nevertheless be explicitly tested for by the user (not shown). Alternatively, PyCafe may be configured with the exception handler disabled for single channel operations, such that status codes are instead returned. On application exit, the `cafe.terminate` method (line 60) closes all PV connections, deletes their handles, and releases all CA resources. A few selected PyCafe data read, write and monitor transactions are presented.

Read and Write Operations

Listing 1 shows the Python syntax for selected synchronous and asynchronous, single and multiple channel data transactions through PyCafe. Methods operating on scalar values support the two basic Python numeric types, namely float and int and the string type, `str`. The Python equivalent of the PV's native data type is the default type in data retrieval operations, unless otherwise specified by the `dt` argument keyword (line 16). Where timestamps and alarm conditions are required, methods that return the `pvdData` struct are invoked (lines 21-22).

A number of data retrieval operations are accompanied by an equivalent 'cache' method that retrieves the last value written into CAFE's internal buffer. These are typically used in conjunction with asynchronous data access interactions and, if necessary, will wait until the asynchronous operation has provided a value or a timeout has been reached (cf. `future` class). Since the underlying CAFE C++ library separates data acquisition from data representation, data of any type may be requested from cache (lines 43-46).

Data from EPICS waveform and subArray records are retrieved using the `cafe.getArray` method (lines 25-26). In addition to the standard Python built-in array types, Cython array features provide support for NumPy arrays and *typed*

memoryviews. The array type may be indicated through the *art* argument keyword, which may take on one of a number of self-explanatory options (line 17). Cython's *typed memoryview* is converted to a regular Python *memoryview* by *CyCafe* when passed as a return value.

Control system parameters may be retrieved as shown (lines 27-28). Since these data are typically static it is usually sufficient to retrieve them from cache, which is always populated on (re-)connection, if not otherwise.

The *set* method interrogates the form of the data input argument and can accommodate any meaningful type (lines 29-30).

The aggregation of channels, whether related or unrelated, into a collection allows several requests to be delivered within a single method invocation, thereby minimizing network traffic and increasing efficiency. Such methods are shown for scalar (lines 36-38) and compound (lines 39-41) data sets. The data container here is itself a Python list, which may legitimately contain elements of different data types (as may arise, e.g., in data retrieval operations that render the data from unrelated channels in their native type). Compound operations differ from their scalar counterparts in that they may also contain a list within a list, e.g., to accommodate waveforms.

Channels may, alternatively, be gathered into a named group to profit from the EPICS synchronous group functionality. Here the group acts as a single logical software entity and subsequent transactions are invoked through intuitive methods that reference the group either by name or handle (lines 48-55). Where associated timestamps and alarm status and severity data are required, a *pvgroup* object may be returned that contains a sequence of *pvd* objects. While synchronous groups are simple and inexpensive to create, they have the inherent disadvantage that a timeout on a group operation is unable to inform on the offending channel(s), consequently rendering the returned data unreliable. (The underlying CAFE library, incidentally, operates a self-regulating timeout policy to increment timeouts in cases where the setting has not been optimized for the local network capacity). To overcome this deficiency, where data from synchronous group operations cannot be verified due to a timeout, a multiple channel transaction, i.e., *cafe.getCompoundPVList* (lines 57-58), is autonomously invoked and the resulting data is repackaged into the *pvgroup* object. In this way, and oblivious to the end-user, data integrity and error reporting are ensured for each individual channel within the group. Indeed, use of this latter method may be preferred by some users over the synchronous group option.

The multiple channel methods presented do not throw CAFE exceptions requiring the user to first check on the overall status of the method invocation; only on error need the status code for the individual channels be examined.

Monitors

A Python callback function may be easily supplied to any asynchronous operation, whether this be a *set*, *get*, or

Listing 1: PyCafe Read/Write Examples

```

1  import PyCafe
2  cafe = PyCafe.CyCafe()
3  cyca = PyCafe.CyCa() #ca enums, status codes
4
5  pvList=['pv1','pv2','pv3',...]
6  waitTime=1.0 #seconds
7  try:
8      cafe.openPrepare()
9      hList = cafe.open(pvList) #ret. handles
10     cafe.openNowAndWait(waitTime)
11 except Exception as e:
12     ...
13
14 #<handlePV> in {hList[0], pvList[0]}
15 #<hPVList> in {hList, pvList}
16 #dt = {'native', 'int', 'float', 'str'}
17 #art = {'memoryview', 'numpy', 'array', ...}
18 try:
19     #get value in native (default) type
20     value = cafe.get(handlePV)
21     #returns structured data, value as float
22     pvData = cafe.getPV(handlePV, [dt='float'])
23     #waveform, returns list in native type
24     valList = cafe.getList (handlePV, [dt='native'])
25     #waveform, returns memoryview in native type
26     mv = cafe.getArray(handlePV, [art='mv'])
27     #control parameter data from cache
28     pvCtrl = cafe.getCtrlCache(handlePV)
29     #write operation for scalars and waveforms
30     cafe.set(handlePV, value) #value, any data type
31 except Exception as e:
32     ...
33
34 #synchronous multiple channel operations
35 #s gives overall status
36 #vList,sList, individual scalar values/status
37 vList,sList = cafe.getScalarList(<hPVList>)
38 s,sList = cafe.setScalarList(<hPVList>, vList)
39 #valList may contain lists within a list
40 valList,s,sList = cafe.getCompoundList(<hPVList>)
41 s,sList = cafe.setCompoundList(<hPVList>, valList)
42
43 #asynchronous multiple channel operations
44 s,sList = cafe.getAsyn(<hPVList>)
45 #apply any future cache method
46 pvData = getPVCache(<hPVList[0]>, [dt='float'])
47
48 #synchronous group operations
49 s = cafe.defineGroup('groupName', pvList)
50 groupHandle = cafe.openGroup('groupName')
51 #<gHandleName> in {groupHandle,'groupName'}
52 valList,s,sList = cafe.getGroup(<gHandleName>)
53 s,sList = cafe.setGroup (<gHandleName>, valList)
54 #returns a pvgroup object, with native data types
55 pvgroup = cafe.getPVGroup(<gHandleName>)
56
57 #returns pvgroup from a multiple channel operation
58 pvgroup = cafe.getCompoundPVGroup(<gHandleName>,
59                                     [dt='native'])
60 cafe.terminate() #tidy up

```


Listing 2: PyCafe Monitor Example

```

1  ...
2  #Callback function
3  def py_callback(handle):
4      #Any method that retrieves data from cache
5      pvData=cafe.getPVCache(handle, [dt='str'])
6      #user supplied code
7      ...
8      return
9  ...
10 #start monitor with user supplied callback
11 mon=cafe.monitorStart(<handlePV>, cb=py_callback,
12                       [dbr=cyca.CY_DBR_TIME],
13                       [mask=cyca.CY_DBE_VALUE|cyca.CY_DBR_ALARM])
14 ...

```

monitor. This is highlighted in Listing 2 where a monitor on a PV is activated. Optional keyword arguments govern the behaviour of the monitor. A notable aspect of the PyCafe interface, here, is that only the handle (i.e., object reference) is, and need be, reported back to the callback function. Since the CAFE API takes the provision to cache the data in its internal storage, the user may call upon any one of a number of CAFE methods that retrieve data directly from the cache. The precedence of sifting through Python dictionaries is dispensed with. A typical use case of the callback would be to trigger the passage of data to a graphical widget.

The CAFE connection event handler has also been configured to trigger the given callback in the event of a channel disconnection or reconnection.

PERFORMANCE IMPROVEMENTS

Cython, in certain use cases, has the potential to enhance performance for CPU-bound operations by several orders of magnitude. Performance differences between Python and Cython are significantly decreased, however, for memory-, input/output-, and network-bound operations. As channel access transactions involve movement of data across the network, the resulting latency is dominated by the data transfer. This, consequently, allows for some leeway in the design of the interface. For instance, in certain cases, static type declaration for arguments can be omitted, preserving flexibility without detriment to performance. This is evident in methods that accept either a process variable name or object handle as a positional argument. Nonetheless, PyCafe, due to Cython's generated optimized C++ code, still manages a factor of four improvement in performance for a single scalar retrieval when compared with the Python/C API, and a 40% performance enhancement compared with ctypes, which are subject to their Python call overhead. For large waveforms, e.g., $\sim 10^6$ elements, the performance difference between Python and Cython essentially vanishes, although interfaces exploiting the memoryview array type result in a gratifying factor of two performance gain when compared with numpy.ndarray.

SUMMARY

The Cython programming language has been used to interface the external C++ CAFE library to Python, thereby providing a well-tested, high-performance, and extensive channel access interface to high-level beam dynamics application developers at SwissFEL. The project software may be downloaded from the CAFE website [9], where further examples of various usages are given.

ACKNOWLEDGMENTS

The author is grateful to Masamitsu Aiba, whose extensive use of PyCafe in the SwissFEL virtual accelerator, deploying a few thousand EPICS soft channels, led to improvements to CAFE's functionality and verified its Cython interface.

REFERENCES

- [1] NumPy, <http://numpy.org>
- [2] SciPy, <http://scipy.org>
- [3] J. D. Hunter, "Matplotlib: A 2D graphics environment", in *IEEE Computing in Science and Engineering*, vol. 9, no. 3, pp. 90–95, May/Jun. 2007, doi:10.1109/MCSE.2007.55; matplotlib, <http://matplotlib.org>
- [4] *SwissFEL Conceptual Design Report*, R. Ganter, Ed. PSI, Villigen, Switzerland, Rep. 10-04, Version Apr. 2012.
- [5] T. Zhang, J. H. Chen, B. Liu, and D. Wang, "Python-based high-level applications development for Shanghai soft X-ray free-electron laser", in *12th Int. Computational Accelerator Physics Conf. (ICAP'15)*, Shanghai, China, Oct. 2015, pp. 23–25, doi:10.18429/JACoW-ICAP2015-MODWC4
- [6] Cython C-Extensions for Python, <http://cython.org>
- [7] EPICS, <http://www.aps.anl.gov/epics/>.
- [8] J. O. Hill and R. Lange, "EPICS R3.14 Channel Access Reference Manual", <http://www.aps.anl.gov/epics/docs/ca.php>
- [9] CAFE, <http://ados.web.psi.ch/cafe/>.
- [10] J. Chrin and M. C. Sloan, "CAFE, A modern C++ interface to the EPICS channel access library", in *Proc. 13th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'11)*, Grenoble, France, Oct. 2011, paper WEPKS024, pp. 840–843.
- [11] J. Chrin, "MATLAB objects for EPICS channel Access", in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper MOPPC146, pp. 453–456.
- [12] J. Chrin, "An update on CAFE, a C++ channel access client library and its scripting language extensions", in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, pp. 1013–1016, doi:10.18429/JACoW-ICALEPCS2015-WEPGF132
- [13] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, "Cython: The best of both worlds", *IEEE Computing in Science and Engineering*, vol. 13, no. 2, pp. 31–39, Mar./Apr. 2011, doi:10.1109/MCSE.2010.118
- [14] K. W. Smith, in *Cython*, Sebastopol, CA, USA: O'Reilly Media, Inc., Jan. 2015, pp. 128–134.

RECENT BEAMLINE INTERLOCK SYSTEM AND STARS AT THE PHOTON FACTORY

T. Kosuge, H. Nitani, H. Ishii, Y. Saito, Y. Nagatani,
High Energy Accelerator Research Organization, Tsukuba, Ibaraki, Japan

Abstract

More than 20 beamlines are installed at the Photon Factory for synchrotron radiation research. Each beamline is equipped with an interlock system to protect users from radiation hazards and avoid vacuum-related troubles. The system is controlled by a programmable logic controller (PLC). Currently, touch panels and their corresponding communication protocols comprise the user interface. They work satisfactorily but are expensive.

We have developed a new type of beamline interlock system that has a PC user interface based on a simple transmission and retrieval system (STARS). We will describe the details of this new system as well as STARS.

BEAMLINE INTERLOCK SYSTEM

An interlock system is installed on every beamline at the Photon Factory in order to protect the users from radiation hazards, maintain a vacuum environment for the beamline, and protect the beamline components from high heat load damage.

Every beamline interlock system works with a programmable logic controller (PLC). The following three types of interlock systems run at the Photon Factory:

- Legacy system with an LED panel
- Field bus and touch panel user interface
- Newly developed field bus and PC-based user interface

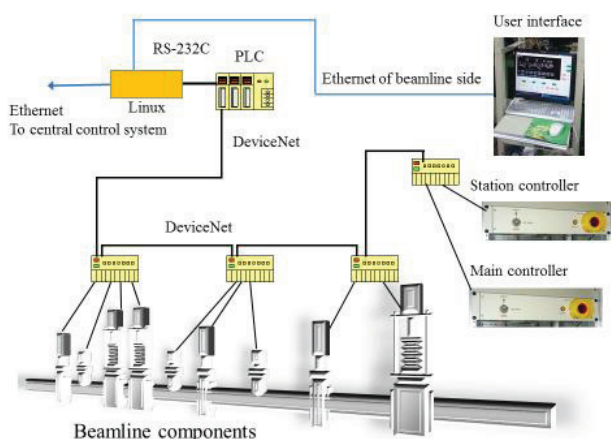


Figure 1: Beamline interlock system with field bus and PC-based user interface.

In 2015, a prototype of the beamline interlock system with PC user interface was installed onto a beamline named BL-19. We checked the reliability and effectiveness of this system and after fixing a few problems, we succeeded in the development of a new and satisfactory beamline interlock system as shown in Figure 1. The PC-

based user interface of the beamline interlock system is provided with a simple transmission and retrieval system (STARS) and it is flexible.

All the legacy systems and touch panel user interfaces will be replaced with the newly developed PC-based user interface in the future. The replacement will be done annually.

STARS

STARS [1, 2] is an extremely simple message transferring software for small-scale control systems. It is installed in various systems at the Photon Factory such as the beamline control system, room access control system, and key handling system.

STARS Server and Clients

STARS consists of a server program (STARS server) and client programs (STARS clients). The server is written in Perl and works on various operating systems (Linux, Windows, Macintosh, etc.). It is a small program that can run on low power computers such as embedded Linux. STARS provides flexible system design capabilities to a developer.

STARS clients are connected to the STARS server using a TCP/IP socket, and they communicate with other clients through the STARS server using text-based messages. STARS users can develop a control system in their preferred programming language on various operating systems if they can handle the TCP/IP socket and text. Addition a new function to the system is possible by making a new STARS client and connecting it to the STARS server. STARS clients can be connected or disconnected without stoppage of the system.

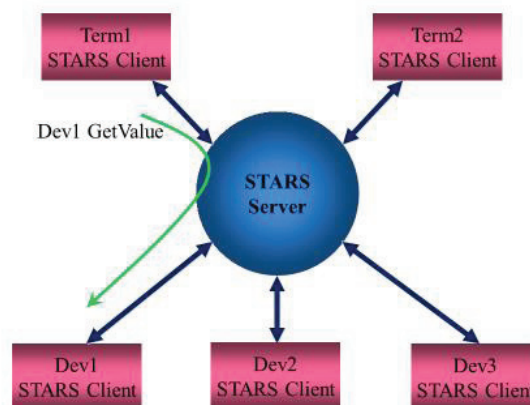


Figure 2: Example of message transferring on STARS.

Communication between STARS Clients

Each STARS client has its own node name, which is used to indicate the destination of the message. If a STARS client sends a message with a node name to the STARS server, the message is delivered to a corresponding STARS client. Figure 2 shows an example of message transferring on STARS. In the example, a client named “Term1” sends a message “Dev1 GetValue” to the STARS server, and the message is delivered to “Dev1” by the STARS server.

NEW BEAMLINE INTERLOCK SYSTEM WITH STARS

The monitoring and logging status of the beamline interlock system is very important to keep the system stable and facilitate maintenance. Originally, STARS was used for the central control system of the beamline interlock system, and a STARS client named “PLC interface” was used for the PLC interface. We expanded the “PLC interface” to support more signal counts. Figure 3 shows the software layout around the PLC interface.

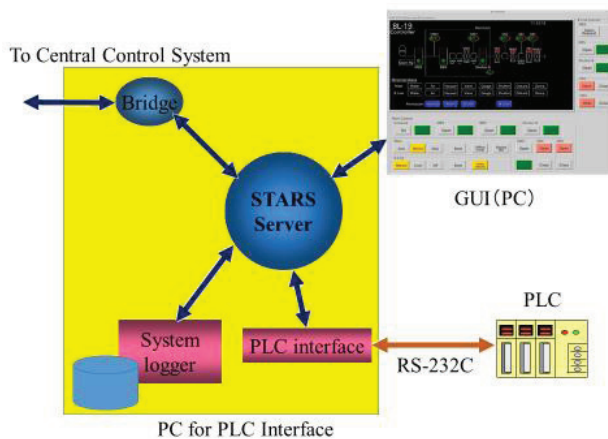


Figure 3: Software layout.

Development of the New PLC Interface

The new PLC interface is a STARS client program that can handle signals previously used by the touch panel. It communicates with the PLC through an RS-232C interface. All status signals are transported when the program is initialized and only the change of information is transmitted if the status has been changed in the PLC. This solution has enough speed using the RS-232C interface.

All interlock logic (radiation safety, etc.) is provided by the PLC, and STARS only provides the user interface.

Advantage of STARS for Beamline Interlock System

We already verified the stability and reliability of STARS with the central control system. STARS has a client certification and command suppression that works effectively to keep a secure beamline interlock system. The useful commonly provided STARS clients such as the

system logger are also effective for the development of beamline interlock system.

Client Certification

STARS has a simple certification checking procedure at connection time as follows:

- Host name checking: Is the host name or IP address of the STARS client found in the allowed list?
- Keyword checking: A STARS client must send a corresponding keyword, specified by the STARS server in the keyword list.
- Node name and host name checking (optional): Specification of the hostname for the STARS client is possible.

This simple procedure works effectively to avoid disconnection of the STARS client and helps maintain beamline interlock system reliability.

Command Suppression

The command suppression function of STARS is used to prevent commands such as “write” or “set” from unauthorized STARS clients. To protect the STARS client, “read only” is possible with this command suppression function. An example of the configuration file (command_deny.cfg) is shown below.

```
#Command deny list
hallmon-br>[AGS]
hallmon-br>System disconnect
```

A regular expression is used for configuration. The first line, which starts with “#,” is a comment. In the second line, the access from “hallmon-br” to the client, which starts without “G” or “S,” is suppressed. In the third line, the “disconnect” command of the STARS server from “hallmon-br” is suppressed.

Figure 4 shows an example of a command suppression function.



Figure 4: Command suppression.

In the example, the access to “GA1” is allowed, and the client can obtain valid data from “GA1” as follows:

```
GA1 GetValue
GA1>hallmon-br @GetValue 1.9400E-01
```

However, the access to “plcif-bl04” is suppressed as follows:

```
plcif GetValue
System>hallmon-br @GetValue Er: Command
denied.
```

System Logger

The system logger is a common STARS client that fetches all the STARS server output messages and writes to the log file. It helps in the troubleshooting the beamline interlock system. In the new interlock system, the number of signals increases because of the adoption of the STARS user interface, and the efficiency of maintenance is remarkably improved.

INSTALLATION OF THE NEW BEAM-LINE INTERLOCK SYSTEM

BL-4 is an X-ray beamline that used the legacy beamline interlock system previously. We replaced it with the new beamline interlock system.

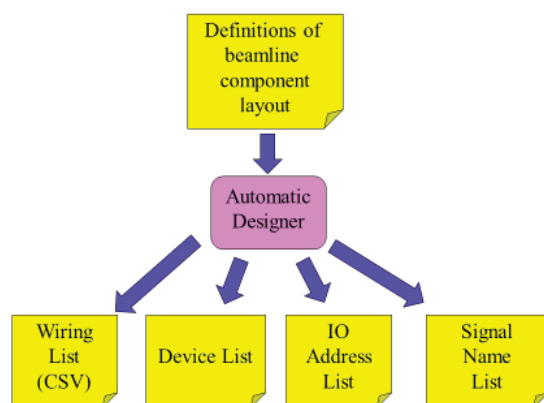


Figure 5: Automatic design.

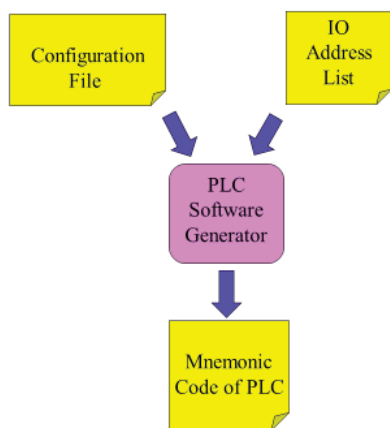


Figure 6: Generating PLC software automatically.

Automatic Designer

We are currently developing an automatic design tool for the beamline interlock system. First, we define the layout of the beamline components such as sensors, vacuum gauges, gate valves, and beam shutters. The automatic

design tool exports the wiring list, device list, IO address list, and signal list for the PLC interface with the definition (Figure 5).

PLC Software Generator

Recently, the PLC software of the beamline interlock system is generated automatically at the Photon Factory. We made a few expansions to support the PC-based user interface.

First, the PLC software generator reads the configuration file and IO address list made by the automatic designer; then, it generates the mnemonic code of the PLC (Figure 6).

Graphical User Interface

The touch panels were replaced with a graphical user interface (GUI) of STARS. Figure 7 shows the GUI panel of the new beamline interlock system.

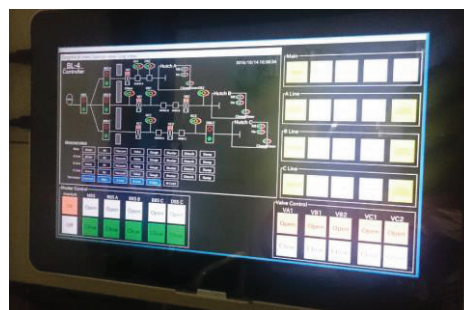


Figure 7: GUI panel of new beamline interlock system.

This GUI uses a .NET framework and the STARS.NET interface library. We are able to use the same tools and skills as those of the beamline control system. Moreover, the use of a desktop computer is more cost effective compared with the special touch panel device.

CONCLUSION

We succeeded in the development and installation of a new beamline interlock system with the STARS GUI. We then verified if the system effectively reduced required labor and cost. We are continuing the replacement of the beamline interlock systems.

At present, a new facility, tentatively termed the KEK light source (KEK-LS) [3], an extremely low-emittance 3 GeV storage ring, has been proposed and the conceptual design report preparation is in the final phase. Our new beamline interlock system is also applicable to the KEK-LS.

REFERENCES

- [1] STARS, <http://stars.kek.jp>
- [2] T. Kosuge and Y. Nagatani, "STARS: Current Development Status," in *Proc. PCaPAC'14*, Karlsruhe, Germany, Oct. 2014, paper WPO019.
- [3] T. Honda *et al.*, "Present Status of KEK Photon Factory and Future Project," in *Proc. IPAC'16*, Busan, Korea, May 2016, paper WEPOW020.

PERSONAL SAFETY SYSTEM IN SESAME

A. Abbadi, A. Hamad, I. Saleh, SESAME, Allan, Jordan

Abstract

SESAME (Synchrotron-light for Experimental Science and Applications in the Middle East) is a third generation synchrotron light source under construction in Allan, Jordan. The personal safety system PSS aims to protect the personnel from radiation hazards coming from accelerator's operation by controlling the access to the radiation area and to interlock the operation of accelerator's sub-systems according to the area status. Phase 1 of SESAME PSS has been installed and commissioned successfully for Microtron and Booster tunnel. Rockwell L72s safety PLC (up to SIL 3 applications) [1] has been used for Microtron, Booster and Storage Ring PSS. Phase 2 includes the Installation of the storage ring's PSS; two new PSS cabinets with remote safety IO modules have been installed for the storage ring PSS and connected to the main PSS PLC via Ethernet safety CIP. Phase 3 is the personal Safety System for SESAME day one beam-lines which is currently under construction. Many procedures and interlocks have been implemented in order to allow SESAME Booster, Storage Ring and Beam-lines personal safety to be managed in a systematic, risk based manner.

PLC BASED PERSONAL SAFETY SYSTEM

In 1998, the first part of a seven-part international standard was published to define the requirements for programmable electronic systems used in the safety related parts of controls systems. This standard is known as IEC 61508, "Functional safety of electrical/electronic/programmable electronic safety-related systems". This seven part standard is driving the direction for future safety PLC developments [1].

The personal safety system of SESAME is a PLC based control system which provides an easy tool to implement the PSS sequences and procedures in a protected programming tool where all changes are recorded. PLC provide the benefit to control the remote safety Input and Output modules through a safety certified communication bus; this reduces cabling and reduces the risk of cabling errors.

Safety PLC has redundant microprocessors, Flash and RAM that are continuously monitored by a watchdog circuit and a synchronous detection circuit, safety PLCs have an internal 'output' circuit associated with each input for the purpose of 'exercising' the input circuitry.

Inputs are driven both high and low for very short cycles during runtime to verify their functionality. Safety PLCs are suited for applications at SIL 2 and SIL 3 where they can be certified for use in most common safety applications so it may prove more cost effective to use the certified package versus taking a new control architecture through the certification process [1].

Figure 1 shows the safety PLC properties in the programming software. The controller can be Locked/Unlocked by a password, no modification is allowed on the safety logic if the controller is locked or working in running mode [1]. Safety signature should be generated after each logic modification; this signature shows the date and time of modification.

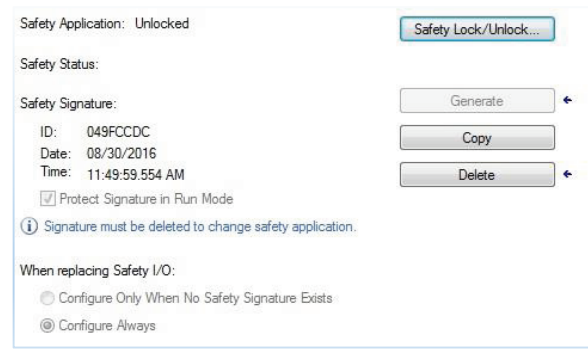


Figure 1: PSS safety PLC (Allen Bradley Guard -Logix L72s) properties.

SIL (SAFETY INTEGRITY LEVEL)

In comparison to similar facilities, the personal safety system at SESAME has been designed and selected to meet the requirements of SIL3, the SIL rating of a safety function is a measure of the degree of confidence in its overall ability to provide the safety function for nearly all of the time and under nearly all circumstances [2]. The most widely adopted functional safety standard is the International Electrotechnical Commission's IEC 61508 [3]. The standard divides SIL ratings into four levels, one through four, each representing an order of magnitude reduction in risk.

SYSTEM LAYOUT

Figure 2 shows the layout of personal safety system PLCs where the safety related functions are performed. A separated PLC for machine PSS and another one for the beam-line PSS.

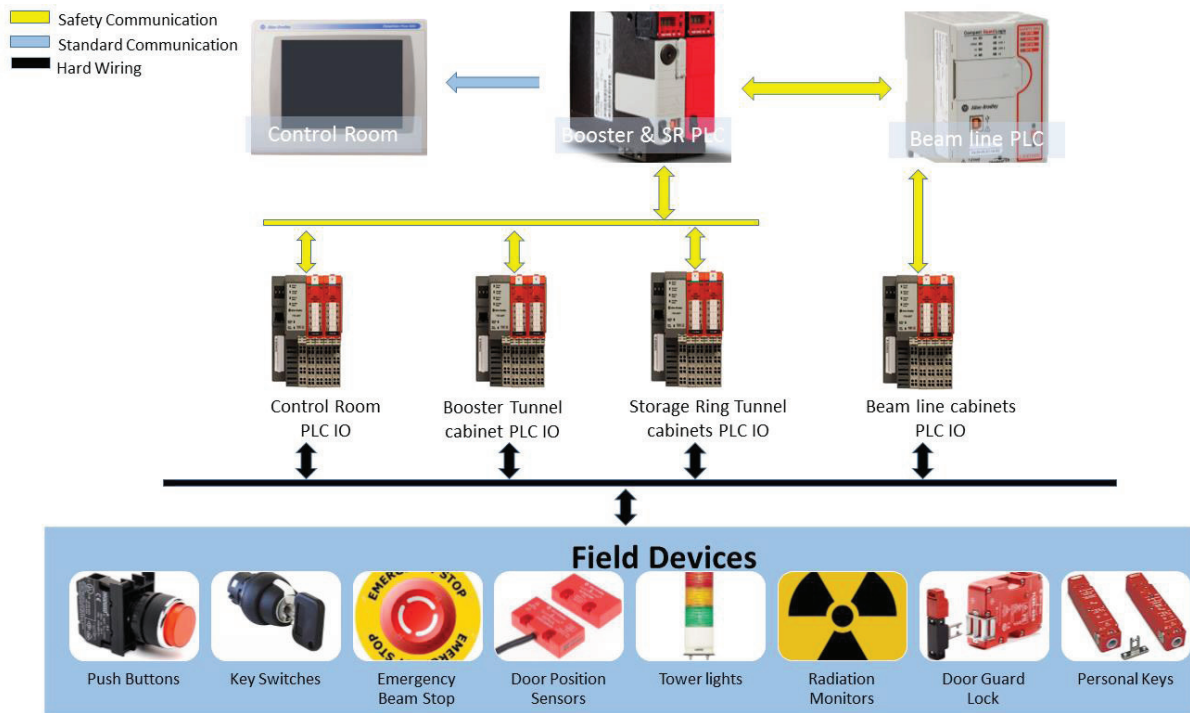


Figure 2: Layout of SESAME's personal safety system.

PSS FUNCTIONS

The PSS functions have been implemented by a combination of software routines and a set of devices dedicated for PSS. The interlocked areas are divided into three parts: Booster tunnel, Storage Ring tunnel and Beam-lines each part has its own emergency stop buttons, search points buttons, horns, flashing lights, door locks, door position sensors and status panel, Fig. 3 shows the personal safety system components in the storage ring area.

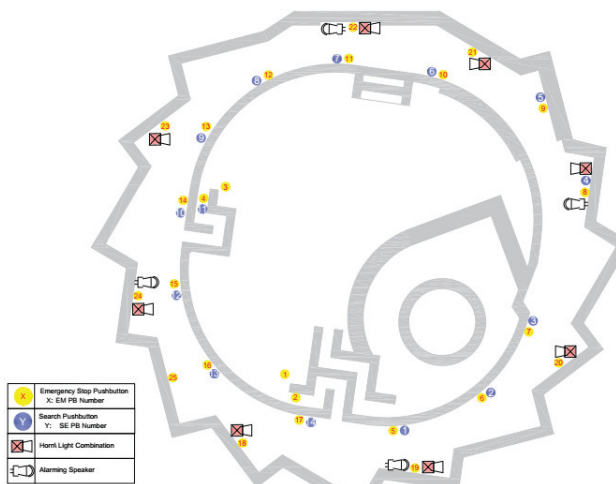


Figure 3: Personal Safety System components locations in the Storage Ring Tunnel.

The personal safety system has many functions:

- It's provide a search (Sweep) and personnel evacuation procedure from radiation area prior to accelerator operation by three trained people, the

PSS PLC monitor, controls the search sequence and visualizes the steps on HMI at control room for operator monitoring, a set of search buttons inside the radiation area should be pressed by a predefined sequence and pre-defined time range and the sequence.

- Audible and visual warnings during tunnel search and prior to machine operation, a set of loudspeakers announce a recorded message for tunnel evacuation and another message prior to operation in addition to alarming flashing lights with horn.
- Visual indication of machine status at each door of radiation area and at control room.
- Redundant and diverse locking mechanisms and position sensor for area entrances to protect the people from entering the radiation area during operation.
- Shut off the machine when a predefined accumulated dose values detected in occupied area by radiation monitors. Combined neutron-gamma radiation trolleys (by Thermo Fisher) are distributed in the occupied critical zones. Three digital signals are continuously provided by each trolley to PSS, monitoring the status of radiation level and error signals. PSS will react accordingly to each of these signals [4].
- Provide a set of emergency stops distributed in the radiation area, control room and next to doors in the service area which can be used by personnel to shut off the machine on emergency case.

LOGIC

The safety PLC monitors a set of conditions connected to its input modules and as result of the programmed logic safety PLC will send the permissions to allow the operation of the interlocked system as Microtron, Booster RF, Storage Ring RF, Injection from booster to Storage Ring and beam shutters, each system has some conditions to be verified, if one or more conditions lost, the PLC will disable the related system.

The Personal Safety System logic has been programmed to work in a three states, the first one is open state which is the mode prior to area search procedure, the interlocked area is free to access, the doors are unlocked, and the beam is not permitted. The second is interlocked; the mode after doing the area search procedure, no people inside the radiation area, no access is allowed to radiation area, doors are closed and locked and beam permits is active. The third state is restricted access; a controlled access to the booster tunnel and storage ring tunnel for a limited number of personnel is available by using a special personal keys located at the PSS panel next to tunnel door (key position is detected by PSS PLC), the beam permit is disabled during the restricted access. This mode ends if all personal keys are returned to their locations on PSS cabinet and the finish button pressed from the control room. The restricted access function is implemented only for the Accelerators PSS in order to make short interventions in the restricted areas without needing a new search patrol [5].

CONTROL ROOM

The control room PSS cabinet contains an HMI display, Fig. 4 shows the main GUI for storage ring PSS, the operator can monitor the PSS status, interlocks state, alarms history and the required conditions for each procedure.

CIP SAFETY PROTOCOL

The communication between PSS PLCs and remote IO modules is carried out through Ethernet CIP safety protocol, the CIP Safety end-to-end protocol gives responsibility to ensuring safety to the end nodes rather than the bridges, routers, or intermediate nodes. CIP Safety cannot prevent communication errors from occurring, but if an error does occur in the transmission of data or in the intermediate router, the end device detects the failure and takes the appropriate action. CIP Safety is certified to be compliant with the functional safety standard IEC 61508 up to safety integrity level SIL 3 [1].

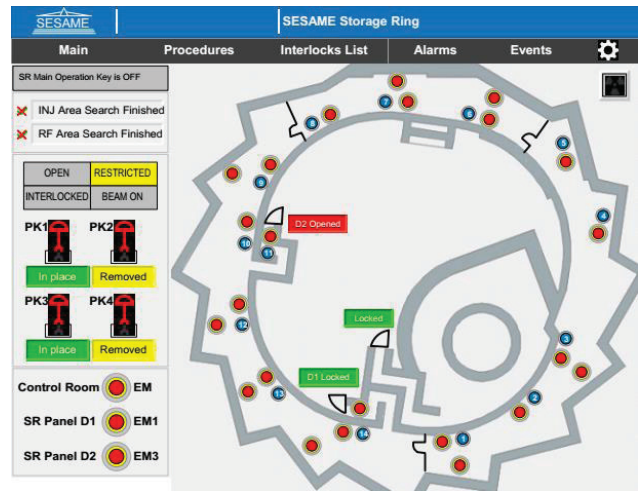


Figure 4: Personal Safety System GUI Main Screen for Storage Ring.

The safety PLC is a de-energize-to-trip system, which means that all of its outputs are set to zero when a fault in communication is detected which disables all the interlocked systems to the safe state [1].

This HMI, only meant for monitoring and diagnostics, reads tags from standard (not-safety) data blocks in the PLCs [5].

ACKNOWLEDGMENT

I would like to thank Dr. Erhard Huttel, Technical Director, the SESAME team, and special thanks to the control team.

REFERENCES

- [1] Rockwell Automation, <http://ab.rockwellautomation.com>
- [2] D. M. Macdonald, *Practical Machinery Safety*. IDC Technologies, Cape Town, South Africa: Elsevier, 2004.
- [3] International Electrotechnical Commission, "Functional safety of electrical/electronic/programmable electronic safety related systems", IEC 61508, International Electrotechnical Commission, Geneva, 2000.
- [4] Thermo Fisher, <http://www.thermofisher.com>
- [5] D. Fernández-Carreiras, "Alba, the PLC based protection systems", in *Proc. ICALEPCS'09*, Kobe, Japan, Oct. 2009, paper WEC003, pp. 397-399.

RECENT IMPROVEMENTS TO THE RIKEN RI BEAM FACTORY CONTROL SYSTEM

M. Komiyama[†], N. Fukunishi, A. Uchiyama, RIKEN Nishina Center, Wako, Saitama, Japan
M. Hamanaka, T. Nakamura, SHI Accelerator Service, Ltd., Shinagawa, Tokyo, Japan

Abstract

The RIKEN Radioactive Isotope Beam Factory (RIBF) is a cyclotron-based heavy-ion accelerator facility that provides the world's most intense beams of unstable nuclei for nuclear physics studies. A major part of the components of the RIBF accelerator complex is controlled by the Experimental Physics and Industrial Control System (EPICS). Here, we present recent improvements to the EPICS-based RIBF control system. First, the alarm system was improved to support stable beam delivery during a long-term experiment. We introduced the Best Ever Alarm System Toolkit (BEAST) based on the Control System Studio (CSS) platform to our control system and started to monitor the vacuum systems and magnet power supplies in order to detect hardware issues or anomalous behaviors of the accelerator components. Second, the data-logging and setting software system was renewed for nearly 900 magnet power supply units. These power supplies are controlled by several different types of controllers and the configuration of the magnet power supplies has become very complicated because of several recently performed extensions and updates of the RIBF accelerator complex. Hence, we have developed new control programs in order to simplify the recording and setting of the data relevant to the operation of all the magnet power supplies.

INTRODUCTION

The RIKEN Radioactive Isotope Beam Factory (RIBF) is a cyclotron-based accelerator facility for nuclear science investigations. It consists of two heavy-ion linac injectors and five heavy-ion cyclotrons, one of which is the world's first superconducting ring cyclotron (SRC). Several acceleration modes can be achieved by changing the combination of accelerators used, and one of them is chosen according to the beam condition required by users [1]. A major part of the components of the RIBF accelerator complex, such as magnet power supplies, beam diagnostic devices, and vacuum systems is controlled by the Experimental Physics and Industrial Control System (EPICS) [2]. On the other hand, the radio frequency (RF) systems used in the RIBF accelerator complex are controlled by different dedicated systems [3]. However, all the essential operation data sets of the EPICS and other control systems are integrated into the EPICS-based control system. In addition, two types of interlock systems, which are independent of the control system, are constructed in the RIBF facility. One is a radiation safety interlock system for human protection [4], and the other is a beam interlock system (BIS) that

[†] misaki@riken.jp

protects the hardware of the RIBF accelerator complex from unacceptable beam losses caused by the misoperation of high-power heavy-ion beams [5]. Figure 1 shows an overview of the RIBF control system.

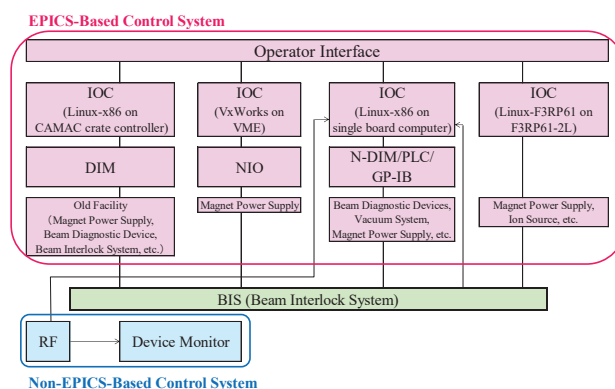


Figure 1: Overview of the RIBF control system.

IMPROVEMENT OF THE ALARM SYSTEM

At the RIBF accelerator facility, various types of heavy-ions are accelerated for various types of experiments. The most operationally difficult example is a 345-MeV/nucleon ^{238}U in which we use one injector linac and four ring cyclotrons in a cascade. Operators of the RIBF accelerator are required to control a large number of accelerators components safely during beam tuning and to maintain their fine tuning during experiments lasting for long periods (typically one month). In this case, the operators should adjust and monitor 600 magnet power supply units and monitor the status of 60 units used for systems such as vacuum pumps and gate valves of the accelerators and the beam transport lines. In addition, when a trouble occurs during an experiment, the operator is required to determine the cause of the trouble and, if possible, remove it immediately. Therefore, disorders of the components, which may cause an accident, should be detected as soon as possible, especially for high-intensity operations of the RIBF accelerator complex performed routinely owing to recent performance updates. However, the existing alarm system, the Alarm Handler of the EPICS only covers some components of the ion source. Hence, we have started to upgrade the existing alarm system.

We newly installed a distributed alarm system, the Best Ever Alarm System Toolkit (BEAST) [6] under the Control System Studio (CSS) [7] that is an Eclipse-based collection of tools to monitor and operate large scale control systems. In the RIBF control system, we have

already used another CSS package, the Best OPI, Yet (BOY) [8] as a display manager in addition to the Motif Editor and Display Manager (MEDM) [9] and the Extensible Display Manager (EDM) [10] by taking merits of each system into account. The BEAST was chosen taking future extension of the CSS at the RIBF control system instead of upgrading the existing Alarm Handler [11].

As the first step of the alarm system upgrade, the vacuum status of the entire RIBF accelerator complex, including the status of vacuum pumps, gate valves, and vacuum pressures, is registered in the BEAST. The BEAST outputs a warning signal when the opening-closing status of a valve changes, a vacuum pump stops, or a sizable change of vacuum pressure is detected. The criterion for a sizeable change in the vacuum pressure is a difference of more than twice the average vacuum pressure over the past 10 readings. The vacuum signals have been monitored by an EPICS Input Output Controller (IOC) at an interval ranging from 0.1 s to 1 s depending on their location. In addition, the difference between the set and the read-back value of the excitation currents of all the magnet power supplies of the RIBF accelerator complex is registered in the BEAST. The BEAST outputs a warning signal when the ratio of the read-back current values to the set value is different by 50% (minor alarm) and 90% (major alarm). The accuracy of the read-back value is insufficient for especially old magnet power supplies, and the small difference between them is not effective for a warning signal. The current value of the magnet power supplies is monitored by EPICS IOC at 10-s intervals.

We registered approximately 700 signals in the BEAST and its test operation has been started in April 2016. We were concerned that we registered too many signals before starting its operation; however, the system runs correctly without showing any delayed responses due to overload. Toward more efficient operation, we are now considering how to set the optimal alarm criterion patterns for each experiment in the BEAST. Since there are 65 beam operation patterns with different combinations of the ion sources, accelerators, and experimental vaults, it is necessary to activate signals relevant to running experiments.

RENEWAL OF DATA-LOGGING AND SETTING SOFTWARE SYSTEM FOR MAGNET POWER SUPPLY

In an experiment, several hundreds of magnet power supplies are used. In order to control such a number of magnet power supplies smoothly and avoid missing any, the operators usually use batch control programs when they control them as a group, such as at the start of beam tuning, the end of beam tuning or the end of an experiment. At the beginning of beam tuning, a batch control program of the system sets values of excitation currents to all the magnet power supplies used for the experiment, instead of one by one from the GUI for each

magnet power supply. Furthermore, the current values used at the experiment are recorded by another batch control program of the system at any time. Since the number of the magnet power supplies and types of their controllers have increased with each update step of the RIBF accelerator complex, both kinds of control programs were developed for each type of controller of the magnet power supplies. According to the type of controller, they are summarized into five groups. The name of each controller and number of magnet power supplies to be controlled are as follows:

1. Network-I/O (NIO) [3], which is a commercially available control system manufactured by Hitachi Zosen Corporation. Approximately 470 units.
2. Device Interface Module (DIM) [12], which is an in-house controller based on the computer automated measurement and control (CAMAC) system developed in 1984. Approximately 350 units.
3. Modules of programmable logic controllers (PLCs) [13]. Approximately 30 units.
4. General-purpose interface bus (GP-IB) controllers. Approximately 20 units.
5. Network-DIM (N-DIM) [14], which is an in-house controller developed as a successor to DIM in 2004. Approximately 30 units.

The batch control programs are started from the GUI for a magnet power supply. When the GUI button for recording the values of excitation currents of the magnet power supplies is clicked, the batch programs for each type of controller start collecting the data of current values from the EPICS Process variables on the EPICS IOC and store them into the files dedicated to each type of controller and location of the magnet under a date directory, which is automatically created daily on the file server. On the other hand, when the GUI button for setting the current value to the magnet power supplies is clicked, the batch programs for each type of controller set the data with reference to the data files for each type of controller in a dedicated directory set by the operator. The data file is edited based on the logged data of a same type of experiment performed in the past, or a result of calculation. In recent years, according to the several updates of the magnet power supplies associated with the facility updates, the configuration of the magnet power supply in the stored data files has changed. To accommodate these changes, we developed patch programs to modify the log files as needed. However, the types of patch programs have increased gradually, and the setting procedure of magnet power supplies has become very complex. For example, when the controller of a magnet power supply is changed and we subsequently use a file stored before the controller change, it is necessary to remove the magnet power supply data from the setting file, and the magnet power supply data must be set separately and individually. Consequently, mistakes in operation, such as failure to set data or to record data, have become more likely to occur. Therefore, we renewed the management system of the magnet power supplies this year.

The new data-logging and setting software system is designed to manage the data of the magnet power supplies using a relational database instead of a text file, and we developed a web-based software system for managing the relational database. In the RIBF control system, we have been using the Oracle database for the management of the static parameters of the accelerator components, such as magnet power supplies, and the information of the beam operation patterns. However, to separate the management of the operational data and the accelerator parameters, we have newly introduced the PostgreSQL database dedicated for this purpose. In the PostgreSQL database, we have created two kinds of tables: one is a table for providing information on the RIBF accelerator complex, such as the list of beam operation patterns, magnets, and their arrangement order from the upstream in each beam operation pattern and the type of controller of each magnet power supply, and the other is a table to store the collected current data from the magnet power supply. The former tables are used as a reference at the time of recording and setting of the current value, and they are modified automatically by reference to the Oracle database to avoid dual management.

The algorithm for recording data of the new system is as follows:

1. The operator inputs the ID number of the beam operation pattern (hereafter, the ID) used at the running experiment and clicks the save button on the MEDM-based GUI for recording the current data into a database table.
2. The new batch control program starts to refer to the table of magnet list tagged by the ID and picks up the included magnet information, such as name and the controller type of its power supply, taking the set and read-back current data of the selected magnet power supplies from the appropriate EPICS IOC. The new program is designed to take the data from all the controller types.
3. The program writes the data into a table created with a tag of date and time, the type of beam, and the ID. The table is automatically created by the web-based system. The result of whether the save succeeded is shown on the MEDM-based GUI every time.

On the other hand, the algorithm for setting the current data to each magnet power supply by the group is as follows:

1. The operator edits the current data into a setting format by using the web-based system shown in Fig. 2.
2. The operator starts the new batch control program for setting through the MEDM-based GUI.
3. The program checks the on-off status and the polarity setting of the magnet power supplies in comparison with the configuration data. It turns on the magnet power supplies or changes the polarity setting as necessary, and sets each current value to each magnet power supply. The program is designed to set the data for all the controller types.

At the first step, we often use the past data stored in the database in addition to setting the calculation results. In order to be able to call the past data, a huge amount of data saved in the text files in the file server were imported into PostgreSQL.

The most significant advantage of the new system is that the batch control programs for data-logging and data-setting are integrated, and we can check the current setting status of the magnet power supplies in the list from the upstream of the beam course, regardless of the type of controller. Furthermore, even if the controller is replaced with the different existing types of controllers in the future, it is able to use the data and the program without having to edit the configuration of the data recorded in the past. Progress and results of the current setting are monitored on the web-based system at present. Thus far, since the setting result was only displayed in character on the window of the terminal running the program, the new system is advantageous even in this respect.

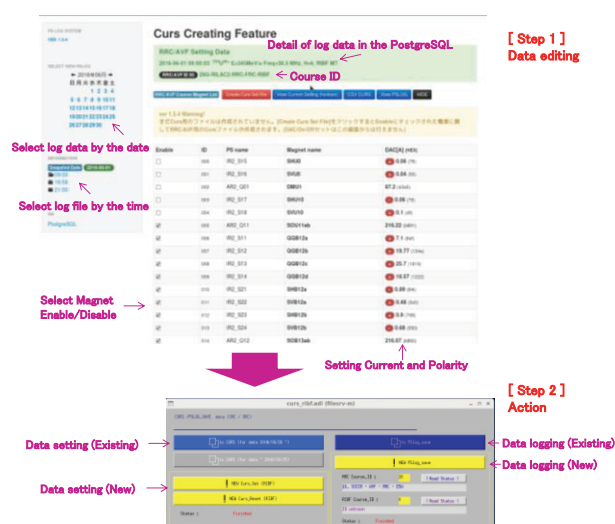


Figure 2: Web-based system for editing the values of excitation currents of all the magnet power supplies used for an experiment and MEDM-based GUI for setting.

REFERENCES

- [1] O. Kamigaito *et al.*, in *Proc. IPAC'16*, pp. 1281-1283.
- [2] EPICS, <http://www.aps.anl.gov/epics>
- [3] M. Komiyama *et al.*, in *Proc. HIAT'15*, pp. 104-106.
- [4] H. Sakamoto *et al.*, "HIS: The radiation safety interlock system for the RIKEN RI Beam Factory", *RIKEN Accel. Prog. Rep.*, vol. 37, pp. 281, 2004.
- [5] M. Komiyama, M. Fujimaki, I. Yokoyama, and M. Kase, "Status of control and beam interlock system for RARF and RIBF", *RIKEN Accel. Prog. Rep.*, vol. 39, pp. 239, 2006.
- [6] BEAST, <https://github.com/ControlSystemStudio/cs-studio/wiki/BEAST>
- [7] CSS, <http://controlsystemstudio.org>

- [8] CSS BOY, <https://github.com/ControlSystemStudio/cs-studio/wiki/BOY>
- [9] MEDM, <http://www.aps.anl.gov/epics/extensions/medm>
- [10] EDM, <https://ics-web.sns.ornl.gov/edm>
- [11] ALH, <http://www.aps.anl.gov/epics/extensions/alh>
- [12] K. Shimizu, T. Wada, J. Fujita, and I. Yokoyama, in *Proc. Cyclotrons '84*, pp. 392-395.
- [13] YOKOGAWA Linux CPU, <http://www.yokogawa.co.jp/rtos/Products/CPU/rtos-prdcpu2-ja.htm>
- [14] M. Fujimaki, M. Komiyama, I. Yokoyama, and M. Kase, "N-DIM as controller of devices for beam transport", *RIKEN Accel. Prog. Rep.*, vol. 37, pp. 279, 2004.

INTEGRATION OF STANDALONE CONTROL SYSTEMS INTO EPICS-BASED SYSTEM AT RIKEN RIBF

A. Uchiyama[†], M. Komiyama, M. Fujimaki, N. Fukunishi
RIKEN Nishina Center, Wako, Saitama, Japan

Abstract

The RIKEN RI Beam Factory (RIBF) was constructed as an extension of our old accelerator facility, the RIKEN Accelerator Research Facility (RARF). The major part of the accelerator components newly developed have been integrated into the Experimental Physics and Industrial Control System (EPICS), but several old standalone control systems were carried over and some new components adopted their own standalone control systems. These non-integrated systems are grouped into two major categories. The first is hard-wired control systems and the other is based on a two-layer remote-control system without a middle layer. From the view point of efficient accelerator operation, the entire control system should be integrated. For this reason, we have replaced hard-wired devices with EPICS-compatible devices, namely, the N-DIM (originally designed by the Nishina Center), and the FA-M3 (Yokogawa Electric Corporation, Tokyo, Japan) controllers. Additionally, to access data in a two-layer system from EPICS, we have introduced a MySQL-based system as the middle layer, and have developed a feature to connect the database through the CA protocol. Thus, we could successfully integrate the system, and it is now possible to acquire all the data through EPICS.

INTRODUCTION

The RIKEN RI Beam Factory (RIBF) accelerator facility consists of five cyclotrons, including a superconducting ring cyclotron, and two linear accelerators, which act as injectors [1]. For the RIBF, we constructed a distributed control system based on EPICS for almost of the accelerator components, including the magnet power supplies, beam diagnostic instruments, and vacuum control systems [2]. On the other hand, some of the accelerator components are controlled by non-EPICS-based small control systems because the RIBF project is also utilized some components of the RIKEN Accelerator Research Facility (RARF) [3], constructed as part of a previous project. In addition, two-layer control systems, which consist of controllers and client PCs, have also been adopted since the start of the RIBF project because they require only a relatively short development time.

From the viewpoint of accelerator operation, the control systems should be integrated because the operational efficiency of the accelerator depends on the interfaces of the control systems being unified. However, due to manpower and time constraints, it would be difficult to integrate all the non-EPICS-based system in a single step. In this study, we report the methods that could be adopted to introduce EPICS-compatible controllers, as well as the

data integration methods for non-EPICS-based small control systems.

STANDALONE SYSTEMS

Before the present study, the non-integrated systems remained are the control system for the 18-GHz electron cyclotron resonance ion source (18-GHz ECRIS) [4], that for the hyper electron cyclotron resonance ion source (Hyper ECRIS) [5], the beam Faraday cup control in RILAC [6], the radio-frequency (RF) systems, the temperature-measuring system, as well as several two-layer systems. We examined a mean of integration into EPICS suitable for each component one by one. Cross-operations are required for some of these standalone control systems because of management requirements.

INTEGRATION METHODS

18-GHz ECRIS

Hard-wired controllers had been used for the 18-GHz ECRIS control system since its commissioning in 1995. These old controllers should be modernized as soon as possible because the 18-GHz ECRIS is used very actively for both RIBF and standalone experiments in RILAC, the latter including superheavy element search experiments. We decided to use a Yokogawa F3RP61-2L programmable controller, in which EPICS is embedded [7], because this configuration has already been implemented in the RIBF control system. The control system for the 18-GHz ECRIS was completed in 2015 [8].

Beam Faraday Cup in RILAC

In the same way as in the case of the 18-GHz ECRIS control system, the RILAC beam Faraday cup control was based on hard-wired devices. We replaced the hard-wired devices with a N-DIM [9], and an EPICS input/output controller (IOC), which consists of single-board computers running Linux x86 [10] to provide a channel access (CA) service for the N-DIMs.

Hyper ECRIS

Hyper ECRIS, developed and operated by the Center for Nuclear Study, University of Tokyo (CNS), is used to provide a variety of metal ion beams for injection in the RIBF [6]. The control system was constructed as a standalone system with a closed network, for which the main controller was a MELSEC-A series programmable logic controller (PLC). Hyper ECRIS is usually controlled by using a Microsoft Windows-based client PC, located in the ion source room. Therefore, this client PC in the ion source room had to be accessed remotely by the accelerator operators from the RIBF control room via remote

[†] a-uchi@riken.jp

desktop protocol whenever beam tuning of Hyper ECRIS was required.

On the other hand, an unintegrated control system between Hyper ECRIS and the RIBF accelerator causes inefficient operation. However, the control system could not be replaced with one based on EPICS for the benefit of the RIBF accelerator operation, because Hyper ECRIS is managed by CNS. For this reason, the conventional control method was left as is, and integration to the EPICS system was supplemented by introducing an EPICS IOC as a gateway between the RIBF control network and the closed network of the Hyper ECRIS control system (see Fig. 1). In the present study, we developed EPICS device support for a MELSEC-A series PLC by using NetDev [11], developed by KEK. Thus, we could construct unified operator interfaces by utilizing CSS/BOY [12].

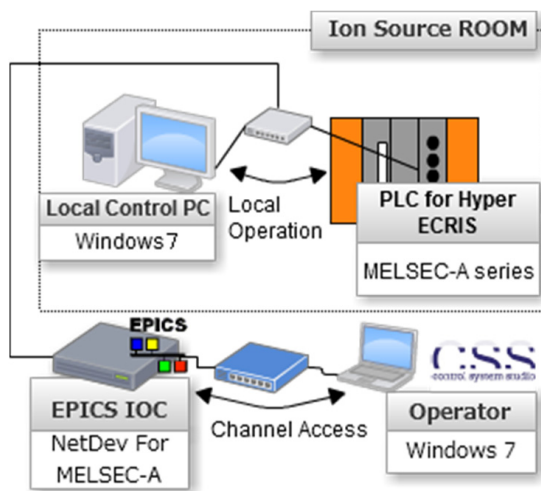


Figure 1: Hyper ECRIS control system. Both the conventional method and EPICS are used.

RF Controls

For the linear accelerator and the cyclotrons, the RF control systems are implemented as a two-layer system consisting of a client, based on Wonderware InTouch [13], and OMRON programmable logic controllers (PLC). Since the accelerator operators are familiar with the InTouch interface, it would not be beneficial to replace the GUIs to send output control commands to the PLCs. On the other hand, there is a need to monitor the RF data along with the other EPICS-based data, such as vacuum pressures, by creating charts and graphs. Therefore, the two-layer systems were left as they were for the accelerator operation, but we inserted a middle layer, based on EPICS, to monitor the data. In this case, the EPICS device support software is realized by the features provided by NetDev [11].

Other Two-layer Systems

When a new instrument is introduced, the control system may not be compatible with EPICS because human resources are required to develop the EPICS device sup-

port software. National Instrument LabVIEW is a suitable platform for the rapid prototyping of a system. In RIBF, LabVIEW-based systems with a two-layer structure are utilized for the gas stripper control [14], the monitoring of the beam phase and intensity [15], measurement of beam energy system [16], and so on. In addition, commercial systems are also implemented to monitor the temperature. To attain data integration, we considered an approach for handling and sharing the data for the system described above.

MYSQL-BASED DATABASE AS MIDDLE LAYER

Since there is a need to store and analyze the data for a system with a two-layer structure, we introduced MyDAQ2, developed by SPring-8, as a common DAQ system for small non-EPICS-based systems [17]. The MyDAQ2 system can store data into a MySQL-based database by sending an ASCII command with the socket and allow viewing of the stored data via a Web application [18]. One of the good features of MyDAQ2 is that it is possible to develop easily a program to store data from other client systems. In addition, by utilizing a JavaScript chart library, we have realized a chart feature with a much higher level of performance than the original MyDAQ2 chart feature which used gnuplot. Thus, the administrator can store data from the LabVIEW-based and commercial systems via MyDAQ2.

We constructed a prototype system capable of handling the data stored in a MySQL-based database by using EPICS CA. First, the system searches for a table name, and EPICS software records are created automatically per the table name. Then, the clients regularly connect to the MySQL-based database via a socket, after which the polling access program obtains the latest values and copies them to premade EPICS software records. For example, the "MyDAQ2:ABC:c1" software record is created in column *c1* in table *ABC*. Using this prototype system, it is possible to obtain the data stored in most of the non-EPICS-based systems via EPICS CA.

Given the success of this prototype, we implemented not only a simple system to copy a value to the EPICS software record, but also a method to handle the MyDAQ2 data through EPICS device support software. The architecture of MyDAQ2 is compatible with the Message And Database Oriented Control Architecture (MADOCA) [19]. As such, it sends the ASCII command with the socket connection to the server, and then receives a response. For this reason, we have developed device support software utilizing StreamDevice [20], because StreamDevice offers the advantage of data processing with ASCII through a socket connection. An example of a protocol file for the MyDAQ2 device support software is shown below.

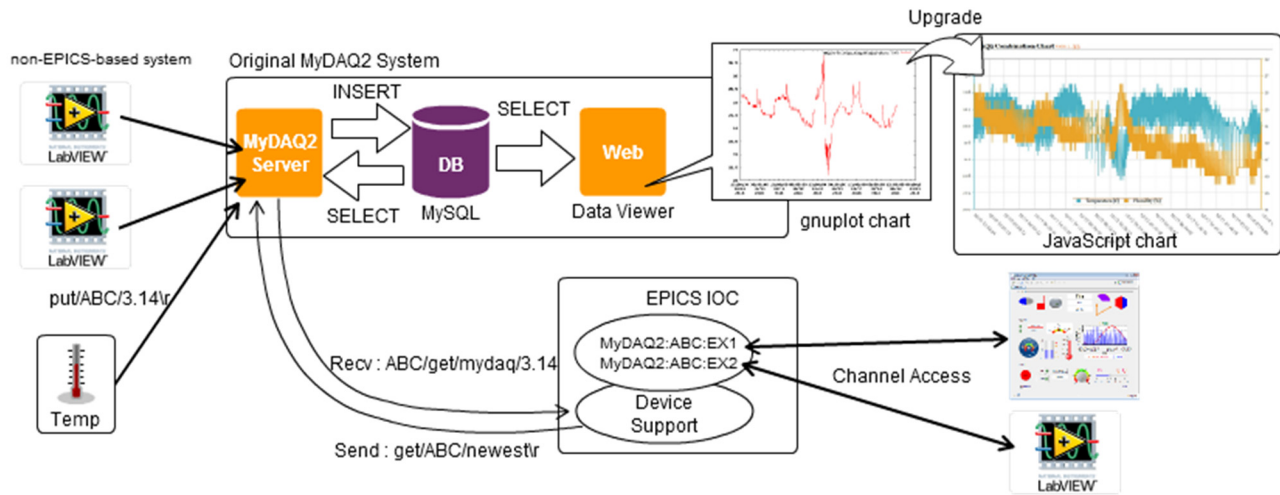


Figure 2: EPICS IOC for MyDAQ2. Newly stored data can be obtained from the MySQL-based database by using the device support software.

```
getMyDAQ2data{
    Separator = "_";
    out "get/\$1/newest";
    in "\$1/get/mydaq/%E";
}
```

The system is illustrated in Figure 2. Simply by inserting data into MyDAQ2, data in the CA protocol can be flexibly acquired, even if the system is a non-EPICS-based system.

SUMMARY

Basically, a RIBF control system is based on EPICS. However, non-EPICS-based systems, configured as standalone systems, have been utilized for some components. Using embedded EPICS technology, hard-wired control systems have been replaced and integrated into the EPICS-based system by using Yokogawa F3RP61-2L and FA-M3 modules.

By using NetDev, Hyper ECRIS and RF system can also be controlled with EPICS while maintaining a conventional two-layer structure. Additionally, we introduced a MyDAQ2 system to enable us sharing data used in LabVIEW-based systems, commercial monitoring systems, and so on. Since all the data for a non-EPICS-based system are stored into a MySQL-based database via MyDAQ2, the data can be accessed via the CA protocol by developing EPICS device support for MyDAQ2.

In our study, the control system integration was successfully completed and we can now handle the data of almost all the RIBF components in an integrated way.

REFERENCES

- [1] O. Kamigaito *et al.*, *Proc. IPAC'14*, Dresden, Germany, 2014, p. 800.
- [2] M. Komiyama *et al.*, *Proc. ICALEPCS'13*, San Francisco, CA, USA, Oct. 2013, p. 348.
- [3] Y. Yano *et al.*, in *Proc. EPAC'94*, p. 488.

- [4] T. Nakagawa *et al.*, *Rev. Sci. Instrum.* 73, 513, 2002.
- [5] Y. Ohshiro *et al.*, *RIKEN Accel. Prog. Rep.* 36, 2003, p. 279.
- [6] M. Odera *et al.*, *Nucl. Instrum. Methods A* 227, 187, 1984.
- [7] A. Uchiyama *et al.*, *Proc. PCaPAC08*, Ljubljana, Slovenia, 2008, p. 145.
- [8] A. Uchiyama *et al.*, *Rev. Sci. Instrum.* 87, 02A722, 2016.
- [9] M. Fujimaki *et al.*, *RIKEN Accel. Prog. Rep.* 37, 2004, p. 279.
- [10] M. Komiyama *et al.*, presented at PCaPAC'16, Campinas, Brazil, paper WEOPRPO12, this conference.
- [11] J. Odagiri *et al.*, *Proc. ICALEPCS 2003*, Gyeongju, Korea (2003), p. 494.
- [12] M. Nishimura *et al.*, *Proc. PASJ16*, Chiba, Japan, August 2016. (To be published)
- [13] WonderwareInTouch, <https://www.wonderware.com/hmi-scada/intouch/>
- [14] R. Koyama *et al.*, *RIKEN Accel. Prog. Rep.* 48, 2015, p. 192.
- [15] R. Koyama *et al.*, *RIKEN Accel. Prog. Rep.* 40, 2007, p. 122.
- [16] T. Watanabe *et al.*, *Proc. PASJ8*, Tsukuba, Japan, August 2011, p. 421.
- [17] M. Komiyama *et al.*, *Proc. ICALEPCS 2011*, Grenoble, France, 2013, p. 90.
- [18] T. Hirono *et al.*, *Proc. PCaPAC08*, Ljubljana, Slovenia, 2008, p. 55.
- [19] T. Matsumoto *et al.*, *Proc. ICALEPCS 2015*, Melbourne, Australia, 2015, p. 954.
- [20] StreamDevice, <http://epics.web.psi.ch/software/streamdevice>

AUTOMATED AVAILABILITY STATISTICS

P. Duval, M. Lomperski, H. Ehrlichmann, DESY, Hamburg, Germany
J. Bobnar, Cosylab, Ljubljana, Slovenia

Abstract

The availability of a particle accelerator or any large machine with users is not only of paramount importance but is also, at the end of the day, an oft quoted number (0 to 100%) which represents (or is taken to represent) the overall health of the facility in question. When a single number can somehow reflect on the maintenance, operation, and engineering of the machine, it is important to know how this number was obtained. In almost all cases, the officially quoted availability is generated by hand by a machine coordinator, who peruses the operation statistics over the period in question. And when humans are involved in such a calculation there might be a latent tendency to avoid the stigma of low availability. So, not only might there be scepticism at 'impossibly high' availability, but comparing quoted availability from one machine with another might turn out to be virtually meaningless.

In this paper we present a method for calculating the machine availability automatically, based on the known (and archived) machine states and the known (and archived) alarm states of the machine. Ideally this is sufficient for a completely automatic determination of the availability. This requires, however, a perfect representation of all possible machine states and a perfect representation of all possible fatal alarms (those leading to down time). As achieving perfection is ever an ongoing affair, the ability for a human to 'post-correct' the automated statistics is also described.

INTRODUCTION

A particle accelerator facility has an operations schedule (potentially 24 hours/day 7 days/week) where the facility is obligated to supply users or experiments with beam. Any unanticipated deviation from this operations schedule is regarded as non-availability. Quite naturally, machine coordinators strive to present a perfect score of 100 % availability at the weekly operations meeting. Traditionally a machine coordinator will pore over machine data, spreadsheets, logbook entries, etc. to obtain the *official* availability of the facility over the period in question.

We are motivated to generate this availability number automatically for several reasons. First and foremost, we can remove the human element entirely if the official availability is generated entirely automatically. Secondly, we can free up a significant amount of time spent by the coordinator calculating such a number by hand. Finally, we can monitor the availability on-line during operations.

REQUIRED SERVICES

The necessary ingredients to device such a system for automatically calculating machine availability over a selected time range consist of three central services. There must be a machine state server which correctly defines all possible declared states in which a facility can be in at any time. There must be a central alarm server with a clear definition of what constitutes a fatal alarm. It should also be realized that the condition of a fatal alarm is inextricably bound to the machine state, as we shall see below. There must also be a central archive server which keeps a history of the state and fatal-alarm information.

Machine State Server

The possible states of an accelerator facility are defined by the machine coordinators and the facility will be in *some* state at any given time. Theoretically the choice could be as simple as *running* or *not running*, but is generally more complicated. For completeness, the TINE [1] state server also recognizes the state *unknown* if there is no proper declared state. Otherwise the state of a machine will be declared to the state server and the machine will be assumed to be in that state until another state declaration is made. The set of all possible machine states is completely configurable.

There exists the question as to what to do about *problems*. Either *problems* is a valid machine state which is likewise declared or *problems* is an attribute assigned to one of the other defined valid machine states. For instance: "this is a declared *user run* but we have *problems*". The TINE state server can handle either option, but we point out that the current configuration treats *problems* itself as a valid declared machine state. This implies that some service must determine that we cannot be in an operational state and officially declare the state *problems*.

Central Alarm Server

The principal ansatz concerning availability is that "if the machine is not available then there must be at least one fatal alarm in one of its subsystems." And if we are treating *problems* as a declared state then a corollary to this ansatz is that "if we are in the *problems* state then there must be at least one fatal alarm".

We perhaps begin to see a number of consistency checks unfolding before our eyes. If the state is *problems* and there are no fatal alarms then this is by definition wrong and needs to be investigated and fixed. Furthermore, if there is a fatal alarm then the state must

be *problems*. If this is not true, then this is likewise wrong.

Ensuring that the control system alarms reflect the true state of the machine is a painstaking procedure and is a task which generally falls on the machine coordinator to undertake and complete.

Central Archive Server and Bean Counting

Our goal is to be able to specify any particular time range and obtain both state and availability information. This is in itself actually easy to realize. As we are never interested in a time granularity smaller than a second or two we need only count the seconds spent in any one state and archive this number. And in a similar fashion, we must only count the seconds where an alarm subsystem has at least one fatal alarm and likewise archive this number. The difference in the archived values at the end points specified by the selected time range provides us with all we need to know. As the archived data represent nothing more than counts (the cumulative number of seconds in a state) we often refer to this as *bean-counting*, a moniker which effectively represents its inherent simplicity.

The Operation History Viewer shown below in Figure 1 and available in the TINE Studio suite [2] in fact makes use of such archived *bean* counts and allows the user to select any time range and examine the machine state and availability history.

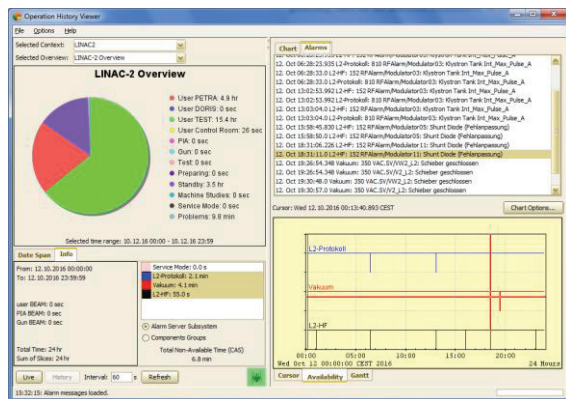


Figure 1: Operation History for the PETRA-3 linac showing data from Oct. 12, 2016.

In addition to the traditional pie-chart display of the total amount of time spent in each machine state, any subsystem of the facility which was not 100 % available over the selected time is noted and presented in a trend chart where periods of non-availability are easily recognized. The fatal alarms (the *blame*) at any given time are likewise easily viewed.

RESULTS

The simplicity of the above technique is alas muddled by the sheer complexity of ensuring the validity of the alarm information. For example, a fatal alarm appearing in the RF system even though there is a perfectly good

user run will destroy the availability calculation. And of course the state declaration must correspond with the true state of the machine. Any such inconsistencies will be spotted by the machine coordinator calculating the availability statistics and if they are religiously forwarded to those persons responsible for generating the alarms or declaring states, then eventually the availability results generated automatically by the above techniques will coincide with those calculated by hand by the machine coordinator.

Corrections

The trial-and-error period involved in ensuring that the automatically generated availability statistics are correct is expected to be long and drawn out. In fact any future upgrade to the accelerator facility is likely to bring new use cases and more trial and error with it.

During this trial-and-error process it is more important than ever to be able to *correct* the raw statistics displayed by the Operation History Viewer above. We do not correct the actual stored state data (i.e. the *bean* counts). Instead we provide a corrections database for both the machine states and the subsystem availability, which is then optionally applied to the statistics displayed in the application.

A machine coordinator can use the same application to correct known false information, for instance if the state change trigger declaring *problems* was somehow missed, etc. A right-click over the states history display (the pie chart in Figure 1) will offer the option to examine or apply corrections in the form of a new popup window, as shown in Figure 2.

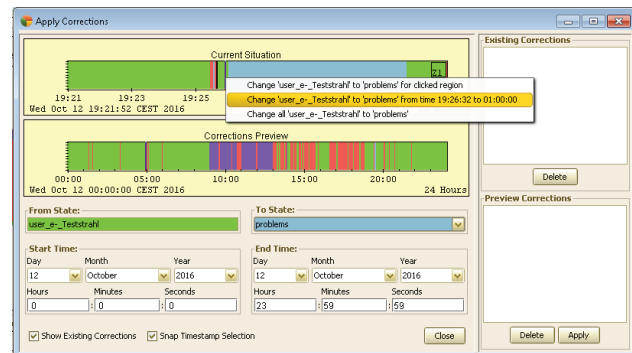


Figure 2: The state correction popup window.

Correcting a state to or from *problems* brings up the issue of correcting the availability. Since the *problems* state automatically implies that the machine was not available so long as it is in this state, then there is likely to be incorrect stored alarm information as well. Namely, we perhaps missed a fatal alarm somewhere (e.g. we know from the logbook that there was unscheduled downtime even though the declared state claimed we were in a user run) or perhaps we recorded a fatal alarm when there wasn't one (e.g. we had a happy user run even though the RF system claimed a fatal alarm). If on the other hand the stored alarm information is correct then the

declaration of *problems* was itself somewhere in error. In fact, a close scrutiny of Figure 1 will reveal that over the 24 hour period that was Oct. 12, 2016 the PETRA3 linac was in the *problems* state for 9.8 minutes whereas there were only 6.8 minutes of *non-availability*. This is clearly inconsistent and needs to be both corrected and investigated.

Apropos correcting availability, there is a second correction popup window designed to correct the availability counts and either assign blame to or remove blame from some subsystem, and this brings up two points. Point one is that the operation history application must itself check the time range of any *problems* state (with applied corrections) against that of the overall non-availability (with applied corrections) and warn the application user if these are not synchronized. Point two is that if the machine coordinator sees fit to apply any correction whatsoever he should inform the responsible parties that he needed to do so and request that steps be taken to fix the problem at the source. This latter turns out to be very important feedback for server programmers who might otherwise not have the overall picture of operations statistics in mind.

CONCLUSIONS

The ability to automatically display operation and availability statistics for a facility over any particular time interval and/or monitor the same on line is worthwhile and relatively easy to implement to first order. The devil as usual is in the details. The technique described here hinges on the proper identification of fatal alarms (defined as those leading to or responsible for non-scheduled downtime) and assigning them to the reason(s) for the non-availability. The operation history depends as well on the absolute correct declaration of the proper state of the facility. If these two points are met then the rest is simple *bean* counting and archiving.

We cannot understate how difficult it sometimes is to ensure that the identification of fatal alarms is in fact correct. This is often an iterative process spanning months if not years. Realizing this, we have added the ability to post-correct the raw data providing the automated statistics. Thus a machine coordinator can ensure that the displayed statistics for any time period is officially correct and at the same time do his part in iterating the system toward perfection.

We expect this to remain an ongoing project for some time. To be useful, this system absolutely requires an engaged machine coordinator who not only knows the systematics of machine operations but is willing to identify inconsistencies, both in the state declaration and in the setting of fatal alarms, and trace them back to their source. With the ability to post-correct the information displayed in the TINE Studio Operations History Viewer, it should not require much extra effort for a machine coordinator who is already calculating operations and availability statistics to see this through. To expedite the iterative improvement necessary a notification system will be added to the corrections dialog, so that the persons

responsible for alarms and state declarations can be informed of those inconsistencies which led to a correction.

If we are persistent in our efforts, then the automated availability calculation can not only be trusted but can be monitored on-line, for example at the beginning and end of a shift. Once the automatic calculation can be trusted, then we can regard the official availability as an *honest* assessment, as we have effectively removed any *human element* in the calculation which might subconsciously exaggerate or minimize downtime (and with the side-effect that the *human* involved is free to engage in other activities).

REFERENCES

- [1] TINE website, <http://tine.desy.de>
- [2] P. Duval, M. Lomperski, and J. Bobnar, "TINE Studio, Making Life Easy for Administrators, Operators and Developers", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, paper WEPGF133.

MULTIPURPOSE VACUUM CHAMBER - AUTOMATION, INTERLOCK-SYSTEM AND SELF-OPERATING VACUUM ROUTINES

R. M. Caliar^{*}, H. G. P. Oliveira, G. L. M. P. Rodrigues, C. H. Tho,
Brazilian Synchrotron Light Laboratory, Campinas, Brazil

Abstract

The LNLS' Beamlines Engineering Group is developing new technologies for SIRIUS Beamlines. To validate and test those technologies, the group has previously worked in a Multipurpose Vacuum Chamber (MPVC). This chamber has a powerful pumping system, several viewports, feedthroughs, RGA and a cryocooler system installed. This configuration delivery a very flexible test environment and relative short pump time to achieve UHV conditions, essential to reduce the test and validation time.

This paper will detail the MPVC automation structure, presenting the state machines, interlock logic and other diagrams that exemplify the automatic routines concepts. Operations like pumping from atmospheric pressure to HV, ionic pump flash and ventilation routines are automatic.

The automated system was developed in LabVIEW, using a cRIO and Touch Panel interface. Based in Ethernet connection and published shared variables, the system has a friendly user interface and archiving for the main variables.

The system was developed in a multi user collaborative ambient. The paper will show the advantages and disadvantages we faced working with LabVIEW as multiuser development tool.

- Cryogenic support system;
- Residual Gas Analyser;
- Complete automation operation;
- Safety system;
- Low electrical noise environment;
- IPS protection for main components.

Those characteristics were obtained as foreseen requirements for the future development projects, for the new Sirius beamlines, their prototypes and concept proves. The automated routines are the base for a pump station under development, which's mechanical sketch shall be similar to some published works [6].

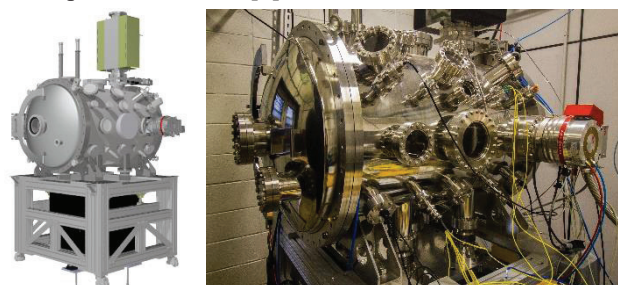


Figure 1: 3D model and actual picture of the Multipurpose Vacuum chamber, a flexible environment for the beamline engineering group.

INTRODUCTION

This work presents de project of a multipurpose vacuum chamber for the beamline engineering group at LNLS, dedicated for new beamlines projects for Sirius, the new Brazilian 4th generation synchrotron [1, 2]. Monochromators [3], KB Mirrors, Front Ends [4] and many others. The first experimental results were already presented on previous conferences [5].

SYSTEM PURPOSE AND DESIGN

The Beamline Engineering group started the mechanical design on early 2015, aiming one flexible environment, with some principal characteristics:

- Ultra-high vacuum (UHV) environment (target: 10-11 mbar with backing procedure and 10-8 mbar without baking);
- Low vibration environment;
- View ports in different angles, to make it flexible to implement optical instrumentation from outside the chamber;
- Low diffraction coefficient viewport from VAb Vakuum-Anlagengau, for interferometer characterization from outside;
- Many power, signal and temperature feedthroughs;

The detailed design was made, and the 3D model can be seen in Figure 1. The group experience conducted the project for a big UHV chamber (800 mm diameter and 1000 mm depth), and simulations were made to ensure a robust low vibration environment Figure 2.

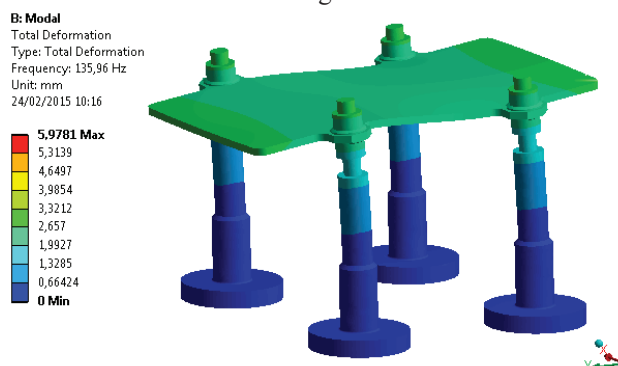


Figure 2: Modal Analyses to guarantee the low vibration environment (first vibration mode).

With the requirements and mechanical projects set, the final release, with hardware, software and interfaces solutions are presented on this text.

^{*} ricardo.caliari@lnls.br

HARDWARE

Vacuum Instrumentation

The selected vacuum pumps and instruments are:

- Mechanical pump: Pfeiffer ACP 40
- Turbomolecular pump: Pfeiffer HiPace 7000
- Ionic pump: Agilent 500 L/S-VACION PLUS 500
- Titanium sublimation pump: Agilent TSP
- Getter pump: SAES Getter NexTorr D2000
- Vacuum sensor: Vacom MVC3-BM
- RGA: Stanford Research RGA 200

With the selected system, the pressure level of 10^{-3} mbar can be reached in about 8 minutes (with mechanical pump), that should be enough for the cryogenics preliminary evaluations, for example. The final pressure of about 10^{-8} mbar can be reached in about 8 hours with turbomolecular pumping. The Ionic system can be used for low vibration operation in UHV.

Control Hardware

Hardware and software solutions from National Instruments were chosen for this system. The previous experience of many group members on LabVIEW programming and the hardware flexibility were essential for this choice.

The cRIO platform with FPGA was elected (Figure 3). The FPGA card was used to implement an interlock safety routine and an Ethernet based application run the end user interface. The I/O's and platform specifications are:

- NI 9148: Ethernet controller with integrated FPGA
- NI 9871: RS-485 serial interface 4 Ch module
- NI 9477: 60 V, Sinking Digital Output 32 Ch module
- NI 9425: 24 V, Sinking Digital Input 32 Ch module
- NI 9215: ± 10 V Analog Input, 16 bits, 4 Ch module
- NI 9217: RTD PT100, 4 Ch module

In order to make a robust interface between the platform and the instrumentation, two break-out-boxes were projected by the Electronics Support Group, in LNLS. One was installed inside the electronic rack and the other directly by the chamber. Robust cables and connectors were chosen to make the interface. Feedthroughs from MDC Vacuum and Accu-glass make the insertion of those signals into the UHV chamber.

AUTOMATION SYSTEM

Interlock and Safety Routines

To guarantee a safety operation, an interlock routine was implemented into the FPGA card presented on the main controller. This is the last hardware layer before the instruments I/O's. The really high speed processing (40 MHz clock) and real parallel characteristics of the FPGA ensure the safety operation, with really small event time response.

The FPGA programming was much easier using the LabVIEW user interface. An easy logic between the most important I/O's was defined to look for every input. This routine deals with problems like: safety button interface, energy power lost, pressure levels watch (to guarantee safety

operation of turbomolecular, ion, titanium and getter pumps).

Data Logging and Trends

The data logging is one of the most important features for the future tests. Using the pressure sensors, the analog input and the RS-485 interface, the system can save the pressure data with a resolution of some mili-seconds. Some pumping data can be seen in Figure 4.



Figure 3: Schematic representation of the interface hardware, using NI cRIO with FPGA card, NI LabVIEW in virtual machine, data bank in central Storage System and NI Touch Panel for final user interface.

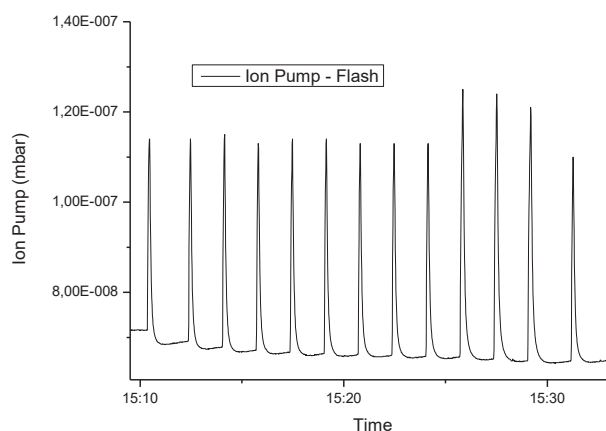


Figure 4: Ion pump pressure over time, using automated routine.

Project Structure

The program was implemented based on Ethernet communication, using the LabVIEW Shared Variables. The NI Distributed System Manager was an essential tool during debug phase, to verify the integrity of all variables. Figure 5 shows the project structure in LabVIEW Project.

The program works with different Sub-VI's running in parallel with the main program, at the host PC. These VI's coordinates the RS-485 communication with each controller. Suppliers have its own protocol and most of them were not available for LabVIEW. The implementation library was developed in this project and a stable version was updated on the NI Forum community.

The Hardware is used in a hybrid mode, where part of the chassis is running in FPGA mode and part using the Scan Interface. This solution was chosen to keep just the critical

I/O's running on the FPGA target and being processed at the safety routine. The lower speed I/O's (temperature sensors, for example) communicates directly with the host.

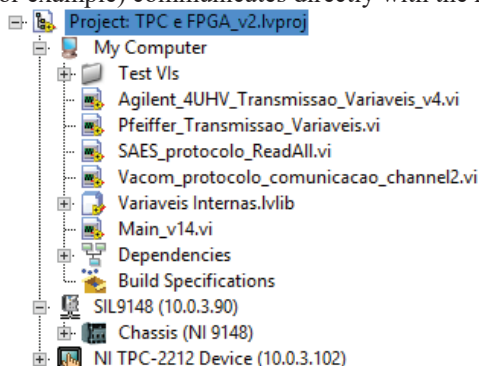


Figure 5: LabVIEW project structure, showing the VIs parallel running on the control computer, the control chassis, containing the FPGA interlock routine and the Touch Panel Controller, with the user interface.

Vacuum Automatic Routines

The main used vacuum pumping routines were automated. Those procedures were implemented to make the day life of the test engineers and technicians easier at the test shop. Another purpose of this implementation was the automation prototype for the future automatic pump stations for the Sirius beamlines. Those stations are already under development.

The automated routines are:

- Pumping: taking the system from atmospheric pressure to high vacuum level (10^{-8} mbar).
- Venting: take the system from any vacuum level to atmospheric pressure.
- Ionic pump flash: start the ionic pump operation, flash-ing it before the continuous operation.

Those routines were implemented in the high level application, using LabVIEW. They are always watched for the safety system and make the interface with the pump controllers.

One result can be seen in Figure 4, were the ion pump flash routine is shown. Figure 6 shows the user interface for the Touch Panel, installed at the control rack.

LESSONS LEARNED

The cooperative development using LabVIEW was really difficult to manage. About three people worked together on the software development. LabVIEW provided a really poor platform for parallel development. Therefore, the programmers should work in different sub-Vis and merge the code by their own.

The project operation relies in almost 100 shared variables, the use of the NI distributed system manager was essential at the debug phase. The non-initialization of these variables generated many issues and strange errors.

The National Instruments local support and Internet Forum community were once more fast and helpful to solve the doubts during the development phase.

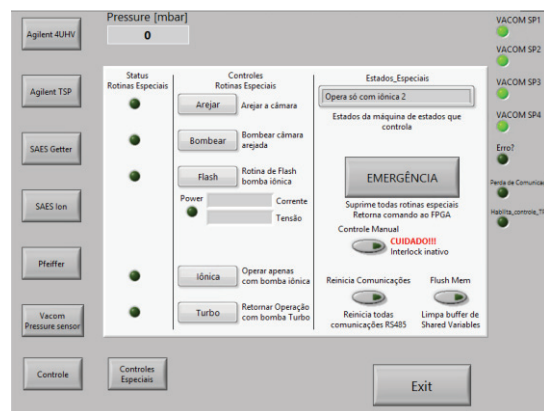


Figure 6: User interface, with the automated routines and special controls view.

RESULTS

Many results were already obtained using this test facility. A historic behaviour of the residual gas (example, Figure 7) into the chamber, over the performed tests, can indicate any kind of contamination and also perform specific vacuum compatibility characterization tests.

Formal outgassing analysis were made based on CERN procedures [7], the results were compared with NASA open databank [8] and the internal procedures validated.

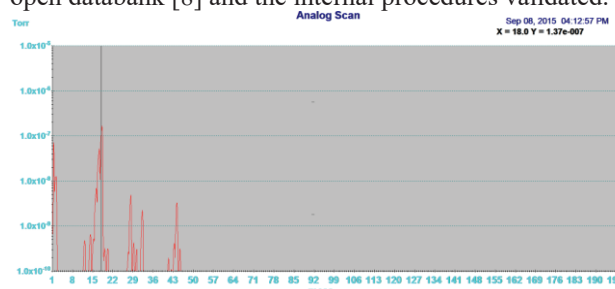


Figure 7: Analog scan using the RGA at a total pressure of about 10^{-7} mbar.

CONCLUSIONS

The system was successfully implemented and tests are being conducted on a stable environment. A beta version started the operation for vacuum experts on October 2015 and the first stable version was set at the really end 2015. Since then, about three big improvement maintenances were already conducted and the system could be really improved. The automated routines are working fine. The implemented logic is under revision and shall be the base for the automated pump station for Sirius Beamlines.

Over the past months, many tests were conducted and a historical database is now populated. Those data provide a much better comparison tool for future tests and an incremental quantitative evaluation for subsystems, like the first high-dynamics DMC prototype.

ACKNOWLEDGEMENT

The authors would like to gratefully acknowledge the funding by the Brazilian Ministry of Science, Technology, Innovation and Communication.

REFERENCES

- [1] A. Rodrigues *et al.*, “Sirius Status Report”, in *Proc. IPAC’16*, Busan, Korea, May 2016, paper WEPOW001, pp. 2811-2814.
- [2] L. Liu, F. H. de Sá, and X. R. Resende, “A new optics for Sirius,” in *Proc. IPAC’2016*, pp. 3413–3415.
- [3] R. Gerales *et al.*, “The new high dynamics DCM for Sirius”, in *Proc. MEDSI 2016*, Barcelona, Spain, Sep. 2016, paper TUCA05.
- [4] L. M. Volpe *et al.*, “High heat load front ends for Sirius”, in *Proc. MEDSI 2016*, Barcelona, Spain, Sep. 2016, paper WEPE06.
- [5] R. Caliri *et al.*, “Studies on flow-induced vibrations for the new high dynamics DCM for Sirius”, in *Proc. MEDSI 2016*, Barcelona, Spain, Sep. 2016, paper MOPE02.
- [6] M. Degenhardt, “Mobile dry pumping stations for PETRA III beamlines”, *Journal of Physics: Conference Series*, IOP Publishing Ltd, Sep. 2008.
- [7] P. Chiggiato, “Outgassing”, presented at C.A.S. 2006 Globe of Innovation, Platja d’Aro, Spain, May 2006.
- [8] Outgassing Data for Selecting Spacecraft Materials System, <https://outgassing.nasa.gov>

DEVELOPMENT OF A VIRTUAL ACCELERATOR FOR SIRIUS

X. R. Resende*, A. H. C. Mukai, L. N. P. Vilela, I. Stevani
Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Abstract

A virtual accelerator is being developed for Sirius, the new 4th generation synchrotron light source being built in Campinas, Brazil [1]. The virtual accelerator is an on-line beam simulator which is integrated into EPICS control system. It consists of a command line interface server with a channel access (CA) layer and with an in-house developed tracking code library written in C++ for efficiency gain. The purpose of such server is to facilitate early development and testing of high level applications for the control system.

INTRODUCTION

Sirius, the new storage ring at the Brazilian Synchrotron Light Laboratory (LNLS), will use EPICS as its control system. Most of the development of the high level applications (HLAs) will take place next year and will be conducted by the accelerator physics group. In the meanwhile, a few client applications have already been implemented to allow analysis of the choices in software development frameworks [2] that were made.

To be able to test and integrate the above-mentioned HLAs in the control system (CS), it was decided that a virtual accelerator (VA) with channel access server layer (CAS) for EPICS should be developed, implemented and made available as soon as possible. The idea is that having a VA allows for early development of control system software, be it for input-output controllers (IOCs) or HLAs.

The VA implements functionalities that can provide simulated process variables (PVs) on the CS that have not been made available yet, thus creating a mock-up CS environment in which early software development is possible. This test environment with VA also has the potential to speed up project development by partially parallelising implementations of applications that consume data from each other.

Commissioning training is also possible using HLAs in the control system provided with the virtual accelerator.

VIRTUAL ACCELERATOR

From the onset it was decided that the virtual accelerator would be composed of two parts: the first, a back-end machine application implementing a simulated virtual accelerator with a channel access server layer (VACA) and the second, a set of front-end virtual IOCs (vIOCS) with which other CS applications are supposed to interact.

Accelerator properties such as beam current and position, injection losses, power supply and RF subsystem setpoints, and so on, are simulated nominally in VACA. The virtual IOCS, on the other hand, encapsulate all PVs that represent the interface between the VA and the rest of the CS. They

also add simulated fluctuations to accelerator properties and implement device-dependent parameters, such as excitation curves of the magnets or BPM calibration parameters.

The advantage of this approach is that virtual IOCs can be gradually replaced by their corresponding real IOCs when they become available, and without having to modify core simulation code, since it is implemented separately in VACA.

```

File Edit View Search Terminal Help
ximenres@lnls208-linux:~$ sirius-vaca.py

      (..oo)
      //  @
      # ,VACA/
      //  ||
      //  ||

Virtual Accelerator with Channel Access server
Version 0.15.2
LNLS Accelerator Physics Group
Documentation: https://github.com/lnls-fac/va
Prefix: VA-
Number of SI pvs: 3472
Number of LI pvs: 836
Number of LI pvs: 6
Number of TB pvs: 147
Number of TS pvs: 125

2016-10-05 09:59:03.988: start LI_V00
2016-10-05 09:59:03.989: init epics sp memory for LI pvs
2016-10-05 09:59:03.910: start BO_V02A
2016-10-05 09:59:03.914: start SI_V17_01
2016-10-05 09:59:03.925: start TB_V01
2016-10-05 09:59:03.930: start TS_V01
2016-10-05 09:59:03.965: start waiting model initialisation
2016-10-05 09:59:03.981: init epics sp memory for TS pvs
2016-10-05 09:59:03.981: init epics sp memory for TB pvs
2016-10-05 09:59:04.027: calc transport efficiency for TB_V01
2016-10-05 09:59:04.327: calc closed orbit for BO_V02A
2016-10-05 09:59:04.367: calc linear optics for BO_V02A
2016-10-05 09:59:04.435: calc equilibrium parameters for BO_V02A
2016-10-05 09:59:04.743: calc beam lifetimes for BO_V02A
2016-10-05 09:59:04.803: calc closed orbit for SI_V17_01
2016-10-05 09:59:04.856: calc ejection efficiency for BO_V02A
2016-10-05 09:59:04.945: calc linear optics for SI_V17_01
2016-10-05 09:59:05.105: calc equilibrium parameters for SI_V17_01
2016-10-05 09:59:05.756: calc beam lifetimes for SI_V17_01
2016-10-05 09:59:05.910: init epics sp memory for BO pvs
2016-10-05 09:59:09.269: calc injection efficiency for BO_V02A
2016-10-05 09:59:09.271: calc ejection efficiency for BO_V02A
2016-10-05 09:59:09.431: calc transport efficiency for TS_V01
2016-10-05 09:59:09.638: init epics sp memory for SI pvs
2016-10-05 09:59:22.194: calc on axis injection efficiency for SI_V17_01
2016-10-05 09:59:22.928: start starting server
  
```

Figure 1: Screen printout of a command line terminal showing a running instance of VACA with the display of its banner and useful information.

VACA

The virtual accelerator with channel access is written in Python3. The choice of a high level programming language allows for rapid development of the intricate functionalities the virtual accelerator has to provide. Python language works as a binding layer between the two main core modules: the CA server and the tracking simulation code. The Python package PCASPy [3] is used to write the CA server module. PVs implemented in VACA have a prefix "VA-" indicating that they represent virtual process variables. For the simulation module, a library called trackcpp [4], that has been developed at LNLS by the accelerator physics group staff, was reused. It is a C++ library of beam dynamics calculations and tracking routines closely based on, and tested against, Tracy [5] and Matlab AT [6] passmethods. This library is converted to a python package using Swig3.0 [7], thus conveying fast and optimised routines that can be conveniently called within Python.

* ximenres.resende@lnls.br

Figure 1 is a screen print of a command line terminal showing a running instance of VACA with its banner and useful information printout. It shows the number of virtual PVs implemented for each major accelerator subsystem, labeled here SI,BO,LI,TB,TS. Also displayed are current model versions for all subsystems, as well as messages indicating that initial calculations are finished, such as those needed to simulate injection efficiency, for example. At last in the display, a message indicating that VACA is ready to respond to virtual PV queries.

VIOCS

The other part of the virtual accelerator is the set of virtual IOCs that respond to PVs with actual the control system names. So far a few vIOCS have been implemented:

- `si_bpm, bo_bpm, ts_bpm, tb_bpm`: they serve BPM positions for all subsystems that are read from VACA, adding emulated measurement fluctuations.
- `si_current, bo_current`: they provide simulated beam currents with fluctuations. Touschek, elastic and inelastic simulated lifetimes are affected by variations of associated parameters such as RF gap voltage and reduced acceptance due to closed orbit variations.
- `si_ps, bo_ps, ts_ps, tb_ps`: provide read/write access to PVs that correspond to power supplies with associated magnet excitation curves.
- `si_rf, bo_rf`: implement radio frequency process variables.
- `si_tune`: emulation of the tune measurement IOC.
- `si_beamsize, bo_beamsize`: emulation of beam size measurement IOC.
- `si_lifetime`: emulation of lifetime calculation IOC.

Most of these vIOCS are written using database records that are distributed with EPICS base. `si_lifetime`, on the other hand, is implemented with PCASPy.

CONCLUSIONS

The virtual accelerator described here has been an invaluable asset for the development of high level applications, as described in Ref. [2]. The core of the HLA development is planned to take place in 2017, mainly by the accelerator physics group staff. This development will certainly benefit from having a VA system available. At this point simulation of basic beam processes are implemented. VACA now properly simulates processes such as: parameter-dependent current decays, closed-orbit control with dipolar correctors, beam optics variations with quadrupoles, injection and ejection that depend on magnet and timing configurations. In the future more functionalities and modifications should be added to VA, such as:

- Timing details of the pulsed signals during injection and ejection processes need be considered.
- Approximate coupling expressions now used for beam size estimates should be replaced with more rigorous Ohmi's beam envelop formalism [8] in trackcpp,
- A cleaner separation between VACA and vIOCS is in order. At this points a few excitation curves are implemented in VACA since it has not been decided yet where they will finally be located in the CS. They can either be stored in the corresponding power supply IOCs – in which case they should be moved to the vIOCS for the VA, or stored in some configuration database service.
- Considerations on moving from EPICS database records PCASPy for vIOCS developments. This may simplify the process of writing and deploying applications.
- Recently a few DISCS [9] services have been adopted. In particular, the use of its naming service module allowed for a standardisation of how devices and PVs are named. As a consequence, a major revision of PV names has taken place recently. VA should be updated to contemplate the new PV naming standard.

REFERENCES

- [1] A. R. D. Rodrigues *et al.*, "Sirius Status Report", in *Proc. IPAC'16*, Busan, Korea, May 2016, paper WEPOW001, pp. 2811-2814.
- [2] I. Stevani, N. Milas, X. R. Resende, L. N. P. Vilela, "High Level Applications for Sirius", presented at PCaPAC'16, Campinas, Brazil, Oct. 2016, paper WEPOPRPO22, this conference.
- [3] PCASpy - EPICS Portable Channel Access Server python package, <http://pcaspy.readthedocs.io/en/latest>
- [4] LNLS Accelerator Physics Group, Tracking library, <https://github.com/lnls-fac/trackcpp>
- [5] H. Nishimura, "TRACY, A Tool for Accelerator Design and Analysis", in *Proc. EPAC'88*, Rome, Italy, Jun. 1988, pp. 803-805.
- [6] A. Terebilo, "Accelerator Toolbox for MATLAB", SLAC-PUB-8732, May 2001.
- [7] SWIG - Simplified Wrapper and Interface Generator, <http://www.swig.org/>
- [8] K. Ohmi, K. Hirata, K. Oide, "From the beam-envelope matrix to synchrotron-radiation integrals" *Phys. Rev. E* 49, 751, 1994.
- [9] V. Vuppala *et al.*, "Distributed Information Services for Control Systems", in *Proc. ICALEPCS'2013*, San Francisco, United States, March 2014, paper WECOA02, pp. 1000-1003

HIGH LEVEL APPLICATIONS FOR SIRIUS

I. Stevani*, N. Milas, X. R. Resende, L. N. P. Vilela
Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Abstract

Has been decided that Sirius will use EPICS as its distributed control system and this year the development of its high level applications started. Three development frameworks were chosen for building these applications: CS-Studio [1], PyQt [2] and Matlab Middle Layer [3] (MML). Graphical user interfaces (GUI) and machine applications have already been designed and implemented for a few systems using CS-Studio and PyQt: slow orbit feedback, lifetime calculation and top-up injection. Specific Sirius data structures were added to the MML scripts in order to allow for EPICS communication through LabCA [4].

INTRODUCTION

Sirius is a synchrotron light source facility based on a 4th generation storage ring that is presently under construction at LNLS in Campinas, Brazil. The project initiated in 2008 with the first lattice studies and in the beginning of 2015 the building construction started. Machine commissioning is expected to start mid 2018 [5]. By this time, indispensable control systems for beam stability – such as tune measurement, orbit correction, injection – should be ready for use, and so should their high level applications (HLAs).

To achieve this goal, the accelerator physics group started developing these applications this year and should intensify its activity with HLA next year, after most of the group's work related to machine components' specifications – that is currently taking most of the group efforts – is done. A virtual accelerator with channel access server (VACA) [6] has been implemented to emulate the real machine, thus providing a testbed environment for high level software development.

APPLICATIONS FRAMEWORK

Figure 1 shows a schematic view of the application framework Sirius will use for HLA development. On a lower level, services are categorized in two groups: IOCs and machine applications. Both of them are channel access (CA) servers, the first providing control access to machine hardware while the second providing machine services. Machine applications developed so far were written in Python using PCASpy [7]. Virtual IOCs were implemented using EPICS Database on top of VACA in order to provide simulated fluctuations for machine parameters. EPICS V4 is another option for implementing machine applications that will be studied and discussed next year.

On top of Fig. 1 are the client applications. For basic process variable monitoring and settings with graphical user interface capabilities, CS-Studio was chosen due to its simplicity and fast learning curve. Most of the control system

interface is expected to be implemented with CS-Studio. For software that requires elaborate logic, algorithms will either be implemented in the IOC level with simple driving GUI clients or using other options, such as PyQt or MML.

The control system can benefit from various support applications for storing and organizing system's parameters, IOC software, device and PV name lists, and so on. The plan is to use a few of the applications in development under the DISCS collaboration effort [8,9]. The device naming and configuration modules of DISCS are currently being tested and used in the HLA development framework. Other DISCS modules, such as its logbook, IOC and machine save/restore services, will soon be tested as well.

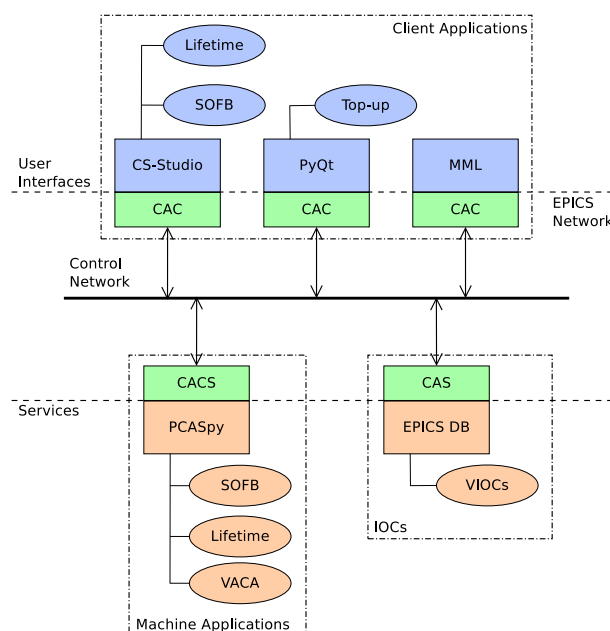


Figure 1: Applications framework schematic.

IMPLEMENTED APPLICATIONS

Some prototypes were built on these platforms listed before to help the developers familiarize with their functionalities and evaluate if they are suitable for Sirius requirements. The HLAs are presented next.

Slow Orbit Feedback

The slow orbit feedback (SOFB) system software is an example of a HLA that has already been written for Sirius. It consists of two modules: a machine application that reads BPMs and controls orbit correctors and a user graphical interface application.

The first one is responsible for all data processing and control, such as orbit/response matrix measurements and

* isabella.stevani@lnls.br

orbit corrections based on singular value decomposition (SVD). Standard useful functionalities were implemented: device selection, response matrix and reference orbit configurations, variable-size buffers with orbit data for averages, optional inclusion of RF frequency in the correction loop and corrector strength adjustments. These functionalities were implemented in Python modules that are imported in a PCASpy SOFB server. Python threads were used to allow for uninterrupted CA service while data processing and calculations are performed in response to prior user interactions.

As depicted in Fig. 1, the second SOFB module is a user-friendly GUI application with a CA client layer underneath that allows for monitoring and controlling the SOFB system with embedded graphics tools, thus facilitating machine commissioning and operation. This module was designed in CS-Studio and it interacts with the SOFB machine application module through a set of PVs.

The prototype application is shown in Fig. 2. Its design was based on the orbit correction interface that is being used in the existing UVX storage ring at LNLS which has a proprietary control system. The main features of the prototype application are plot displayed with measured orbit, widgets set for manual correction, the machine application mode selection and widgets for configuring sampling parameters for orbit average calculations. Although not all functionalities have been implemented on the client application yet, the current version should suffice for initial commissioning stages of the storage ring. Selection of BPMs and correctors for the correction algorithm, as well as response matrix and reference orbit configurations, will be added in the near future.

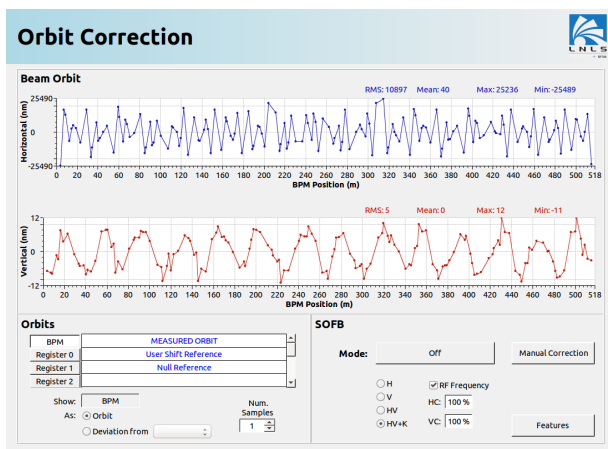


Figure 2: Orbit correction display application in the CS-Studio environment.

Lifetime Calculation

Another example of implemented HLA is the lifetime calculation. Just as in the case of the SOFB system, here two separate application modules were created. In order to simplify the user interface, the algorithm that computes beam

lifetime was moved to a machine application implemented in EPICS DB, while the client application was designed in CS-Studio. The calculation method in the machine application is the same as the one already in use in the UVX storage ring and that has been proven reliable and stable over the years. The details of its implementation can be found here [10].

As always with beam lifetime calculations, there is a trade-off between precision and quick response to lifetime variations. This trade-off is realized by means of two input parameters, precision and sampling time, that define the time window and number of points for the lifetime calculation. In the user interface it is possible to display the lifetime in units of hours, minutes or seconds, as shown in Fig. 3. Being able to display lifetime in minutes or seconds might be useful specially on the early days of machine commissioning.

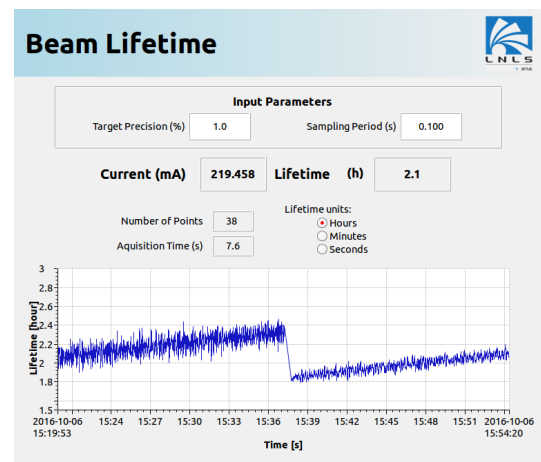


Figure 3: Lifetime display application in the CS-Studio environment.

Top-up Injection

After the commissioning phase, the Sirius injection system will operate in top-up mode and a control application will be necessary to conduct the required periodic injections. A simplified version of this HLA was developed to test the simulation of the injection process in VACA. It consists of a PyQt application that monitors the beam current in the storage ring and starts the injection cycle in order to maintain the current level within the desired tolerance. The connection between the CA protocol and the Python language was done with the PyEpics package [11]. The user interface contains a display of the beam current over time and the fill pattern in the storage ring, as shown in Fig. 4.

In order to include this application in the Sirius control system, several functionalities must be added. For example, in this version, it is only possible to choose between multi-bunch or single-bunch injection mode. However, as the timing system will support the injection to any bucket, the final version of this HLA should allow the specification of any filling pattern.

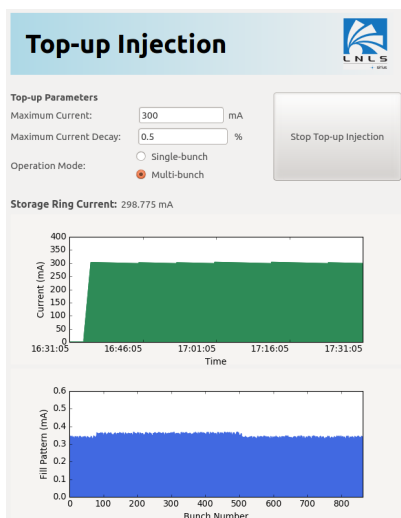


Figure 4: Top-up injection display application developed with the PyQt framework.

CONCLUSION

The development of Sirius' HLAs started this year. Three development frameworks were chosen and then tested on three important machine systems: SOFB, lifetime calculation and top-up injection. Results were promising and the plan is to continue improving these systems – adding missing functionalities, for example – and start the development of new HLAs such as LINAC gun and timing system controls, which are also fundamental to machine commissioning.

Also, has been decided that the control system could benefit from applications in development under the DISCS collaboration effort. Its device naming and configuration modules are currently being tested and used in the HLA development framework. Other DISCS modules will soon be tested as well.

ACKNOWLEDGMENTS

The authors would like to thank L. M. Russo from the Diagnostics group for fruitful discussions on EPICS and for his help in configuring all services modules from DISCS which we have been testing.

REFERENCES

- [1] Control System Studio, <http://controlsystemstudio.org/>
- [2] PyQt, <https://wiki.python.org/moin/PyQt>
- [3] G. Portmann, J. Corbett and A. Terebilo, "An accelerator control middle layer using MATLAB", in *Proc. PAC'05*, Knoxville, United States, May 2005, paper FPAT077, pp. 4009-4011.
- [4] T. Straumann, "labCA – An EPICS Channel Access Interface for scilab and matlab", May 2008, <https://www.slac.stanford.edu/grp/ssrl/spear/epics/extensions/labca/manual/>

- [5] A. R. D. Rodrigues *et al.*, "Sirius status report", in *Proc. IPAC'16*, Busan, Korea, May 2016, paper WEPOW001, pp. 2811-2814.
- [6] X. R. Resende, A. H. C. Mukai, L. N. P. Vilela and I. Stevani, "Development of a virtual accelerator for Sirius", presented at the PCaPAC'16, Campinas, Brazil, Oct. 2016, paper WEPOPRPO21, this conference.
- [7] PCASpy Documentation, <http://pcaspy.readthedocs.io/en/latest/>
- [8] V. Vuppala *et al.*, "Distributed Information Services for Control Systems", in *Proc. ICALEPCS'2013*, San Francisco, United States, March 2014, paper WECOA02, pp. 1000-1003
- [9] DISCS, <http://openepics.sourceforge.net/about-discs/>
- [10] A. J. Burns *et al.*, "Real time monitoring of LEP beam currents and lifetimes", in *Proc. EPAC'94*, London, England, pp. 1716-1718.
- [11] PyEpics, <http://cars9.uchicago.edu/software/python/pyepics3/>

BEAMLINE SUPERVISORY SYSTEM USING A LOW-COST SINGLE-BOARD COMPUTER

G. T. Semissatto[†], F. H. Cardoso, H. F. Canova, J. Souza
LNLS, Campinas, Brazil

Abstract

Sirius is the new accelerator facility, under construction at the LNLS (Brazilian National Synchrotron Light Laboratory) site, in Campinas, São Paulo. The new machine is a 3 GeV, low emittance storage ring designed to accommodate up to 40 experimental stations. During beamline operation, supervisory systems are an important tool to provide information about machine status and beamline operation modes for the beamline's users. A modern TV based broadcast system was developed to meet this application, using low-cost single board computers with an interface to EPS/PPS system. The details about hardware, software configuration, user's requirements as well suggestions on further improvements, will be presented.

INTRODUCTION

At UVX, the current machine at LNLS, the machine status broadcasting system is based on antiquated cable television topology. Furthermore, beamline status is monitored only using audible sounds (at hutch armed status) and light indicators. This system has become obsolete and is no longer manageable, with an additional drawback regarding the support for people with color vision deficiency (such as Daltonism). A new and modern supervisory system is envisioned for Sirius, to provide readable warnings, useful information and customized sounds for beamline users.

A broadcast system, based on web browsers and LED TV's is proposed. The system displays the machine status (such as storage ring current, beam lifetime, machine energy, top-up) and beamline operations modes (photon beam status: beam on, beam off, imminent beam, hutch armed, etc).

The supervisory system was developed using a low-cost single board computer, the Beaglebone Black [1], integrated with the Personnel Protection System (PPS) and Equipment Protection System (EPS), both based on industrial programmable logic controllers (PLCs). Beaglebone Black (see Figure 1) is a powerful low-cost single board computer equipped with a Sitara ARM Cortex-A8 processor running at 1 GHz. The board provides 512 MB of RAM, two 46-pin expansion connectors (with much general-purpose I/O pins), on-chip Ethernet, a microSD slot, a USB host port and HDMI video interface. The board also runs many distributions of embedded Linux providing support for a variety of programming languages.

By means of Python scripts and web browsers, the system was designed for low cost, high flexibility and expandability. This also allows for a efficient way to create fully customized warning messages for users.

BEAMLINE SUPERVISORY SYSTEM

System overview

During normal operation, the system provides machine status (storage ring parameters) using a web-based application. The operation group, based in the storage ring control room, can configure this information. Besides the storage ring information, the beamline's users have the need to know the photon beam status, the beamline operation mode and any other information required for that specific experimental station, such as lasers, high pressure cells, robots, etc.

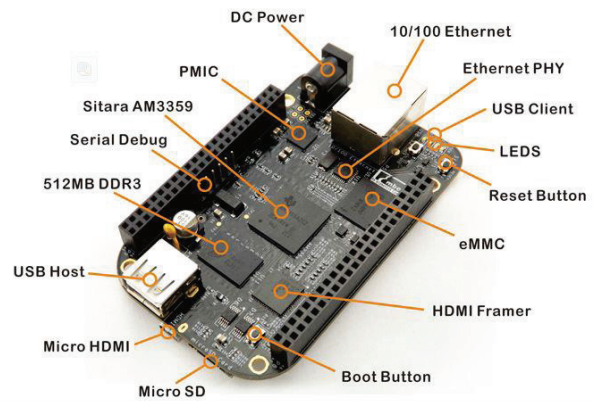


Figure 1: Beaglebone Black.

The beamline supervisory system flowchart is shown below (see Figure 2). The PPS is an engineered interlock system that monitors the various devices installed in the beamline, for personnel safety and provide emergency beam shutdown. PPS is based on PLC's that monitor the positions of the front-end shutters, beamline shutters and beamline hutch doors. This system also defines all the safety functions to avoid any accidental exposure to radiation inside the beamline hutch.

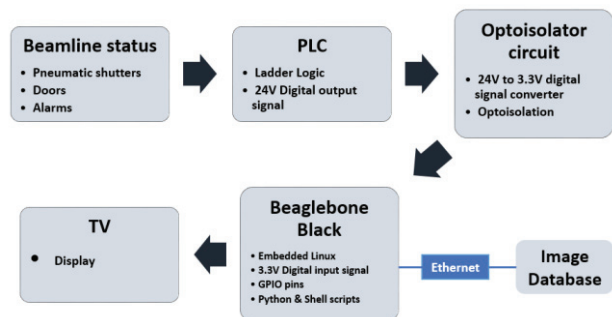


Figure 2: Supervisory system stages.

[†] guilherme.semssatto@lnls.br

Once the beamline mode is determined by each PPS, the PLC will send digital signals to the signal conditioner mezzanine board. The mezzanine board translates the 24V logic level from PLC to 3.3V LVTTTL used by the Beaglebone board.

A total of 8 digital signals is provided by the PLC. One of these signals is used as an interruption or control signal, while the remaining 7 signals are used to identify the beamline status, as defined in the protocol identification software. Considering this range of signals, 128 status messages can be controlled by the Beaglebone board. Treatment of the received signals and image display is controlled by Python script using the Pygame package [2] and Adafruit Beaglebone IO Python library [3].

HARDWARE OVERVIEW

The Mezzanine Board

As previously mentioned, the standard voltage of PLC's is 24V. The Beaglebone Black works with LVTTTL levels (3.3V). In this sense, a logic translation circuit, using optocouplers, was designed (see Figure 3).

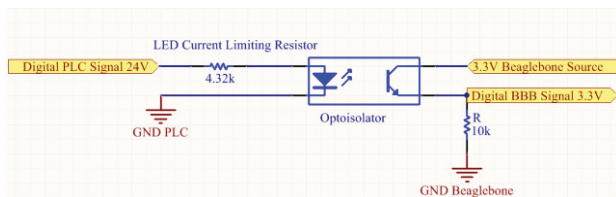


Figure 3: Opto-isolator circuit.

When the PLC's digital signal is at a high level, the photodiode is directly biased and makes the phototransistor transmit its collector signal (3.3V Beaglebone source) to the emitter (GPIO pin). Thus, it guarantees insulation of signals without the need for an additional external voltage source. The circuit receives the PLC ground reference and the 8 digital signals, which are reduced to 3.3V logic reference, are then transmitted to their respective GPIO pins.

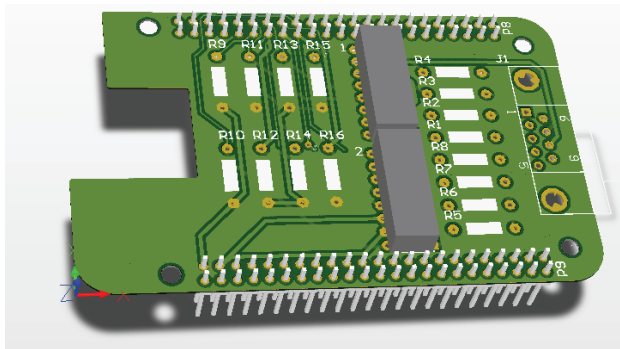


Figure 4: 3D view of the optocoupler circuit mezzanine board prototype.

The processing board interprets the signals received and shows the proper image with the relevant message, stored in a common directory on the internal network in which all boards are connected. The connection to the TV is performed using the available micro HDMI video output of

the Beaglebone. The TV screen resolution of is not of relevance, since the software has been implemented to adapt to any resolution.

To accommodate the optocouplers circuits, a mezzanine board was designed (see Figure 4).

The Beaglebone standard mezzanine cards are called "capes". The prototype board design was developed using Altium designer PCB software that provides schematic capture and Printed circuit board (PCB) design (see Figure 5).

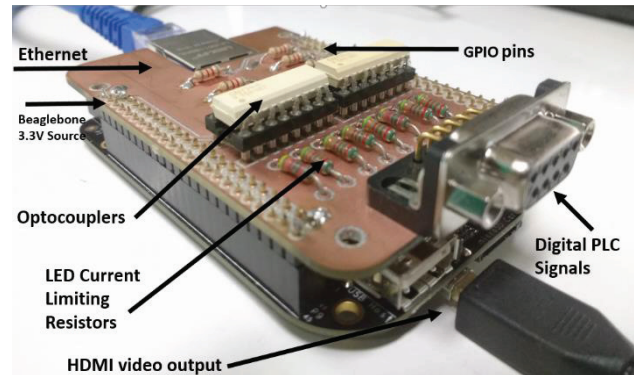


Figure 5: Prototype cape embedded in Beaglebone.

SOFTWARE OVERVIEW

Operating System and Software Configuration

When there is no change in the beamline status, Beaglebone shows the standard web page, displaying the storage ring information, on the screen (see Figure 6). The default web page provides information about the machine status such as beam energy (MeV), instant beam current (mA), electron beam lifetime (hours) and the status of all gamma shutters presented in the experimental stations. The electron beam of the current graph shows the trend of the last 15 hours.



Figure 6: full frame display of Web-based storage ring status.

When the beamline status changes, the PPS' PLC generates an interrupt signal to the Beaglebone board. A single Python script that is constantly running in each Beaglebone performs treatment of the signal.

The Python script uses the Adafruit GPIO library to read the interrupt signals and Pygame package libraries to

search and load images in full-screen mode (see Figure 7). All this set of actions (digital signal variation + interrupt generation + search and loading of images) is performed in less than 150 ms. Pygame showed the fastest response to display images in comparison to other tested Python modules (OpenCV, Pillow, PyQT, etc.).

After extensive tests, Debian 7.5 (codename "Wheezy"), was found to be the most responsive operating system for the application.

Image Database

At Sirius, it estimated that tens of single board's computers, running this application, would be used. In this regard, maintainability must be optimized. All boards will share a common directory to access images. This will ensure that updates in the alert images database are common to all boards.

For this purpose, a Windows network point was created and mapped on all boards. Images were inserted into this network point and can be modified by any user with the appropriate permissions. Windows/Linux network mapping is done and managed using GNU/LINUX that supports Common Internet File System (CIFS) [4].

Automation Scripts

After the initial setup, the supervisory system operates autonomously (plug and play). Automation mechanisms were implemented using Shell scripts that run when Beaglebone boots up. These scripts have been implemented for two main purposes:

- Ensure standard Python script and web page, described above, are always running.
- Ensure Windows/Linux network mapping is achieved every time the board is restarted.

In order to perform a fast initial setup in a new Beaglebone board, another Shell script was developed. The script acquires and configures all modules used in the project. With this script, a new Beaglebone can be easily configured.

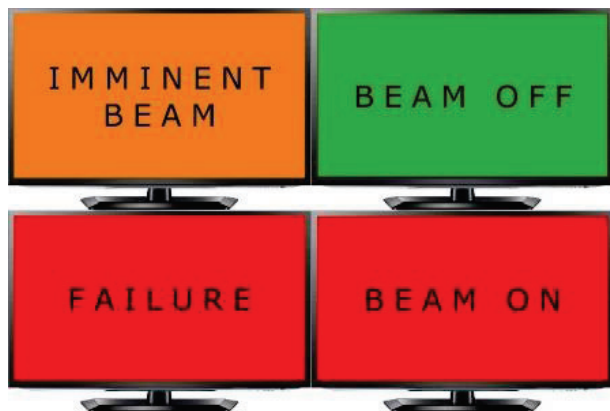


Figure 7: Examples of images showing Beamline Status on TV.

In order to perform a fast initial setup in a new Beaglebone board, another Shell script was developed. The script gets and configures all the modules used in the project.

With this script, a new Beaglebone can be efficiently configured.

CURRENT STATUS

The system is under evaluation at the UVX machine. The Sirius' prototype hutch will receive the supervisory system and the PLC code must be developed to meet the logic required for full compatibility.

Moreover, an improved procedure for global edition (global upgrade) is also under development.

CONCLUSION

Based on the advance of the computer networks and low-cost, high resolution LED TV's, web-based broadcasting supervisory system can provide concise and comprehensive information about the storage ring status.

Using low-cost single board computers, a new and modern supervisory system was successfully designed for Sirius' experimental stations. The proposed topology aims to provide readable warnings, useful information and customized sounds for the beamline users. People with Daltonism could now be warned about the beamline status.

During normal operation, the system provides machine status (storage ring parameters) using a web-based application. With respect to changes in the beamline mode of operation, interruption from PPS PLC's are treated by the system and relevant images (with priority) are shown on TV's spread in the experimental station.

ACKNOWLEDGEMENT

The authors would like to thank colleagues from LNLS who contributed to valuable discussions and provided feedback. Special thanks to the Beamline software group (SOL).

REFERENCES

- [1] Beaglebone Black, <https://beagleboard.org/black>
- [2] Pygame package, <http://pygame.org>
- [3] Adafruit IO Python, <https://learn.adafruit.com>
- [4] Samba, <https://samba.org>

VDE - VIRTUAL DOCUMENTATION ENVIRONMENT

H. F. Canova*, D. H. C. Araujo, M. P. Donadio, R. R. Gerales
LNLS, Campinas, Brazil

Abstract

At LNLS hundreds of motors are used at the beamlines to move parts, equipment or full systems, according to different profile, synchronization and accuracy requirements. Historically, the documentation of motion axes of the LNLS beamlines was either done only at the moment of their installation and commissioning, or not properly done at all. Thus, after some time, keeping track of changes and performing maintenance could turn out to be very challenging, and there was the clear need of some solution to ensure that every change in motors would be reflected in their documentation. In 2012 the migration of the beamlines control system to the EPICS (Experimental Physics and Industrial Control System) [1] platform pushed the development of a new documentation system. In a first version, it consisted of a smart spreadsheet that generated the EPICS configuration files automatically. Later the spreadsheet evolved to a web-based system the VDE - Virtual Documentation Environment, which allows the beamlines staff to change the motion axis parameters without the need of a deep knowledge about EPICS and ensures the complete motion axis documentation intuitively. Also, changes in motors will not work in EPICS if the documentation is not updated, guaranteeing the link between documentation and the real system.

INTRODUCTION

Currently, around 700 motorized mechanisms are installed at LNLS beamlines, being used, for instance, to perform sophisticated optical alignment of mirrors and monochromators and position samples. These motorized mechanisms can be abstracted as motion axes, which are composed of motors (mostly stepper motors), gear boxes, transmission elements and encoders, when necessary. All these axes are integrated into the EPICS distributed control system of the beamlines and endstations.

The most common motion controllers at the LNLS beamlines are Galil [2], Parker [3] and IMS [4] devices whose EPICS IOCs (Input/Output Controller) run in a dedicated National Instruments PXI chassis [5], for the main beamline components, or in a Virtual Machine, in the case of separated endstations.

DPM

In 2012, with the need to document the motorized axes of the beamlines, until then based only on mechanical drawings, and to facilitate the software configurations, by providing all the relevant mechanical parameters, a standard configuration tool, the so called DPM, has been implemented. It was based on two spreadsheets, one being used as a library and

the other, the master spreadsheet for the beamline. In the first, there were records of models and manufacturers, for motors, drivers and gearboxes, for instance, whereas the latter concentrated the information of each axis of the installation in a line on the spreadsheet. It also had the necessary information to perform all the needed calculations and determine the primary parameters of the axes as speed and resolution, for example. After all the axes were correctly registered in the master spreadsheet, the DPM was able to generate the necessary configuration files for the EPICS IOCs using VBA macros in the spreadsheet. Finally, all the configuration files were stored in text format and a LabVIEW application was used to transfer the file to the IOC server by FTP.

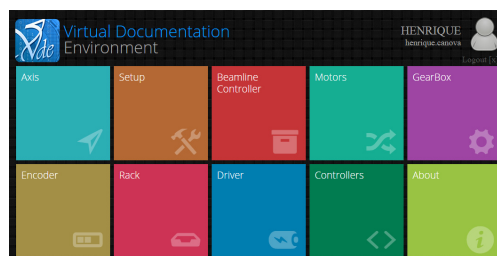


Figure 1: VDE home screen.

VDE

With the need to make a more robust tool, in order to avoid errors during the operation, and to add more features, the VDE was created. Allowing for greater versatility a web-based platform running on an Apache server, and a relational MySQL database is used. The main language of the system, to access the database and to manipulate the data, is PHP/JavaScript, whereas the pages designs are based on HTML5 and CSS3.

The VDE consists of a series of different web pages, each one giving access to each part of a motor axis and its respective entries. The home screen (Fig. 1) shows all the available options in the platform, with even though some of them are for the administrators only.

The system is integrated with the LNLS Active Directory System to provide a way for the user to login using the institutional username and password from any computer in the campus. Thereby, this enables the VDE to set different access levels to restrict access to certain sets of motors for the desired users. For example the home screen, viewing access to the registered common equipment is allowed for general user, however, only admins can edit or register new items. Naturally the administrators also have access to all axes in the database.

* henrique.canova@lnls.br

Figure 2: VDE axis configuration - Software options.

Figure 3: VDE controller screen.

Concepts

The VDE is based on the concept of Setups. Each installation (beamline or endstation) can have one or more Setups, each one consisting of a group of motor controllers (Fig. 3) and their respectively configured axes. This concept was created because the motor parameters often change at the beamlines depending on the experiment requirements, i.e., the same axis may have, for example, different speed and acceleration levels, or synchronism targets, for different experimental setups.

Previously, the beamlines staff needed to individually change the motors parameters for each experiment. Using the VDE, the setups may be pre-configured and stored, so that all the axes can be reconfigured at once.

Axis

After the registration of all existing motor controllers at the beamline or endstation, and the creation of the Setup structure, the axes are configured in a dedicated page (see Fig. 2). In this page, there are specific tabs for each compo-

nent of the given axis, namely: general description of the axis, motor type, gearbox, transmission rate, geometric conversion, limit and home switch configuration, encoder, controller configuration, driver stage and software data (EPICS).

In the software tab, the records of the EPICS Motor Record [6] IOC are exhibited. Part of them can be modified by the users, whereas some are automatically calculated using data from other tabs. As an example, the step resolution (MRES) is determined using information from the motor, the gearbox, the transmission rate, and the geometric conversion tabs. For some cases, advanced options are available to modify IOC parameters and to create extra substitutions files.

Equipment

According to this work flow, the components must be registered in the VDE, building some kind of library. This work is shared by the beamlines electronics support group (GAE), which is responsible for the configuration of drivers and the physical connections, and by the beamlines software support group (SOL), responsible for the configuration and installation of the IOC in EPICS. Currently, motors, gear boxes, encoders, motion controllers, power stages (drivers) and racks (the controller chassis) from several manufacturers are already supported, but whenever new devices are to be installed, they must be properly registered in the system.

File Generation

After all the axes are properly configured, the generated configuration files, including the Motor Record substitution and command files of the IOC, can be sent to the EPICS host computer by SSH - Secure Shell - protocol under Linux. Figure 4 shows the auxiliary screen in which the user can select which controller family is to be configured. Currently, VDE can generate IOCs configuration files to Galil, Parker, and IMS controllers.

Figure 4: VDE - send configuration screen.

Figure 6: VDE administration page.

Reports

In addition to the online documentation, VDE is able to generate two types of report: one simply with the list of all registered axes in the Setup (see Fig. 5) and a detailed one with all the information gathered from each axis.


 XRD1		Page 1
Controller: IMS 1 Address: COM3		Controller: PARKER 1 Address: COM7
1 FENDA BRANCA SUPERIOR	5 NIVELADOR 1	
2 FENDA BRANCA INFERIOR	6 NIVELADOR 2	
3 FENDA BRANCA DIREITA	7 NIVELADOR 3	
4 FENDA BRANCA ESQUERDA	8 CURVATURA	
5 FREE	1 FREE	
6 FREE	2 FREE	
7 FREE	3 FREE	
8 FREE	4 FREE	
Controller: GALIL 1 Address: 10.2.41.34		Controller: GALIL 2 Address: 10.2.41.33
1 ROLL	1 TELA FLUORESCENTE 2	
2 MCTH2	2 FENDA ESPALHAMENTO SUPERIOR	
3 YAW	3 FENDA ESPALHAMENTO INFERIOR	
4 VERT	4 FENDA ESPALHAMENTO ESQUERDA	
5 CURVATURA	5 FENDA ESPALHAMENTO DIREITA	
6 MONO	6 FREE	
7 MONITOR DE FIO	7 FREE	
8 TELA FLUORESCENTE 1	8 FREE	
Controller: GALIL 3 Address: 10.2.41.32		Controller: GALIL 4 Address: 10.2.41.31
1 CAMARA SEGMENTADA Z	1 CABECA GONIOMETRICA EIXO X	
2 CAMARA SEGMENTADA X	2 CABECA GONIOMETRICA EIXO Y	
3 FENDA ADC INFERIOR	3 CABECA GONIOMETRICA EIXO AX	
4 FENDA ADC SUPERIOR	4 CABECA GONIOMETRICA EIXO AY	
5 FENDA ADC ESQUERDA	5 FREE	
6 FENDA ADC DIREITA	6 FREE	
7 FREE	7 FREE	
8 FREE	8 FREE	

Figure 5: VDE - axis list report.

Administration

On the administration page (Fig. 6), it is possible to modify configurations, such as IP address of the EPICS host computers, and to add manufacturers do the database.

CONCLUSIONS AND PERSPECTIVES

The VDE is currently applied to all experimental stations and beamlines at LNLS and it is expected to be applied in the new accelerator, Sirius. The VDE development is ongoing and different manufactures as Newport [7] and Aerotech [8], are expected to be integrated to the file generation system soon. Another tool that is in the testing phase is the capacity to automatically change, not only the IOCs configuration but also lower level settings of the drivers, as current and resolution, for example.

ACKNOWLEDGEMENT

The authors would like to thank the colleagues from LNLS who contributed to valuable discussions and feedback about the VDE.

REFERENCES

- [1] EPICS, <http://www.aps.anl.gov/epics>
- [2] Galil Motion Controllers, <http://www.galilmc.com>
- [3] Parker Motion, <http://www.parkermotion.com>
- [4] Intelligent Motion Systems - IMS, <http://motion.schneider-electric.com>
- [5] J. R. Piton *et al.*, "HYPIIE: A HYPERVISORED PXI FOR PHYSICS INSTRUMENTATION UNDER EPICS", BIW 2012, Newport News, MOPG031.
- [6] EPICS: Motor Record and Device/Driver support, <http://www.aps.anl.gov/bcda/synApps/motor>
- [7] Newport, <http://www.newport.com>
- [8] Aerotech <http://www.aerotech.com>

USING TKINTER OF PYTHON TO CREATE GRAPHICAL USER INTERFACE (GUI) FOR SCRIPTS IN LNLS

D. B. Beniz[†], A. M. Espindola[‡], Brazilian Synchrotron Light Laboratory, Campinas, Brazil

Abstract

Python is being widely used to create scripts which cover different necessities in computational scenario. At LNLS (Brazilian Synchrotron Light Laboratory) we successfully developed Python scripts to control beamlines operations, including a case of GUI (*Graphical User Interface*) creation using Tkinter [1] for one of LNLS beamlines, DXAS (*Dispersive X-ray Absorption Spectroscopy*) [2]. In this article its motivation and some implementation details will be presented.

MOTIVATION

The decision to use Python to build a GUI was based on the previous experience with such programming language in LNLS. There is a library package, called Py4Syn [3], developed in LNLS with Python version 3.4 and in use to control beamline devices, like motors and detectors, and to operate a sequence of actions to perform specific experiments by synchronization of a set of such devices and storing of collected data into files formatted in columns to facilitate their analysis. Controllers of such devices have software abstractions, IOCs (*Input/Output Controllers*), developed in EPICS (*Experimental Physics and Industrial Control System*) [4] which resources, PVs (*Process Variables*), are available in the laboratory network via CA (*Channel Access*) [5] protocol. Python offers packages to control such resources, with PyEPICS [6], to perform mathematical calculations and data matrix manipulation, with NumPy [7], and to display data as graphics, with Matplotlib [8], this way it facilitated the development of Py4Syn.

Once we had the tool to elaborate scripts to orchestrate synchrotron beamline experiments, Py4Syn, the new challenge was to offer a GUI that helped users to inform scripts parameters, control and monitor their execution. LNLS adopted CS (Control System) Studio [9] as the tool to monitor and operate EPICS IOCs. It is a great option to monitor and interact with EPICS PVs, however, it is not recommended by their developers to control complex scripts in Python. We tried to use CS Studio to control Py4Syn scripts but the performance was unsatisfactory. Then, we decided to build a GUI in Python, once the scripts were written using that language, and Tkinter arose as a good start as it is the standard GUI package of Python and we found a large number of tutorials and code examples in the Internet. The first experience in LNLS with Tkinter to build graphical interfaces for Python scripts was with DXAS beamline, which was being reformed between 2014 and 2015.

[†] douglas.beniz@lnls.br
[‡] alexey.espindola@lnls.br

ARCHITECTURE OVERVIEW

In Figure 1 the overview of current software architecture solution for control system of DXAS is presented.

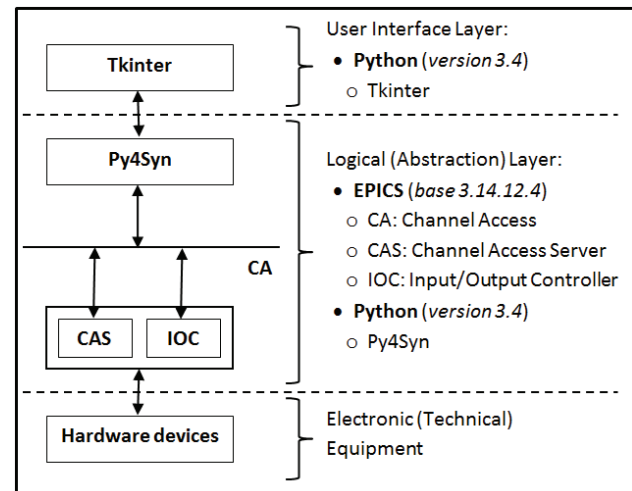


Figure 1: Overview of DXAS Solution Architecture.

Electronic (Technical) Equipment

At lower level of control system are the typical technical devices present in synchrotron beamlines. In fact, they have their own controllers which receive instructions, via serial (RS232/RS248) or Ethernet connection, for example, and then command the equipment. Some devices present in DXAS beamline of LNLS are:

- Galil DMC-4183: motor controller
- Parker OEM750: motor controller
- Heidenhain MT 2501: optical encoder
- Keithley 6485: picoammeter
- Kepco BOP: power supplier
- OMRON E5CK: digital controller of furnace
- LakeShore 331: temperature controller of cryogenic cooling system
- Stanford SR570: low noise current preamplifier
- Princeton Instruments PyLoN: CCD camera

Logical (Abstraction) Layer

Over the devices controllers is the first abstraction of them, build in EPICS, with correspondent IOCs for each one of the devices. Those devices of the same manufacturer and model share the same IOC program, but run in individual instances. Basically, a set of instructions to get information or to send a command to devices is organized in those IOCs as PVs, where each PV is a record, or piece of data, with some attributes to format, configure or simply return related information.

All PVs are broadcasted in the network via CA protocol, in which subnet where CAS is connected they are accessible.

The second abstraction level of those PVs is made by Py4Syn, which offers a set of Python objects representing each one of devices controlled by EPICS IOCs. Py4Syn also implement a set of utility tools to perform scan of motor positions while a counting detector is accumulating information of beam intensity transmitted across a sample, or to vary a furnace temperature while a CCD is acquiring spectra images to analyse sample absorption of x-ray, for example. It also allows a combination of motor movements, like a mesh of two motors, and mathematic calculations based on the measure of one or more detectors.

User Interface Layer

Finally, the top layer of this architecture is the GUI. The focus of this article is the graphical interfaces for scripts that perform automation of procedures to execute experiments in DXAS beamline of LNLS. Such GUIs were implemented, and are being used to operate the beamline since the beginning of 2016, using, mainly, Tkinter package of Python.

The main finalities of such interfaces are to receive parameters and to control the flow of experiment operations. Parameters can be those necessary to configure devices, like the current/voltage of a power supplier of a magnet coil, or the temperature stages (amplitude and duration) of a furnace, or parameters to define initial and final conditions of each experiment, like interval of motor positions to scan, or beam energy amplitude to perform the experiment, or energy interval to scan. Flow operations involve actions of start, pause, when applicable, or stop the experiment.

Some interfaces also show information of some devices that are monitored in short intervals, updated each 100 ms, e.g., like current furnace temperature, power supplier voltage, motor position, among others.

TKINTER CONCEPT

Tkinter, or “Tk interface”, is a module of python that provides an interface to Tk GUI toolkit, developed in TCL (*Tool Command Language*) and multiplatform, with support for Linux, MAC OS and MS Windows. Tk is natively present in Linux and MAC OS, and can be easily installed on MS Windows, it is not part of Python. Tkinter is part of Python, being called “Tkinter” in versions prior to 3, and “tkinter” on version.

Widgets, geometry management and event handling are the three main concepts of Tk, which also apply for Tkinter.

Widgets

Often referred to as controls, or window elements, widgets are all visible components on a graphical interface. Some examples are frames, labels, buttons, text entries, checkboxes, tree views, scrollbars, and text areas.

Inside a Python script, widgets are objects, or instances of classes that represent mentioned window components. To instantiate an object on Tk, and then on Tkinter, is necessary to indicate its parent, what maintain a window hierarchy between all elements. On that hierarchy, the main window is the root. Each widget has a set of configuration options which control how they are displayed or how they behave, like a “text” option for components that display some text, as a label, or “command” option when they accept events, as the mouse click of a button.

Geometry Management

An important step of interface design is to organize the widgets onscreen window. The most useful method to do that using Tk, or Tkinter, is by a geometry manager, like “grid”. In practice, “grid()” is a method available to all supported widgets saying to then where exactly to be positioned in an invisible matrix of columns and rows.

Combination of nested frames and grid is the better approach to design a Tk/Tkinter interface.

Event Handling

Tk/Tkinter manages the event loop that receives user actions over the window components, controlled by operating system, like button presses, keystrokes, mouse movement, and window resizing.

Individual widgets know how to respond to events. Basically, it provides a callback that can be assigned to a procedure in Python code as a configuration, like “command” for button widgets. For events without a callback command associated with them, it is possible to use an event binding, which in practice is the use of “bind()” method on a widget to capture any event and then execute an arbitrary procedure or method.

Another important method available for widgets is “after()”. Using it is possible to create an execution thread forked from the main application loop. At this new thread, a set of Python instructions is performed in parallel with interface updating. Besides, widget that calls it continues to be responsive to any user input.

TKINTER SOLUTION OF DXAS

For DXAS beamline of LNLS the main techniques supported are detection of very weak signals XANES (*X-ray Absorption Near-Edge Spectroscopy*), XAFS (*X-ray Absorption Fine Structure*), XMCD (*X-ray Magnetic Circular Dichroism*), XRMS (*X-ray Resonant Magnetic Scattering*), Catalysis and Cryogenics experiment (analysis of *X-ray Absorption* during a very high or very low temperature exposition).

So, the graphical interfaces built for DXAS control system solution cover these operations:

- **Spectroscopy**
- **X-ray Absorption**
- **XMCD / XRMS**
- **Catalysis / Cryogenics**
- **Motor Scan**

And some specific operations that include devices monitoring:

- **Slits** (with virtual motors built by Py4Syn conjugating real motors)
- **E5CK** (with temperature ramp programming and furnace monitoring)
- **Kepec BOP** (to set and monitor power supplier configuration and amplitude)
- **Generic graphics plot** (to plot graphics from any supported file by this solution)
- **Generic data consolidation** (to calculate results for all supported files generated by this solution)

Figure 2 below shows an excerpt of source code of XMCD and XRMS interface. On that, it is possible to see a LabelFrame and some Button widgets being instantiated, with their configurations being set, as described above in “Tkinter Concept” session.

```

236 # -----
237 self.buttonsFrame = LabelFrame(self.master, text = 'Run experime
238 self.buttonsFrame.pack(side = TOP, fill = X, pady = 2, ipadx = 2
239 self.startButton = Button(self.buttonsFrame,
240 command = self.StartExperiment,
241 text = "Scan",
242 font = textFont4,
243 background = "green",
244 fg="black",
245 state = DISABLED,
246 height = 1,
247 width = 8)
248 self.startButton.pack(side = LEFT, anchor = W, padx = 10, pady =
249 self.stopButton = Button(self.buttonsFrame,
250 command = self.StopExperiment,
251 text = "Cancel",
252 font = textFont4,
253 background = "red",
254 fg="white",
255 state = DISABLED,
256 height = 1,
257 width = 8)
258 self.stopButton.pack(side = LEFT, anchor = W, padx = 10, pady =
259

```

Figure 2: Code Excerpt of XMCD and XRMS interface.

Figure 3 shows another code excerpt, this time of E5CK interface. Here we see the mechanism of instantiating an independent thread to update furnace temperature on the UI after each 1 second, as described above in “Tkinter Concept” session.

```

231 def callback_curr_temp():
232     try:
233         self._currTemp.set(round(self.furnaceDevice.getValue(),
234 self._currPower.set(round(self.furnaceDevice.getPower(),
235 self._currP.set(round(self.furnaceDevice.getP(),2))
236 self._currI.set(round(self.furnaceDevice.getI(),2))
237 self._currD.set(round(self.furnaceDevice.getD(),2))
238     except (ValueError, TypeError):
239         pass
240     global processRunning
241     # Updates total time
242     try:
243         totalFurnace = self._get_total_program_time()
244         if (self._radioTime.get()==0):
245             totalFurnace = totalFurnace * 60
246         self.Furnace_Table._totalTime.set(str(round(totalFurnac
247     except:
248         pass
249     # And update it every 1 second later
250     self.master.after(1000, callback_curr_temp)
251     # Trigger of callback (point of return)
252     self.master.after(1000, callback_curr_temp)
253

```

Figure 3: Code Excerpt of E5CK interface.

Finally, Figure 4 gives an example of how GUI for DXAS beamline control system solution implemented using Python Tkinter looks like.

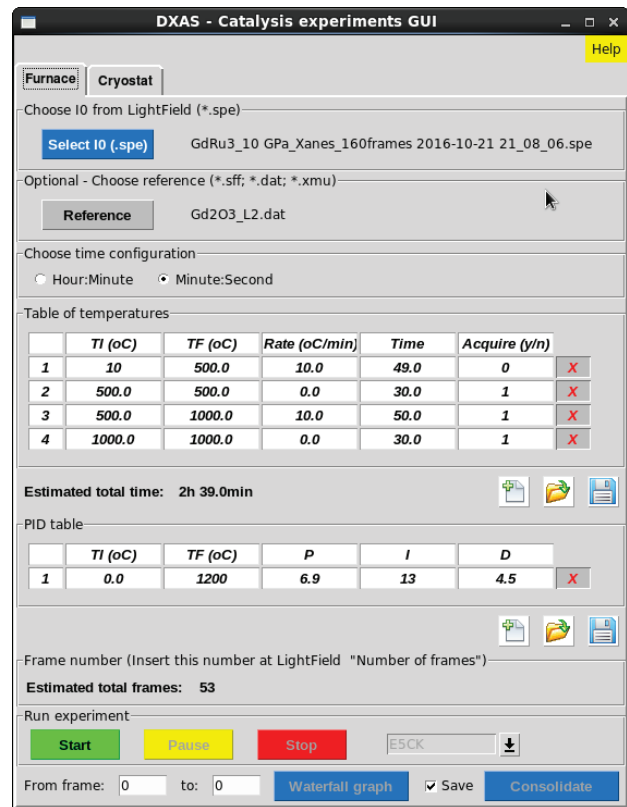


Figure 4: GUI of Catalysis Experiments.

REFERENCES

- [1] Tkinter, <https://wiki.python.org/moin/TkInter>
- [2] DXAS beamline of LNLS, <http://lnls.cnpem.br/beamlines/xafs/beamlines/dxas/>
- [3] Py4Syn, <http://py4syn.readthedocs.io/en/latest/>
- [4] EPICS, <http://www.aps.anl.gov/epics/>
- [5] CA, <http://www.aps.anl.gov/epics/docs/ca.php>
- [6] PyEpics, <http://cars9.uchicago.edu/software/python/pyepics3/>
- [7] NumPy, <http://www.numpy.org/>
- [8] Matplotlib, <http://matplotlib.org/>
- [9] CS-Studio, <http://controlsystemstudio.org/>

DEVELOPMENTS OF THE ‘CERBERUS’ LASER INTERLOCK AND HAZARD DISPLAY SYSTEM AND ASSOCIATED DESIGN TOOL

D. A. Pepler[†], R. Bickerton, A. Tylee, STFC Rutherford Appleton Laboratory, Didcot, UK

Abstract

Following on from the successful implementation of ‘Cerberus’ a comprehensive laser interlock / control and hazard display system [1, 2] on the Vulcan High-Power Laser at the Central Laser Facility (CLF) [3], the last few years have seen the safety system become a CLF standard and its use extended to many different laser systems and laboratories within the department.

This paper will provide an overview of the system, its most recent enhancements and in particular the developments of a design tool and the potential for this system to be used in other fields.

INTRODUCTION

Vulcan is a large, complex and Petawatt class (10^{15} W) neodymium-doped glass laser system. It has a footprint of two Olympic-sized swimming pools with some 250 m of beamlines from its initial suite of optical pulse sources through to a target. The short timescale (10^{-12} to 10^{-9} s) laser pulses pass through many optical components that are situated in a number of rooms which can be kept isolated (and which are then effectively controlled autonomously) or coupled together with remotely-controlled and pneumatically driven shutters that link two or more rooms together. All entry and exit points, laser enclosures and sources and the interconnecting shutters are part of the interlock system. Any potential increase in the level of hazard (e.g. enabling a laser source) requires simultaneous human operator authorisation key action from all affected areas. Reducing the level of hazard, for example by closing a shutter, can be achieved at any time with a single touch. Unauthorised entry or detection of a fault condition will automatically shut the affected areas down rendering them safe by disabling all local hazards and closing any links to other areas.

SYSTEM DESCRIPTION

In Greek mythology Cerberus is generally depicted as a three-headed guard dog and the laser interlock and control system was partially given that name because of its guarding nature and its threefold constituent parts – a control Programmable Logic Controller (PLC), a safety PLC and a Hazard Information Display system.

The control PLC provides the operator with multi-page touchscreen display that allows for the indication and control of various items such as laser hazard, shutters and beam-stops.

The safety PLC conforms to the international standard IEC 61508 [4] at Safety Integrity Level 2 (SIL2) and provides the underlying safety by constantly monitoring the state of the system and either permits or denies a par-

ticular action to take place on the basis of the required and present level of authorisation.

Within each room of the laser facility the Hazard Information Display comprises of a local PC with a dual monitor. At a glance, the first monitor gives operators an overview of the safety state of the whole system and the second monitor shows the specific laser hazards present within the room. Screenshots from these are shown in Fig. 1 below.

The overview screen is located at a rooms central control point whereas the hazard display is also duplicated in multiple locations around the room. Both screens are also duplicated at the entry point with the laser hazard display having a clear green or red colouration to denote whether the system is safe (no laser hazards) or not. For Vulcan, laser hazards would commonly be continuous or pulsed invisible near-IR lasers operating at 1053 nm and which if present require special protective eyewear to be worn.

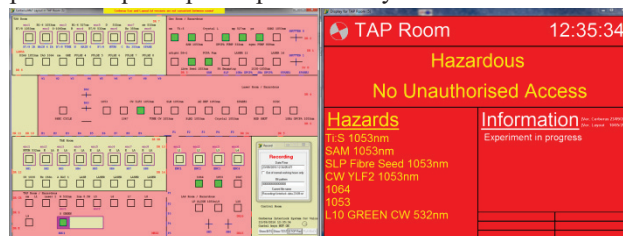


Figure 1: Images from the Hazard Information Displays showing left) the system overview screen, and right) the individual area's laser hazard warning display.

Apart from the display function, the Cerberus application also has the ability to record all activity and changes within the interlock system in a text-based log file. This provides a way of checking for unexpected anomalies or out-of-normal-working-hours activity. This log can also be played back in real or enhanced time.

SYSTEM DEVELOPMENTS

Because all laser hazards are enabled through Cerberus controls and normally would turn the hazard display red demanding that goggles be worn, the inclusion of newer visible (532 nm) alignment lasers that are predominately eye-safe necessitated a separate indication. In the case where *only* eye-safe lasers are in use a new mode was implemented to complete the ‘traffic light’ warning notification. This is demonstrated in Fig. 2 where only a 532 nm laser is enabled and the hazard display is green apart from an orange band displaying the yellow / black flashing message ‘Hazardous at Focus’.

This approach was seen as visually clearer than having the whole screen orange so as to provide good discrimination between orange and red. If a non-eye-safe laser were to be added then the display would revert to the standard red ‘Hazardous’ indication.

[†] dave.pepler@stfc.ac.uk

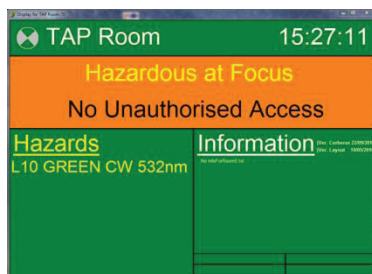


Figure 2: Laser hazard display with an eye-safe laser enabled.

In the original Cerberus application, the local PC's all individually built-up a picture of the system by interrogating every control PLC for itself. This was OK for small systems but as the number of rooms increase there's an exponential rise in ethernet traffic that can result in data collisions as multiple PC's attempt to extract data from the same PLC at the same time causing 'black-outs'. This issue was eliminated entirely by re-working the Cerberus code such that a PLC is only read by the local PC which then broadcasts out the data to the rest of the system via a single UDP packet. This allows for each local PC to be aware of the entire system with minimal effort and to easily update at 5 Hz. It also allows for a ready expansion of the system as well as for passive monitoring by other control systems associated with the operation of the Vulcan Laser.

The application has also been enhanced such that if a local PC 'dies' (which obviously means it is no longer broadcasting its PLC data) then within 1 s one of the other local PCs will take up the task of reading and broadcasting the PLC data instead and will do so in addition to its own. On recovery of the missing PC, it will revert to normal operation within 200 ms.

DESIGN TOOL DEVELOPMENT

A significant number of laser labs within the CLF are now using this application as standard and the differences that are seen in the individual laboratory scale, room connectivity, access points and internal hazards can all be accommodated through the use of a text based configuration file. Everything that is to visually appear on the overview screen or hazard displays (see Fig. 1) requires a record in the configuration file that details its category (e.g. room, door, enclosure, laser, shutter); its name; screen position, colour and controls. For example, a record for an eye-safe laser would look like the following:-

```
Laser 29 Title B1-6_527nm 250 35 11;
Laser 29 Position 270 80 300 110;
Laser 29 Beam 1100 70 1080 70;
Laser 29 InEnclosure 3;
Laser 29 OnBitPosition 3/1/12;
Laser 29 ComingOnBitPosition 3/12/12;
Laser 29 NoHazardInRoom 1 2 4 5 6;
Laser 29 EyeSafe;
```

These descriptions are used by the application to be informed as to where the relevant data comes from (room / word / bit) and where and how to draw the indicators on screen. These descriptors can be readily modified since they are in plain text but as the systems have grown more involved and complex creating a new configuration file from scratch can be a bit tedious.

ISBN 978-3-95450-189-2

In order to assist the rapid construction of new descriptor files there is a design tool being developed that is intended to allow a graphical approach to be taken which will allow an overview screen layout to be created far more quickly and accurately. It will allow any category object (such as a room, laser, door etc.) to be uniquely created and then simply dragged into the overview screen area and dropped into position. The associated record lines for an object will then be processed and where possible, automatically populated according to where an object is placed. For example, a door dropped onto room 3 would be declared to be InRoom 3. All data would be fully editable at any time as clicking on any object would allow it to be moved and its text record to be displayed and updated.

Figure 3 shows the development system with the bulk of the rooms populated with various objects and the entire Room 6 in the process of being relocated. It also shows the information panel open to the right of the main design area indicating the selected objects data – in this case the data for Room 6.

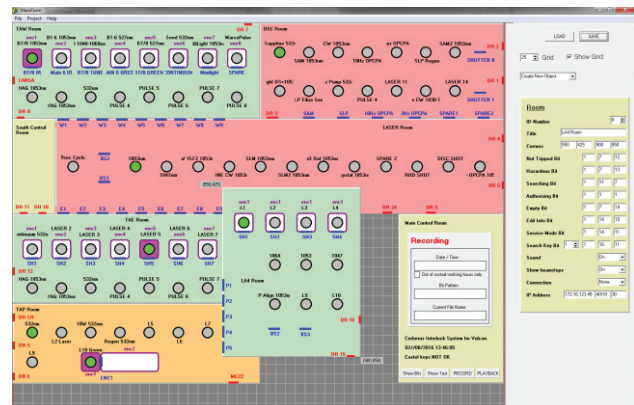


Figure 3: Design tool for the Cerberus overview screen.

CONCLUSION

Once in full production the editing tool will allow easy construction and modification of the system descriptor files. It also will have the potential to modify existing or introduce additional objects to the design suite which could be useful in translating the Cerberus interlock and hazard displays into other fields - for example, for radiation safety access systems.

REFERENCES

- [1] CLF Annual Report 2004-05, "A new interlock system for the Vulcan laser", pp 241-242.
- [2] CLF Annual Report 2006-07, "The Argus/Cerberus interlock system used throughout the CLF", pp 236-237.
- [3] Central Laser Facility, <http://www.clf.stfc.ac.uk>
- [4] "Functional safety of electrical/electronic/programmable electronic safety-related systems", IEC 61508 / BS EN 61508 (7 parts).

OPEN HARDWARE AND COLLABORATION

J. Serrano, CERN, Geneva Switzerland

Abstract

Open Source Hardware (OSHW) follows the lead of Free and Open Source Software (FOSS) and has similar goals: ensuring developers can share their work without artificial hurdles, improving quality through peer review, avoiding vendor lock-in and providing for a fair playground in which projects can thrive and accommodate contributions without compromising their long-term future. The paper introduces OSHW and then attempts to answer a number of questions: (i) what are the perceived benefits and issues of OSHW, in general and in the context of public research facilities? (ii) what is new with respect to FOSS? (iii) what makes OSHW projects succeed or fail? The paper uses real examples of OSHW projects and practice throughout – mostly CERN-related because they are as good as any other and well known by the author – and concludes with some thoughts about what the future holds in this domain.

INTRODUCTION

What is Open Source Hardware? The word “open” is often abused. Some people consider it good marketing to assign this label to a product they have designed, without much concern to what is actually meant by it. Having identified this issue quite early on, the Open Source Hardware Association (OSHWa) published an official definition [1] of Open Source Hardware (OSHW) in 2010. It is inspired by the definition of Open Source Software [2] and the four freedoms of Free Software [3]. As such, it focuses on the freedoms granted to users of OSHW designs, which include the right to study the design documents, modify them, share the modifications, build hardware based on the designs, and commercialize the results. The definition is important insofar as it enables efficient communication by clarifying what OSHW actually means.

With the definition under our belts, let’s move on and discuss the OSHW phenomenon. More and more designers share their design documents on the Web, using licenses which comply with the OSHW definition. Why? Because designing non-trivial hardware has become easier in the last years, opening this domain to a larger subset of the population. Things which are easy and useful to do have a special relationship with freedom. If a world power decided to forbid travel to Jupiter, few people would complain. If, on the other hand, Spanish citizens were suddenly forbidden to cross the French border, there would understandably be strong protests. This is why it is important to guarantee that people who want to share their designs can do so within a robust legal framework which guarantees a number of important freedoms.

As we will see throughout this article, much of what is happening in OSHW and many of the issues we confront in its practice have a direct counterpart in the Free and Open

Source Software (FOSS) world. The democratization of access to computing in the late 80’s and early 90’s explains to a large extent the emergence of Free Software. We see here again the need to provide a solid conceptual and legal setting for a freedom when it becomes easy to exercise by many. FOSS is a reference and a source of inspiration for many OSHW practitioners.

There are many examples to illustrate the revolution in empowerment that has happened in hardware design in the last few years. Consider the design of a GPS watch. It used to require the efforts and talents of a small or medium-size enterprise. Today it can be done by a few friends staying after work over a few months [4], even if they restrict themselves to using only FOSS tools in the development. The availability of simple, powerful, ready-made, modular electronic boards has enabled an even bigger community of integrators. They use hardware from companies such as Adafruit, Sparkfun and Arduino – which have themselves grown to a considerable size – and add simple customizations to generate a variety of innovative products.

For reasons of legitimacy and focus, this article deals mostly with the particular case of electronics development in the framework of big public scientific institutions, but most of its reasoning and conclusions should apply directly elsewhere. The existence of very successful commercial and non-commercial OSHW organizations and projects tells us that this paradigm is useful. We try to see what contexts favor these developments and take a critical look at perceived advantages and disadvantages of OSHW.

PERCEIVED BENEFITS

As one would expect, the perceived benefits of FOSS apply quite clearly to OSHW:

Reuse. People don’t spend their time and money re-inventing the wheel, and those resources then become available to add quality to a project and innovate. Reusing existing designs is also a very powerful risk management technique. Of course, proprietary design can also be reused, but in a necessarily reduced scope. Proprietary designs benefit from fewer pairs of eyes scrutinizing them and finding bugs. They also tend to be more geared to solving a particular problem and are therefore harder to use in a new setting. Finally, in a design reuse scenario, proprietary designs often need dedicated and costly legal efforts to guarantee the licensee is granted a sufficient degree of freedom.

Avoidance of vendor lock-in. All other things being equal, vendor lock-in is never in the interest of the user of a given technology. OSHW makes lock-in virtually impossible. On the other hand, vendor lock-in is often part of the commercial strategy of proprietary design

companies. For some institutions, such as the military, lock-in can even result in concerns about security.

Dissemination. Some public institutions, such as CERN, have it in their official mandate to maximize their impact on society. One important component of this impact is the use of the technologies developed at CERN in different settings, beyond the realm of High Energy Physics research. The invention and dissemination of the HTTP protocol is a paradigmatic case. Inspired by FOSS and by the standard scientific research paradigm, CERN has actually applied these ideals in the domain of experimental data [5] and scientific publications [6]. The extension to OSHW is natural for institutions that want to maximize the dissemination of hardware designs.

Motivation and recognition. This works for individuals as well as private and public institutions. It is often more interesting and motivating to work on technology which others will find useful. Developers normally prefer to work in teams than to work in isolation. OSHW naturally induces collaboration. It is also a great way to gain a reputation based on one's work. For companies, recognition and prestige translate almost immediately into monetary rewards, so they often consider them an important part of their overall commercial strategy.

As we will see later, the materialization of the benefits above does not follow automatically and necessarily from the practice of OSHW. The latter is rather a catalyst for these benefits to happen in an otherwise well-planned, sensible project.

Benefits of OSHW as seen from the perspective of commercial companies depart slightly from those seen in the FOSS world. The standard arguments apply quite well. OSHW provides a level playing field in which big and small commercial actors can compete freely. Proprietary technologies tend to favor winner-takes-all situations because small players trying to innovate cannot capitalize on a rich basis of pre-existing designs and are often confronted with a patent minefield. Patents of course can affect OSHW companies too, but the situation can at least be partially mitigated by the use of OSHW licenses which include a patent license for all downstream licensees, such as the CERN Open Hardware License [7] and the TAPR Open Hardware License [8]. These licenses are also persistent, in the sense that they require that publication of modified design files happens under the terms of the original license. This perpetuates the virtuous circle of sharing.

OSHW is different from FOSS in that hardware manufacturing needs to happen at some point, and that typically requires a bigger investment than that needed to publish software. There are many ways of making money that closely mimic those for FOSS, such as providing support or design services. But ultimately, most OSHW companies will try to manufacture and sell hardware as well. This means that there is often a higher risk because the necessary initial investment is higher. We will look at this issue in more detail in the next section.

One aspect of OSHW that makes it particularly compelling for small companies is the ability to easily bring in extra help in a given domain for which the company does not have a lot of expertise. The author was once confronted with a situation in which a company was late delivering a newly-designed VME-based ADC board. This company excelled in the ADC side but was struggling with the VME interface part, a domain in which the CERN group had lots of experience. It would have been in the interest of all parties to have the client help the provider in this particular part of the design. After a few months of delay, the company replied that they would actually have accepted to open up their design to the customer in order to receive help. But this was not possible because the VME core they were using was also a proprietary black box for them. So things were doubly locked up and there was no way for the CERN team to help the company fix the bug. In total, 8 months were spent on a problem for which a week should have been enough.

If FOSS can serve as a reference, OSHW should also make hiring and retaining talent easier. It is a recurrent problem in physics laboratories that new recruits take a long time to learn the very specific proprietary technologies used to solve a particular problem. When they finally do, they sometimes need to leave because of the end of a contract, and the person replacing them needs to go through a similar cycle. The use of more standard technologies would make it easier to integrate new people and would also make a more compelling case for candidates to apply. Their learning in the lab would be a worthwhile addition to the set of talents they would carry forward to their next job. Now, a standard solution does not *need* to be open source, but if we relate to experience in the software world, we see that FOSS implementations of standard technologies typically gain more traction than proprietary alternatives, for the usual reasons. In addition, these solutions often need to be slightly adapted to serve the particular needs of a laboratory, and that customization is much more natural in an open source scenario.

PERCEIVED ISSUES

Looking at the possible disadvantages of an OSHW paradigm for hardware development and procurement, it is useful to treat the case of users and providers separately. It is difficult to find any disadvantage for users, all other things being equal. However, quick user polls sometimes reveal some degree of reluctance to the concept, which can be broadly categorized as:

- Where does this stop? We are advocating OSHW PCBs, but what about the chips on those PCBs? Are we happy with those being proprietary? The easiness argument applies in this situation. It is not within the reach of many people to design an Application Specific Integrated Circuit (ASIC), so the loss for not making its design accessible is not that great, today. As ASIC design becomes democratized, we will see more open source designs like the lowRISC System-on-Chip (SoC) [9].

Then there is the notion of *commodity*. Some chips have direct pin-compatible replacements. If a chip is a commodity in that sense, the risk of lock-in is greatly reduced and the case for open-sourcing is not as strong. The same can be said of PCBs which are standardized to a large extent at the I/O level, such as PC motherboards.

- Will there be providers for a piece of hardware if I specify I want it to be OSHW? Experience reveals that this fear is largely unfounded. For better or worse, we live in a world of fairly active competition among many small, medium and large enterprises. This means that the chances of not finding a commercial partner, if a user has money to spend, are close to zero. As for many of the issues raised in this article, the explanation is not necessarily linked with OSHW: it's just plain old supply and demand.

Then there are a number of arguments which belong in the expectations management category. It is unreasonable to expect that publishing a design under an OSHW license will allow the designer to control all manufacturing and distribution of hardware based on that design. This is because of the way licenses work: they grant a right to the licensees that they would otherwise *not* have, in exchange for the promise of the licensee to accept a set of conditions from the licensor. The famous GNU Public License (GPL) works that way. Licensees do not have, in principle, the right to distribute a piece of code, or a modified version of it, for which they do not hold the copyright. The licensor grants this right in exchange for a promise by the licensee to distribute the original or modified source under the same licensing terms. There is no law that forbids a person to build hardware based on a published design, and to distribute that hardware later on, at least not in the general case. That means that our possibilities of controlling the distribution are necessarily reduced. OSHW licensing is mainly the licensing of the *design documents*. This is a powerful mechanism for ensuring freedom, but not as powerful as some would expect it to be.

It is also unreasonable to expect the same performance from systems which have received wildly varying amounts of resources for their development. This happens quite often in the software world. An incumbent proprietary technology on which the user has spent millions in license fees gets compared with a FOSS alternative on which the user has not invested at all.

There is no magic in FOSS as there is no magic in proprietary development. Features and quality are introduced by people who should be paid for their work. The real question is what would be the state of the FOSS (or OSHW) alternative had all those millions gone into its development, and whether it makes sense to switch at some point in time and start financing that option. This can represent a prohibitive initial investment in some cases, especially if big amounts of the work of the user have gone into adding features to the proprietary solution and “enriching” its ecosystem, thereby worsening the lock-in case.

The last big family of unreasonable expectations refers to frustration induced by the realization that bug fixes and

improvements to a design don't start automatically flowing into the inboxes of the designers even if they “put it on the Web.” The promise of OSHW not upheld! But, was the design of sufficient interest and usefulness to others? Was it properly documented? Was it easy to extend and contribute back? OSHW in itself does not help answer any of these questions.

To be complete on the issues chapter, we should also describe common concerns from companies which have evolved from a more traditional proprietary paradigm in which the search for big margins is linked with a certain amount of lock-in. The paradigmatic example which proves it is possible to survive and thrive while publishing the complete designs is Arduino. The potential user is offered a great variety of cheaper clones available in online commercial platforms. Many buy them. After all, these are functionally equivalent copies, compatible with the whole software ecosystem that surrounds Arduino. But many people also buy Arduinos designed, manufactured and sold by the original company. The same is true for products from Sparkfun, Adafruit and other vendors. Why is that?

Here we reach an important point of divergence between FOSS and OSHW. It is easy to download a copy of Debian, Ubuntu, Firefox or other successful FOSS products and obviate the donations button in the Web interface. On the other hand, it is impossible to receive an Arduino or one of its clones without paying any money. Pulling out one's wallet is a high-inertia move. That explains why people often choose not to donate to FOSS tools they download. But in OSHW the high inertia must be overcome in any case. So once the wallet is out, what should we do? Choosing the option which will support the OSHW paradigm which gave us this great product to begin with seems easy at that point, provided of course that the extra expense is reasonable. With that extra expense comes peace of mind for things as varied as PCB manufacturing quality, support, nurturing a community or the potential abuse of one's credit card number. The success of these OSHW companies and others proves that this mechanism works. Experience also shows that clones of successful hardware designs are a fact of life, irrespective of whether the design is OSHW. So it would look like business models based on the impossibility of cheap knock-offs are only viable for very high-end devices and companies with a large legal department.

From the perspective of a design group in a lab, it has also been suggested that there is a higher initial investment in OSHW projects, related to the fact that these projects are also a showcase for a given group and so special care is taken to make them high-quality. This typically means a good organization, documentation and quality assurance practices such as the design of one or more test fixtures, the development of test software and various design reviews. In fact, these are all good things to aim for and the fact that OSHW somehow pushes designers in that direction should not be an issue. But in certain situations the extra initial effort may not be justifiable, even in view of larger future benefits. OSHW is not for everyone all the time.

WHAT'S SPECIAL ABOUT PUBLIC INSTITUTIONS?

Public institutions are different from private companies in many aspects. One fundamental difference stems from the underlying interests they are supposed to serve and manifests in their rational economic behaviors. While private companies represent the interests of a selected set of individuals who own the company, a public institution represents the interests of the much larger set of people who form a society, most of whom finance these institutions by paying taxes.

It is expected of public institutions to purchase goods and services from private companies in the course of fulfilling their mission. Doing so in a way that will maximize their positive impact on the society they serve is an extremely important part of their work, and often far from trivial. What may look like a straightforward option from an engineering perspective in terms of cost and schedule might not look so compelling from the vantage point of a tax payer, and vice-versa.

For better or worse, economics is not a hard science, and the maximization of positive impact on society even less so. Should a public laboratory strike a private deal with one or a few companies for the exclusive transfer of a technology? Or should it open it completely, therefore enabling competition from companies both from member and non-member states of the institution? The optimal solution is not always clear-cut, and it is easy to fall prey of a kind of Stockholm syndrome and mistake the interests of individual companies with those of the whole society. Sometimes they are aligned and sometimes not. Keeping this in mind is crucial if designers and technology officers in public institutions are to fulfill their roles efficiently.

The fact that OSHW developed in a public institution is released for the whole world while development cost is only born by a subset of countries or regions can also complicate matters. The fear that companies in these regions are not sufficiently favored is often unfounded. It comes from an over-estimation of the actual value of the design files within a complete project. A close collaboration between the designers in the public center and those in a company can give the latter a considerable competitive advantage, even in a complete OSHW scenario. Many of the most important aspects of a project are not part of the set of released files. These include design intent, ideas that did not work, possible new applications of the technology and an assessment on how best to move forward, to name only a few.

The case of small and medium-size enterprises deserves special mention. Initial investment is often an issue for them. As we have seen, OSHW can lower the barrier for these companies to access a market. In a common scenario, one or more public “tractor” institutions make the necessary initial investment, thereby enabling many commercial actors to develop a technology further. It is important for public institutions to realize the benefits these actions bring about.

Another important aspect is procurement. Similar arguments apply. Should a public institution require that hard-

ware it purchases have its design published as OSHW? If that incurred extra cost, would it be justified by the benefits it would generate for the laboratory, other similar centers and elsewhere in society?

As is often the case, FOSS leads the way. More and more countries are specifying that software purchased by public institutions using a public budget adheres to standard file formats, and sometimes will even specify that the software itself be FOSS. Specifying FOSS for purchases of software development work is even more commonplace. The whole chain of reasoning for these decisions maps entirely onto the hardware development case. It is therefore to be expected that public institutions will specify OSHW more and more for hardware they have developed by external contractors, as a means of maximizing the positive impact of their spending.

One important challenge remains in the domain of publicly-funded scientific research. A modicum of healthy competition among research groups is deemed an important ingredient to maximize the chances of scientific discovery and the development of useful technologies. There is however a risk that these technologies, which are often just useful infrastructure on the way to discoveries, do not get published by a given group out of fear that their competitors will out-perform them in the scientific part, thanks in part to their publicly-released infrastructure. This is effectively how a private company would (legitimately) behave, but with public money. From the perspective of a tax payer, the re-development of basic infrastructure in different research groups is inefficient. But the very survival of a group may in fact depend on such factors. It therefore would seem important to identify counter-productive incentives provided by the funding agencies where applicable and suggest changes in policy to avoid this waste of resources.

RECIPES FOR SUCCESS AND FAILURE

Most of what makes OSHW projects succeed or fail is basically common sense that one could apply to proprietary projects as well: knowing what the problem is one is supposed to be solving, preparing a plan, informing relevant stakeholders about it, gathering a good team of developers, following up progress, documenting, etc. There are however some factors which play a particularly important role in the case of OSHW projects.

When a design team goes open, there is an expectation that other designers will find the project interesting and contribute in different ways. It is important to bear in mind that these developers are offering the project their most valuable asset: their time. In exchange, they will expect to be treated as first-class citizens within the project, and in particular they will expect that their opinions are taken seriously. A team releasing an OSHW project will therefore help the project be successful by keeping an open mind and allowing ideas coming from elsewhere to make their way into the product. This means that there are chances that the product will in the end not be exactly what the original team had imagined, and that should be for the better. Keeping this flexibility is easier

said than done. It is a fact of life that competent developers often have quite a clear idea of how things should be done. But how to react to a situation where two competent developers have different clear ideas? Losing talent unnecessarily is a tragedy for an OSHW project. Flexibility is key in order to find a good balance.

But how will competent developers be attracted to a project to begin with? Often they will have a problem to solve themselves, and if an OSHW project comes close enough and shows flexibility, the case will be compelling enough for them to join. This means that an OSHW project will have more chances of succeeding if the original designers anticipate the fact that it will be used elsewhere. The use of standards and consequent enlargement of the initial scope is a useful way to render a project more neutral as far as applications are concerned. This is for example what the White Rabbit [10] project did. Its original scope was to replace the aging General Machine Timing system in the accelerators at CERN. But basing its infrastructure on standards like Ethernet and the Precise Time Protocol, it has evolved into a general solution to deal with all sorts of problems related to hard real-time distributed controls and data acquisition. A large community of users [11] and developers have found ways of using and improving this technology in many different contexts, inside and outside CERN. The overall savings are phenomenal, and developers enjoy collaborating with a network of very competent and helpful peers.

Another common pattern in successful OSHW (and FOSS) projects is walking a part of the path together with partners whose ultimate interests might not be fully aligned with yours. The perfect example is the Linux kernel. Big companies such as Intel, Google and IBM contribute great amounts of code to the kernel, because they consider it basic infrastructure on which they do not compete. This infrastructure enhances the value of proprietary offerings they support on top of it, on which they compete. Taking this point of view to the extreme, even commercial companies which sell Linux distributions could be seen as adding a proprietary layer (e.g. their internal processes) on top of a GNU/Linux system. Although their ultimate goals and sources of income are not necessarily aligned with those of other kernel developers, their contribution is invaluable to the success of the project. Similarly, in OSHW, a piece of hardware could be an end product for some or a piece of basic infrastructure for others. And the same applies to FOSS tools for hardware design, about which more in the following section.

THE FUTURE: A CALL FOR ACTION

OSHW promises better hardware, more accessible to bigger numbers, an educational experience for developers and a fun way to collaborate and get work done. It also revisits the traditional roles of provider and user, allowing for an array of intermediate profiles and opening up new ways of collaborating. So what will it take for it to be more successful and

pervasive, following the lead of FOSS? In the opinion of this author, two main things: tools and organization.

It is not surprising that the first developments the FOSS community tackled in the 90's were for the basic infrastructure one needs to develop more software: editors, compilers, the C library... FOSS is about sharing, and sharing is severely handicapped when your colleagues cannot open your source code, modify it, compile, test and send it back. In a landscape of incompatible proprietary compilers, the choice of one of them as a standard is arbitrary and sharing of code is highly inefficient. Fortunately for the FOSS people, the skill they needed to solve the problem was what they were best at: software development. And so they proceeded to write these tools and FOSS promptly took off with the great results we know today.

The situation for OSHW is more complicated. The dominant PCB design tools are proprietary. Even the proprietary market is highly fragmented, so the chances are relatively low that the recipient of design files holds a valid license for the tool with which the design was made. FOSS tools exist (KiCad, GEDA and others) but they are still not up to the level of the best proprietary tools in terms of features and quality. Things are complicated by the fact that the skills needed to improve the situation (software development) do not match the skills of most OSHW practitioners (hardware design). These hybrid profiles are difficult to come across, so the evolution to a hardware design scene dominated by FOSS tools is harder than in the case of FOSS development. The role of the big OSHW companies as trend setters will be important in this respect. Olimex [12], with their recent migration to KiCad, shows the way.

Halfway between hardware and software development we find the realm of Field Programmable Gate Array (FPGA) design using Hardware Description Languages (HDL). There is great potential for development in this domain too. A collection of high-quality HDL cores and a set of FOSS simulation and synthesis tools would bring great benefits to society. There, as well, an important amount of work remains to be done. There are very complete VHDL and Verilog simulators (GHDL, Icarus Verilog) but no high-quality mixed-language FOSS simulator. Synthesis efforts have not benefited from the support of the FPGA vendors so far, but the Yosys project [13] has shown the way and more exciting developments are to be expected in the coming years, hopefully with the involvement of FPGA vendors. Effort is also needed on developing appropriate strong and weak copyleft licensing regimes for HDL design. This will allow this domain to benefit fully from the multiplicative effect of copyleft.

How can all this be achieved? The role of the Free Software Foundation and other organizations in the rise of FOSS is well known and acknowledged. We have seen the positive effect of organizing and pooling resources in many cases. Take the example of the SCOAP3 initiative at CERN. This Open Access action has effectively turned the market of scientific publications in the field of High Energy Physics upside down. Institutes hosting researchers put together all

their articles and ask publishers to quote prices for their editing and publishing. In this way, readers have free access to all papers without paying. This maximizes the dissemination of these research results. The overall cost for taxpayers is reduced by pooling all articles together and having publishers compete on editing and publication costs. This revolutionary change in the paradigm for scientific publications was only made possible by institutes getting together and by creating a layer of organization and coordination.

The same should work for OSHW. Organizations like OSHWA and FOSSi [14] have already started taking these roles and working on important aspects such as OSHW certification and a repository for HDL cores [15]. GOSH [16] focuses on the particular case of OSHW for science. Other problems like HDL licensing and PCB development tools could also benefit from the help of these or similar organizations.

OSHW – as FOSS – suffers from difficulties in quantitatively evaluating its impact. The complete freedom given to users automatically implies that tracking their use of a technology can only be done indirectly. Modern research on economics acknowledges the fact that many actions are ultimately beneficial even if the quantitatively formal case is difficult to make for them, and warns against decisions based solely on measurable outcomes. We have already seen how OSHW can make attracting and retaining talent easier for organizations. This is one example of an outcome whose positive impact is difficult to quantify. There are however efforts underway to provide funding agencies with more compelling cases regarding FOSS and OSHW, and a certain degree of organization would also help in this regard.

Ultimately, once all the infrastructure is in place, the flow of hardware designs will be unrestricted. For the reasons explained above, it is quite clear that designers working in public institutions can play a key role in the adoption of OSHW and the production and distribution of vast amounts of OSHW designs. How this will be organized is up to us.

CONCLUSION

The border between hardware and software is becoming ever more blurred. Design entry for hardware happens through CAD programs or text editors in the case of HDL. These “sources” can then take different paths. If they are fed to a simulator along with testbench code, they effectively behave as software. If they are fed to a production line, the end result is a tangible product. The legal world still uses different formalisms for hardware designs and software source code, but the conceptual similarities are undeniable.

Since the advent of FOSS, software developers have benefited from a privileged environment. They can share their work and bring new talent into a project easily. They can learn from each other while working in local or distributed teams, much more easily than they could in the past. The result is better software, more efficiency and more freedom for the users. OSHW brings that to the world of hardware development. However, it needs to overcome the inertia of a

legal, social and economic paradigm based on proprietary design.

In this article we have looked at the pros and cons of transitioning to OSHW, with a special emphasis on design and procurement work carried out in public research laboratories. The aim has been to be more fair than neutral, and to trigger further discussion in laboratories and elsewhere.

ACKNOWLEDGMENTS

The preceding analysis and opinions are necessarily the result of countless discussions with many colleagues and friends, and also considerable amounts of reading, watching and thinking. There is also a fair dose of practical experience at work. Trying to thank everybody here would be futile, but a few poor souls deserve special mention because they read this long article and helped me make it better: David Cobas, Mick Draper, Pietari Kauttu, Rafael Pezzi, Eduardo Ros, Daniel Tavares and Federico Vaga. Thank you all.

REFERENCES

- [1] Open Source Hardware Definition, <http://www.oshwa.org/definition/>
- [2] Open Source Definition, <https://opensource.org/osd>
- [3] The four freedoms of Free Software, <https://www.gnu.org/philosophy/free-sw.en.html>
- [4] The F-Watch, <http://www.ohwr.org/projects/f-watch/wiki>
- [5] Open Data at CERN, <http://opendata.cern.ch>
- [6] Open Access at CERN, <http://library.web.cern.ch/OpenAccess>
- [7] CERN Open Hardware License, <http://www.ohwr.org/projects/cernohl/wiki>
- [8] TAPR Open Hardware License, <https://www.tapr.org/ohl.html>
- [9] lowRISC, a fully open-source, Linux-capable RISC-V-based SoC, <http://www.lowrisc.org/>
- [10] The White Rabbit project, <http://www.ohwr.org/projects/white-rabbit/wiki/>
- [11] White Rabbit users that the WR development team knows of, <http://www.ohwr.org/projects/white-rabbit/wiki/WRUsers>
- [12] Olimex and KiCad, <https://olimex.wordpress.com/tag/kicad/>
- [13] Yosys Open SYnthesis Suite, <http://www.clifford.at/yosys/>
- [14] The Free and Open Source Silicon Foundation, <http://fossi-foundation.org/>
- [15] LibreCores, <https://www.librecores.org/>
- [16] Gathering for Open Science Hardware, <http://openhardware.science/>

PandABox: A MULTIPURPOSE PLATFORM ADAPTED FOR MULTITECHNIQUE SCANNING AND FEEDBACK

Yves-Marie Abiven[†], Jérôme Bisou, Guillaume Renaud, Frédéric Ta
Synchrotron SOLEIL, Gif-sur-Yvette, France

Isa Servan Uzun, Michael Abbott, Andrew Cousin, Tom Cobb
Diamond Light Source, Oxfordshire, United Kingdom

Abstract

Synchrotron SOLEIL and Diamond Light Source are two third generation light sources located respectively in France and the UK. In 2015, both facilities started the collaboration project “PandA” to overcome technical limitations of SPIETBOX at SOLEIL and Zebra at Diamond as well as to manage obsolescence of the products. The collaboration enables both institutes to share the technical leadership on hardware, firmware and software developments. The initial objective is to achieve multi-channel encoder processing to synchronize motion systems and data acquisition during experiments addressing simultaneous and multi-technique scanning. However, its design based on Xilinx Zynq SoC is thought to be powerful and modular in terms of firmware as well as for hardware. This flexibility permits envisaging derivative applications and interfacing to different third party hardware. This paper details the organization of this collaboration, status of the ongoing project in terms of hardware and firmware capabilities and the results of the first tests at both sites.

PROJECT STATUS

The first step on the project confirmed that the organisation of the project was efficient, sharing leadership between institutes. After few months agreeing specifications in term of hardware and firmware, the last 10 months have been intense, focusing on the design. On one side SOLEIL designed the 3 main boards which are integrated in a 19” rack. On the other side Diamond developed the firmware and software to interface PandABox to EPICS or TANGO control system, up to being able to configure the system and make acquisition. At each step of the design both institutes worked together towards reviewing and validating designs. The First prototype presented in Figure 1 was delivered in August. The prototype hardware was tested successfully along with initial firmware and software releases at both sites during September, and only a few hardware modifications were identified. Update of the electronic and mechanic designs were completed in October and second prototypes were ordered with expected delivery in January.

On the firmware side, Diamond delivered first version of the firmware allowing both institutes to start hardware tests: Processor, FPGA, SFP, FMC and all inputs and outputs. Communication via a TCP server permits to remotely work with the platform: Reading on-board sen-

sors, setting and acquiring encoder position or generating triggers on TTL outputs. All standard configurations required for first application are available. Today the functional tests of the firmware are ongoing and already promising.

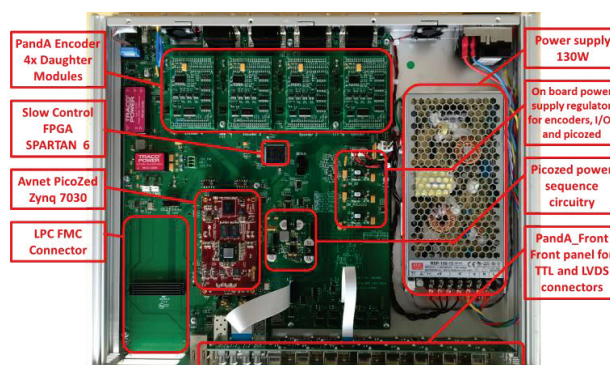


Figure 1: PandABox inside view.

HARDWARE ARCHITECTURE

The hardware architecture is designed to meet the requirements for simultaneous and multi-technique scanning with support for a wide range of encoder protocols. The architecture presented in Figure 2 was developed to be modular and flexible in order to be open for a maximum numbers of applications. The platform is built with:

I/O interfaces

- Multi-Channel TTL and LVDS I/Os for synchronous triggering and clocking. 4-Channels of encoder interface I/Os, supporting Incremental, Absolute encoder protocol (AquadB, SSI, EnDat and BiSS). Integrated on the platform as a mezzanine boards allowing this functionality to be updated.
- A fully compliant Low-Pin Count FMC slot for interfacing to analog and digital off-the-shelf boards or custom I/O modules.
- JTAG allowing programming and debugging for Zynq and Spartan-6 during prototyping.
- Serial port, giving a terminal for linux administration.

Communication interfaces

- 3-Channels of SFP Gigabit Transceiver interface for UDP triggering, Timing System, Diamond Communication Controller or custom high-speed serial connectivity.
- A Gigabit Ethernet connectivity for control systems integration and high-speed data acquisition.

[†] yves-marie.abiven@synchrotron-soleil.fr

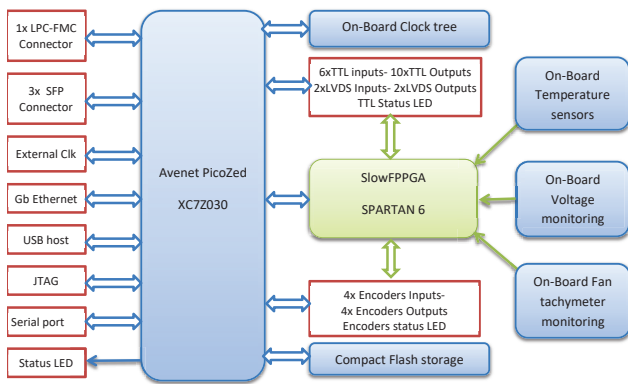


Figure 2: PandABox architecture.

Control and data processing

- A Slow Control FPGA to control the configurations of the I/O encoder signals specific to encoder type, TTL inputs impedance and some status LEDs on front panel. It also interfaces to on-board temperature, voltage and fan-speed sensors for monitoring purposes.
- The main processor, SoC Zynq 7030, based on an off-the-shelf product AVNET PicoZed [1] for tighter system integration. It is the heart of the system, managing data acquisition, data processing and data exchange with control system.

Additional functionalities

- USB Host, permitting to connect USB stick to upgrade the firmware is available in the rack

Hardware Design

PandABox is develop in a 1U 19" rack which integrates four boards: a carrier board, a front panel, 4x encoder daughter boards and the PicoZed SoC mezzanine.

Panda Front and back panels It was designed giving access to all triggering capabilities (TTL, LVDS I/Os), communication interfaces (SFP, Gbe) and an optional FMC as show on Figure 3. The connection to the carrier board is done through 2x FFC cables. The encoders I/O as well as the auxiliary functionalities, such as USB, JTAG and serial port are implemented on the carrier board and accessible from the back-panel, as shown on Figure 4.



Figure 3: PandABox Front Panel.



Figure 4: PandABox Back Panel.

- Panda Encoder Daughter Board provides interfacing to industry standard encoders for the platform. The rack can be equipped with four boards.
- PicoZed System-On-Module is the backbone of the system. It embeds a Dual-core ARM Cortex-A9 and an Artix 7 FPGA. On one side the ARM processor runs a TCP server on an embedded Linux OS which provides the control system. On the FPGA side, encoders' positions are processed for triggering and data acquisition as detailed in Figure 7.

Panda carrier boards Inside the rack, Panda carrier board is an 8-layers PCB designed to connect all the boards together. It embeds all the power supply circuitry to regulate the voltages required for Picozed and I/Os up to 6 different voltages from 1V to 24V and powered by an external 12V. An on-board $\pm 15V$ is available for FMC boards requiring this external power supply for DAC or ADC based applications.

The Slow Control FPGA has been centrally positioned on this board to facilitate routing of the functions it provides:

- Five temperature sensors distributed on the carrier board and monitored via an I2C bus as well as the separate voltage monitoring.
- Panda Encoder daughter boards, through the control lines which configured the logic circuitry for AquadB or serial communication for the type of encoder connected to a dedicated channel. It allows Incremental, SSI, BISS or Endat encoder protocols.
- The configuration of the Panda front panel, High-Z or 50-Ohm impedance for the TTL Inputs, status LEDs. All the control is done through shift register connected in I2C.

The four encoder interfaces have been designed to allow modularity of this function. The stacked 15-way D-type connectors used for RS422 encoders are mounted on the carrier and connected to the encoder daughter board connectors. Connector's gender modification is possible since the adaptation is done on the daughter board. This approach permits each institute to maintain its pinout. Concerning the daughter boards, today it is designed with buffers to interface the different encoder types, but can be re-designed for new custom functionalities (ie. serial DAC, ADC interfacing).

A special care has been taken for routing the differential pairs and clocks, industrial standard fully-compliant Low Pin Count (LPC) FPGA Mezzanine Connector (FMC), the Gb/s Ethernet and the MGTs. These include pair matching and length tuning to the Picozed connectors. Concerning the FMC module, a specific space behind the connectors has been kept free for use of ELF specific board [2].

At the heart of the system architecture is the PicoZed System-On-Module. The PicoZed module contains Xilinx Zynq-7030 All Programmable (AP) SoC and the common functions required to support the most SoC designs, including memory, configuration, Ethernet, and clocks. It

provides easy access to over 100 user I/O pins through three I/O connectors on the rear of the module.

Prototype Validation

The first prototype is available at both institutes and fully tested. After correcting few design and manufacturing errors, electrical tests were successfully carried out. The Picozed quickly boots up and the boot sequence can be monitored through a terminal on the UART. For the slow control FPGA, during initial tests, the FPGA was configured from on-board PROM. Now, it is programmable via the Zynq MIOs, taking about 11 seconds. Once running, and configured on the network, communication with the platform is functional over the TCP server. All digital TTL and LVDS I/O on the front end as well as the encoder inputs and outputs on the rear panel are fully operational.

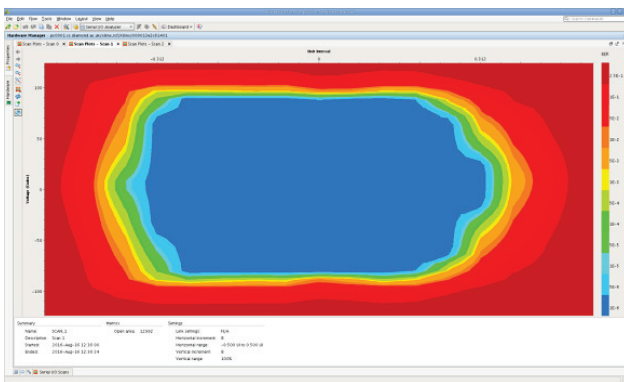


Figure 5: Eye Scan eye using with IBERT core. Line rate of 6.25Gbps with 550m fibre loopback patch.

Concerning SFP connectors, the Gigabits transceiver were tested with loopback using IBERT (Integrated Bit Error Ratio Tester) from Xilinx. This IP has been specifically implemented for testing purposes. SFP and GTX are validated at 6.25Gbps & 2.5Gbps using GTX-CLK0@125MHz with up to 550m fibre, giving acceptable eye scan result as shown on Figure 5.

A similar testing for the FMC connected were carried out using a dedicated firmware functional block. The test validates all FMC I/O, GTX connectivity and clocking pins.

FIRMWARE ARCHITECTURE

Figure 6 below depicts the system architecture for PandaBox firmware and software. Linux OS running on the ARM processor manages the following peripherals:

- QSPI flash contains persistent configuration.
- SD card contains rest of Linux file system.
- Network interface to system.
- Programs FPGA, manages software interface.

The firmware is structured into numerous Functional Blocks (FBs) on the Programmable Logic (PL) part of the Zynq device, with each block configured via a number of registers. Processing System (PS) on the same Zynq device is used to realise required software architecture includes control system interfaces as shown in Figure 7.

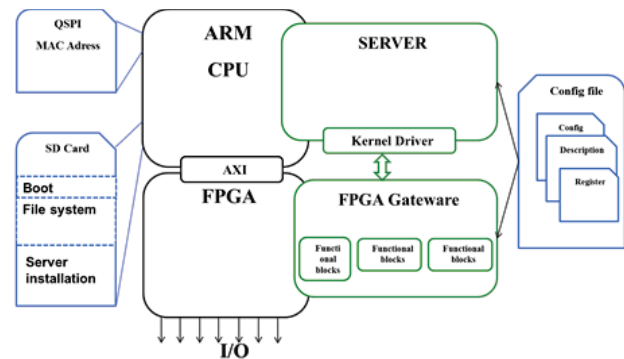


Figure 6: PandaBox overall system architecture for firmware and software.

Flexibility of the architecture is achieved through a set of Config files which describe each Functional Block's input and output ports, configuration registers and descriptions. This approach allows to design and compile a custom set of Functional Blocks into the firmware with access from the TCP Server.

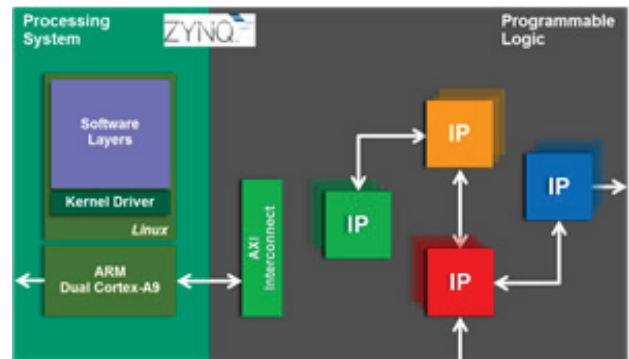


Figure 7: PandaA firmware flexible architecture overview.

Functional Blocks

The list of functional blocks implemented includes:

- Function generators to implement any 5-input Boolean function.
- Set/Reset Gate blocks with configurable inputs, and an option to force its output independently.
- 32-bit Pulse divider with programmable divisor and two pulse outputs (divided and non-divided).
- Pulse Generator blocks to produce configurable width and delayed pulses triggered by its input and also able to handle pulse trains.
- Sequencer blocks to perform automatic execution of sequenced frames to produce timing signals.

Similarly, a wide range of position based blocks are defined:

- Encoder Input/Output blocks supporting multiple encoder protocols.
- Quadrature Input/Output blocks to enable of any discrete signal on the system for quadrature operations.

- Counter/Timer blocks to implement multi-channel 32-bit up/down counters synchronous to system clock.
- Analog input/outputs up to 8-channels through FMC module connectivity.

Position Compare and Capture

Four (4x) independent Position Compare (PCOMP) blocks are implemented to generate a stream of trigger pulses, which can be position based, time based, analog input based or externally triggered. A new feature included to the position compare functionality is the ability to use LUT-based arbitrary arrays of position or time points instead of regularly spaced series of values.

This facilitates:

- Position based capture – where output pulses are generated when the encoder position reaches a certain value, or series of values.
- Time based capture – where output pulses are generated at regular time intervals, storing the encoder positions as well.
- Analog input based capture – where output pulses are generated when analog input position reaches a certain value, or series of values.
- External trigger capture – where the motion controller stores the encoder position on an external trigger signal.

By cascading multiple position capture blocks, it is possible to mix modes together, such as outputting a time based pulse stream within a series of position based gates to support continuous scanning experiments.

A central Position Capture (PCAP) block is responsible for capturing user-defined fields across the design including timestamps, encoder values, analog input values, counters and system status on trigger events.

Functional Blocks' Connectivity

To achieve connectivity between all FBs, as well as the physical inputs and outputs, two internal buses are defined. The System Bus is a 128-bit wide register is composed of concatenating all physical I/Os, and registered discrete outputs from logic and position-based blocks. Similarly, all 32-bit outputs of the position based blocks are concatenated together to create the 1024-bit wide (32x 32-bit) Position Bus.

By connecting the input of any logic block to the system bus via a 128x1-bit multiplexer, it becomes possible to cascade multiple blocks to achieve the desired functionality.

Each physical output is also taken from the system bus in the same way (see Fig. 8). Position Bus feeds all the 32-bit position inputs of all position compare blocks using 32x32-bit multiplexer so that position compare processing can be done on encoder, timer or analog input values.

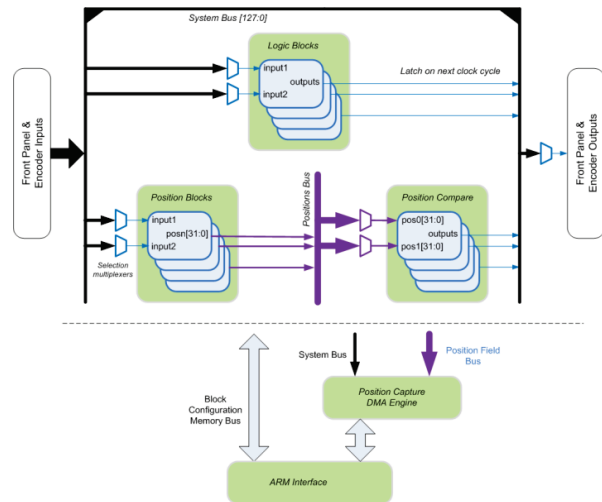


Figure 8: System and Position Bus enables full connectivity among FBs during runtime.

SOFTWARE ARCHITECTURE

Figure 9 depicts standalone software architecture, built on top of firmware specific kernel device driver, running on Zynq PS.

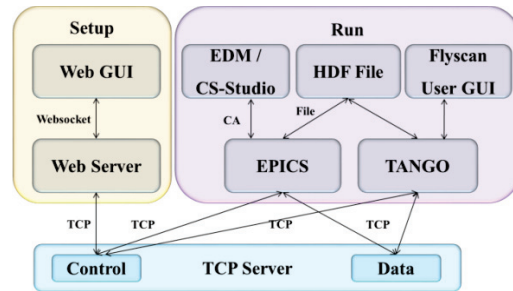


Figure 9: PandaBox software architecture.

Kernel Driver

The kernel driver provides low-level interfacing between the firmware FBs and upper software layers through the TCP server. Its firmware specific implementation includes:

- A generic register address space for direct access to each firmware FBs' configuration and status registers.
- A Read DMA engine to receive position capture from PCAP block in real-time.
- A Write DMA engine for writing user-defined tables to PCOMP blocks for position compare operation.

Panda TCP Socket Server

The Panda socket server provides a bridge between the register interface to the FPGA firmware controlling the hardware and other users software. The interface provided by this server is designed to be simple and robust. The structure of PandaBox firmware is directly reflected in the functional interface provided by this server: most commands read or write specific fields.

The socket server publishes two socket end points, one for configuration control, the other for streamed data capture. The configuration control socket accepts simple ASCII commands and returns all data in readable ASCII format. The data capture socket supports no commands and simply streams captured data in a lightly structured binary format.

Webserver and Web GUI

Building on top of the TCP server, a webserver currently under development provides configuration level access to a PandABox via a web GUI. It is written in Python using the pymalcolm [3] framework, and exposes all blocks and their fields by introspecting what is available on the TCP server. It serves up a React [4] based user interface which communicates to the webserver using a JSON over websocket protocol. A key part of this user interface is being able to visualise and “rewire” connections between blocks, as well as setting parameters.

Once a PandABox is configured, the runtime control portion normally consists of “Arming” and then listening for data on the TCP server’s data port with EPICS or TANGO software.



Figure 10: Web GUI interface showing an example application design with FBs.

EPICS and Tango

For EPICS and TANGO interface; it is envisaged to implement fixed block setup for common use cases and areaDetector driver or device server connected to DATA port to capture data and write to HDF file.

FIRST TESTS AND NEXT STEPS

First Tests

Integration of the prototype units have been under progress at both SOLEIL and DIAMOND to exercise its robustness. At Diamond, a complete hardware test setup was established in the lab including GeoBrick motion controller, X, Y and Z-stages (motors and encoders) and a PandABox under control of Mapping software framework [5]. The real-world application for Beamline I08, as shown in Figure 10, is being tested in terms of logic functionality and data capture. At SOLEIL, PandABox was successfully tested on the SOLEIL Nanoscopium beamline, replacing the SPIETBOX classically used in the

Flyscan [6] architecture. A 1000x1000 continuous 2D scan was performed in a bidirectional (raster ‘zigzag’) mode as shown in Figure 11.

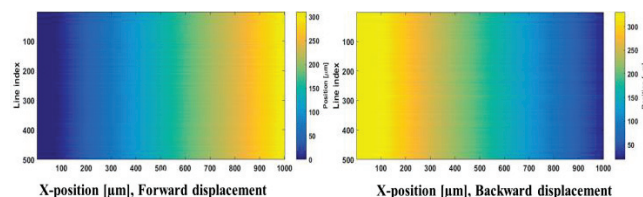


Figure 11: Plot of the 1000x1000 scan, plotting back and forth displacement on a single line, showing repetability in both direction.

Future Firmware FB Upgrades

PandA FPGA firmware architecture enables adding new FBs into the design a straight forward process. Following future FBs are defined to be added to PandA capabilities:

- A 4 Input/4 Output Channel 24V IO FMC module is currently under development, to be used to interface GeoBrick motion controller.
- Micro-Research Event Receiver FB is going to be integrated to recover timing events and clocks distributed across the event network.
- Simple, reliable, UDP/IP FB is to be implemented for triggering external instruments.

Next steps will be the deployment and commissioning of PandABox on beamlines for mapping and flyscan applications.

REFERENCES

- [1] Avnet, Picozed Product Brief, <http://zedboard.org>
- [2] D-TACQ Solutions, <http://www.d-tacq.com>
- [3] "Github: pymalcolm", <https://github.com/dls-controls/pymalcolm>.
- [4] "React: A JavaScript library for building user interfaces", <https://facebook.github.io/react/>.
- [5] R. D. Walton, “Mapping Developments at Diamond”, in *Proc. ICALEPCS’15*, Melbourne, Australia, Oct. 2015, paper THHB3001.
- [6] N. Leclercq et al., “Flyscan: a Fast, Simultaneous and Multitechnique Data Acquisition Platform for the SOLEIL Beamlines”, in *Proc. ICALEPCS’15*, Melbourne, Australia, Oct. 2015, paper WEPGF056.

NEW CONTROLS PLATFORM FOR SLAC HIGH-PERFORMANCE SYSTEMS*

Till Straumann, R. Claus, J. M. D'Ewart, J.C. Frisch, G. Haller, R.T. Herbst, B. Hong, U. Legat, L. Ma, J.J. Olsen, B.A. Reese, L. Ruckman, L. Sapozhnikov, S.R. Smith, J. Vasquez, M. Weaver, E. Williams, C. Xu, A. Young, SLAC, Menlo Park, California, USA

Abstract

The 1 MHz beam rate of LCLS-2 precludes the use of a traditional software solution for controls of "high-performance systems" which operate at this rate, such as BPMs, LLRF or MPS. Critical algorithms are ported into FPGA logic and administered by ordinary PCs via commodity Ethernet. SLAC has developed a controls architecture which is based on FPGA technology interconnected by 10G Ethernet and commercially available ATCA shelves. A custom ATCA carrier board hosting an FPGA, memory and other resources provides a "common platform" for many applications which can be implemented on AMC cards which are plugged into the carrier. A library of firmware modules including e.g., timing, history buffers and reliable network communication together with corresponding software packages complement the common platform hardware and provide a standardized environment which can be employed for a variety of high-performance applications across the laboratory.

INTRODUCTION

The LCLS-2 XFEL is currently under construction at SLAC. A superconducting linac supplies electron bunches at a rate of up to 1 MHz. Several diagnostics, controls and protection systems must be able to resolve individual bunches and process data in real-time. A conventional control system based on computers with peripheral devices and software based algorithms is – at the current state of the art – not capable of meeting the necessary throughput and latency requirements (considering that the actual data rate some subsystems have to handle is much higher than the beam rate of 1 MHz).

Therefore, the bulk of processing must be implemented in programmable logic, leveraging a high level of parallelism in hardware. SLAC has developed a new platform which is based on the following technologies:

- ATCA [1] form-factor and commercial shelves.
- IPMI management [1].
- 10 G Ethernet communication.
- FPGA on generic ATCA carrier.
- Application-specific AMC cards.

An library of firmware components and a "common-platform" software framework complement this versatile and powerful instrumentation and controls platform.

* Work supported by the US Department of Energy, Office of Science under contract DE-AC02-76SF00515

Since the hardware design has already been presented elsewhere [2,3] this paper shall focus on the communication protocols and software framework.

HARDWARE OVERVIEW

The platform employs a COTS ATCA shelf that provides standardized mechanical support, power, cooling, management and interconnect. The backplane's fabric interface is configured in a dual-star topology supporting up to 10 Gbps per channel (with a 40 Gbps upgrade path). A block diagram of the system is depicted in Fig. 1.

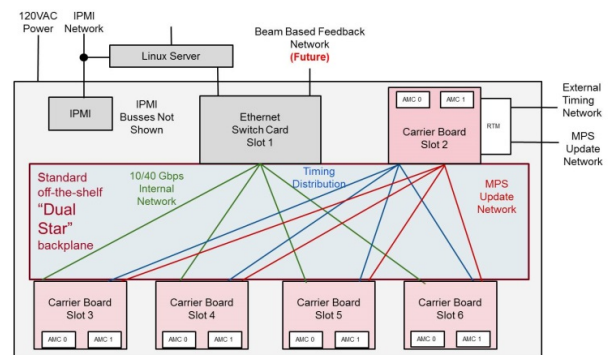


Figure 1: Common Platform System Diagram.

One star is concentrated at a 10 Gb Ethernet switch that provides the main connectivity between ATCA boards as well as an external Linux server computer (Fig. 1) which is responsible for interacting with the firmware.

The second star is used to broadcast timing data and to gather MPS (machine-protection) information that is further aggregated by the custom general-purpose carrier board (as described in the next paragraph) in slot 2'. Timing and MPS connectivity to the outside is provided by a custom RTM.

One core element is the custom carrier board that hosts a Xilinx Kintex Ultrascale XCKU040 or -060 FPGA and 8 GB of DDR3 memory. The FPGA's SerDes interfaces are connected to the Fabric interface as well as four AMC bays. These bays can receive single- or dual-wide, full-height AMCs. Additional LVDS and high-speed I/O is available via the zone-3 connector to RTMs.

A variety of AMC cards (commercial and custom) can be plugged into the carrier. A common use case are high-speed ADCs for applications like BPMs or bunch-length monitors. Placing analog components on well-shielded AMC cards ensures that EMI effects can be minimized.

Firmware Library

Many applications require a large set of complex, common functionality such as DDR-memory, JESD, Ethernet communication, timing-system support, MPS uplink, diagnostics, etc. These components are supported by an extensive library of firmware modules (Fig. 2), thus greatly simplifying the task of an application developer.

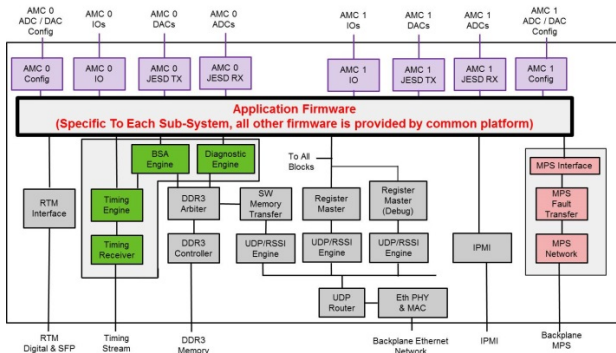


Figure 2: Common Platform Firmware.

COMMUNICATION PROTOCOLS

Communication between firm- and software is achieved by means of a stack of protocols as shown in Fig. 3. On top of Ethernet there is a layer of pseudo-IP/UDP. This is (in firmware) a non-standard version of the widely-known IP protocol that supports just enough functionality (e.g., no fragmentation or IP options, no ICMP, etc.) to interoperate with a standard network stack.

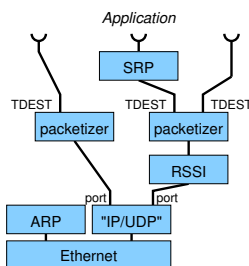


Figure 3: Protocol Stack; some layers are optional (e.g., no RSSI at LHS UDP port).

The *RSSI* layer implements a reliable transport. It is based on [4–6] with minor deviations, one of which adds a flow-control feature. RSSI provides functionality similar to TCP but is much simpler and adequate in the high-performance, very reliable, switched LAN environment of the common platform’s ATCA backplane.

The next, “packetizer”, layer handles datagram fragmentation to allow datagrams bigger than the Ethernet MTU. Note the inverted arrangement compared with TCP/IP where fragmentation is handled *below* reliability. Obviously, the task of reassembling large datagrams is much easier to do and more efficient when layered on top of reliable delivery.

The packetizer header features a so-called “TDEST” ID (having origin in the AXI [7] technology which is widely used in firmware). TDEST serves a similar purpose as UDP port numbers do, i.e., datagrams can be multiplexed to/from different endpoints based on TDEST.

The top layer is the “SLAC Register Protocol” (SRP) which defines simple read- and write- operations to addressable entities such as registers or memory. The use of SRP is optional, i.e., there are use-cases where raw datagrams (e.g., “events” or “streams”) are delivered to or received from the application.

With TDEST multiplexing it is possible to share a UDP/RSSI channel between multiple endpoints which may or may not use SRP.

In software, the protocol layers are handled by *protocol modules* which are assembled into a “stack” using the so-called “builder-API” (see below).

COMMON PLATFORM SOFTWARE

While all high-performance operations which require a high, sustained throughput and/or very low latency are implemented in firmware there are still plenty of tasks left for implementation in software: configuration, slow monitoring, diagnostics, etc. A Linux server communicates with firmware over 10 GbE (Fig. 1) and acts as a gateway to a traditional control system such as EPICS. However, rather than directly interfacing the control system to the firmware, an abstraction layer, the “Common Platform Software” (CPSW) was designed.

- Provide abstracted access to firmware entities (mostly: device registers) hiding details of Ethernet communication, endianness, offsets, etc.
- Present firmware entities as a hierarchical name space (similar to Linux’ sysfs).
- Offers a standardized interface to firmware. Can be accessed e.g., from python (development) and from EPICS (production).

CPSW APIs

An important design goal for CPSW was a clear separation of APIs from implementation and partitioning of APIs according to their core functionality. CPSW is written in C++ and APIs (except for the “developer-API”) are defined as abstract classes. Access to an API and the objects it defines is obtained via factories which produce “smart” shared pointers. This approach alleviates the user from having to manage object lifetime explicitly and completely encapsulates implementation details. There are three layers of APIs:

- “User-API”. All interactions of applications with the firmware use this API. It offers “lookup” and “access” operations in a hierarchical namespace (e.g., “find a register”, “get value”).

- “Builder-API”. This API is used to create the hierarchy of object instances which represents the entities implemented in firmware. It uses abstract representations of basic building blocks (“memory-mapped container”, “register”).
- “Developer-API”. This API exposes the headers of the underlying classes and allows existing classes to be extended, etc. It is used to implement new building blocks that can be instantiated via the builder-API and accessed via the user-API.

The user- and builder- APIs are independent views of the underlying classes and are abstract. This guarantees that implementation details can be changed without affecting an application. The separation between user- and builder- API allows for a complete separation of the application proper that accesses and manipulates firmware from the code which constructs the tree of drivers. I.e., when the layout of a firmware register changes then only the “build” process needs to be modified but the core application is guaranteed to be unaffected.

User-API If a user wants to manipulate e.g., a firmware register then the corresponding entry is first located by name (similar to a directory lookup; blue arrows in Fig. 4) yielding a “Path” handle. Access to the functionality offered by an item is performed via a so-called *interface*. A CPSW *interface* is also an abstract class that offers access methods (e.g., “get integer value”). The user attempts to “create” or “open” an interface to the target entry from the Path handle. If the underlying class indeed implements the desired interface then a smart pointer to the interface is passed to the user (green arrow in Fig. 4); otherwise an exception is raised. The user may then go on and access the interface.

Basic interfaces include “ScalVal” (get/set integer value), “DoubleVal” (get/set floating-point value), “Command” (execute an operation).

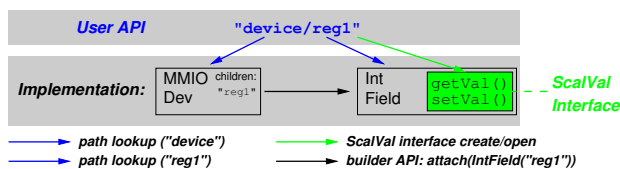


Figure 4: CPSW User-API and Implementation Layers.

YAML Firmware/Hardware Description

Traditionally, a driver-writer often has to extract detailed information about a device (such as register offsets, bitmasks, etc.) from a manual and translate it into definitions which would typically be coded into a header file. This process is cumbersome and error-prone.

The separation between user- and builder- API was also driven by the desire to automate these steps. This is accomplished by the firmware engineer providing device descriptions in the *YAML* [8] data serialization format which meets

our requirement of a standardized, human- and machine-readable interchange format. See Fig. 5 for an example.

```
device:
  class: MMIODevice
  children:
    reg1:
      class: IntField
      sizeBits: 32
      at: { offset: 0x00 }
```

Figure 5: Excerpt YAML Description of Example in Fig. 4.

Larger systems are assembled from device descriptions and result in a *YAML* file which defines the complete hierarchy implemented in firmware.

A CPSW application can then simply load a *YAML* description without the need to explicitly use the builder-API. Any firmware modification (as long as it doesn’t directly affect functionality “seen” by the user-API) can then be handled in a completely transparent way – provided it is properly reflected in the *YAML* description.

CPSW Extensions and Dynamic Loading

CPSW supports dynamic loading of driver classes making it possible to extend the core functionality in a modular way.

Python Bindings

Bindings for Python2.7 and Python3 are provided.

CONCLUSION

The new platform supports a lot of common functionality at the hard- firm- and software level which can be leveraged by specialized high- performance systems in order to reduce development time and cost as well as simplify maintenance.

REFERENCES

- [1] PICMG, “AdvancedTCA Overview,” <https://www.picmg.org/openstandards/advancedtca/>
- [2] J. Frisch *et al.*, “A FPGA Based Common Platform for LCLS2 Beam Diagnostics and Controls,” in *Proc. IBIC’16*, Barcelona, Sep. 2016, paper WEBPG15.
- [3] R. Herbst, “ATCA Based Accelerator Controls & Detector Platform”, EPICS Collaboration Meeting, Oak Ridge, September 2016, <https://conference.sns.gov/event/11/session/1/contribution/39>
- [4] D. Velten, R. Hinden, and J. Sax, “Reliable Data Protocol,” July 1984, <https://tools.ietf.org/html/rfc908>
- [5] C. Partridge and R. Hinden, “Version 2 of the Reliable Data Protocol (RDP),” April 1990, <https://tools.ietf.org/html/rfc1151>
- [6] T. Bova and T. Krivoruchka, “Reliable UDP Protocol,” *DRAFT*, February 1999, <https://tools.ietf.org/html/draft-ietf-sigtran-reliable-udp-00>
- [7] ARM Ltd., “AMBA Documentation,” <http://infocenter.arm.com/help/topic/com.arm.doc.set.amba/index.html>
- [8] O. Ben-Kiki, C. Evans, and I.d Net, “YAML Ain’t Markup Language Version 1.2,” <http://www.yaml.org/spec/1.2/spec.html>

OPEN HARDWARE EXPERIENCE ON LNLS' BEAM DIAGNOSTICS

G. B. M. Bruno*, D. O. Tavares, H. A. Silva, F. C. Sant'Anna,
J. L. Brito Neto, L. M. Russo, L. A. Martins, S. R. Marques
LNLS, Campinas, Brazil

Abstract

LNLS Beam Diagnostics Group has decided to adopt and develop open-hardware technologies for most of its projects, partnering with other institutes and companies to design and build a complete RF BPM electronics, from the BPM pick-up to the operator interface. That decision resulted in advancements and learning, bringing new technologies, flexibility and knowledge, but also brought some hardships and new challenges. This paper details the history, advantages and difficulties of this open hardware approach.

INTRODUCTION

Sirius is the new synchrotron light source currently being built by LNLS in Brazil, with 518.4 m circumference and expected to be installed in late 2017, for commissioning in mid 2018 [1]. A total of 255 units of in-house-developed BPM electronics will be built in order provide feature-rich data acquisition system for button and stripline BPM pick-ups of the storage ring, booster and booster to storage ring transfer line.

The Sirius digital BPM electronics consists of 4 main pieces of hardware: a standalone RF front-end electronics for analog signal conditioning (RFFE), a digitizer mezzanine card for analog-to-digital conversion (FMC ADC), a FPGA-based carrier board acting as a digital back-end for acquisition, signal processing and communication (AMC FMC Carrier - AFC), as depicted in Fig. 1 and a COTS MicroTCA.4 crate providing a wealth of services for the digital back-end and its mezzanine cards, including power supply, cooling, crate management, a CPU host and shared trigger and clock lines for all crate slots. The crate hosts several digital back-end boards (up to 10), whereas each digital back-end board hosts 2 digitizer mezzanine cards. Each digitizer has 4 ADC channels thus being able to digitize the set of 4 analog signals provided by each BPM pick-up.

Apart from the COTS crate and its infrastructure components, all electronics developed by LNLS for the Sirius RF BPM electronics are open hardware, licensed under the CERN OHL license [2] and available through the CERN Open Hardware Repository [3]. The following sections will give a brief motivation for the adoption of open hardware, in general and in the context of Sirius electronics development, and then will describe the evolution of three open hardware projects: RFFE, FMC ADC and AFC.

THE OPEN HARDWARE APPROACH

There are several different definitions and intents associated with the names "open hardware" and "open source

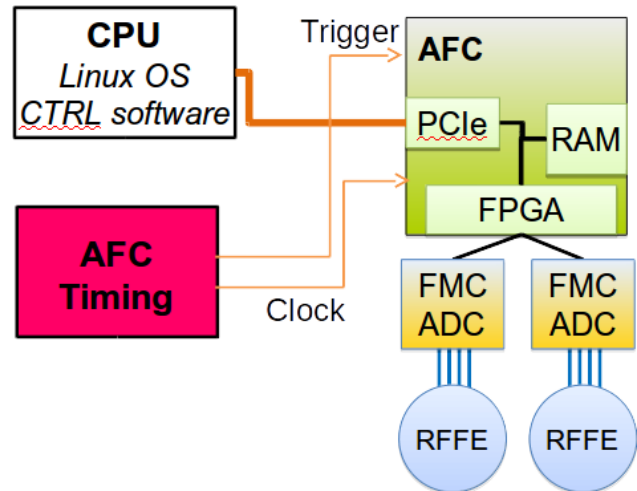


Figure 1: Simplified scheme for Sirius BPM electronics.

hardware", which are, sometimes, conflicting ones. The most common meaning, borrowed from the free software and open source software initiatives, requires the user to have access to the design files, being allowed to do modifications, to distribute and to use the project without restrictions [4]. Public release of the work may be required by some definitions, such as the one defined by the Open Source Hardware Association (OSHW) [5], while some licenses, such as CERN OHL [2] may only require documentation to be sent to users who acquire the products.

An important and frequently misunderstood aspect of the open hardware definition is its use in commercial applications. Most open hardware definitions do not prohibit commercial use of the project, in fact, most of them explicitly allow commercial use, indistinctively of application or profit intention. In effect, open hardware advocates usually do not consider restrictions on commercial use to be legitimate on open hardware projects [5,6]. All definitions, licenses and projects discussed in this work allow for unrestricted commercial use.

The motivations for adopting open hardware often includes better collaboration among users and suppliers, reducing of work duplication on similar projects, and allowing for quick advance on technologies by improving existing designs. For science instrumentation, open hardware may be preferred as a mean to allow or facilitate experiment duplication and to fully document and submit the results to peer review. An open hardware adopter is also less dependent on a specific company, as several different companies may produce and support the same open hardware design, or variations of it.

* gustavo.bruno@lnls.br

Besides those advantages, the adoption of an Open Hardware License such as CERN OHL, with common and well known terms, assures users and commercial companies they have the legal rights to allow using, modifying, producing and selling the developed hardware without risk of facing legal sanctions.

In addition to this legal assurance, the CERN OHL license requires derivative works to also be licensed by the same terms. Thus, users who receive modified versions of CERN OHL-licensed designs are entitled the same rights defined by the license, such as receiving the design files and being allowed to distribute it. These obligations also apply to designers who use part of a licensed design on other projects.

These rights and obligations, along with the centralized repository, initiatives and tools, intend to encourage the community to share the knowledge, design files and to avoid an open project from becoming closed, contrarily to the original designer goals. They also give legal assurance to commercial entities, which may profit from them by selling implementation and granting support.

Open Hardware at LNLS

The LNLS Beam Diagnostics Group decided to adopt open hardware for its BPM system after the resolution of designing its own version of the electronics was made, in order to bring knowledge and experience to the LNLS team in the area, and also to improve our autonomy and facilitate upgrades and improvements to assist in building a high-performance light source.

In this context, and inspired by the Open Hardware Initiative created by CERN in 2011, we took the open hardware path in a desire to allow better coordination with other laboratories, to reduce design effort by making use of existing designs and to help future maintenance by sharing the design updates with other agents. The open-hardware approach also mean we would be less dependent on third parties in these and future projects.

The following sections look into the development path which has been taken for each of the projects of the Sirius RF BPM electronics development. Despite the seemingly uniform approach around open hardware, the projects evolved differently, presenting both unexpected difficulties and opportunities of this.

ELECTRONICS DEVELOPMENT

AMC FMC carrier

The AMC FMC Carrier, or simply AFC [7], is a fairly complex board of our BPM system, and also the most versatile. At Sirius' BPM electronics, it works as the digital back-end of the system, hosting the FMC digitizers, doing digital signal processing and communicating with the rest of the system.

The AFC, as shown in Fig. 2, is a double-width PICMG AMC compliant board, and able to make use of most of the AMC Ports. It can be extended via two FMC HPC connectors and one μ RTM connector, all capable of receiving and

transmitting clock, LVDS and high-speed digital signals. Internally, it features a Xilinx Artix-7 FPGA, a clock crossbar able to dynamically route clock signals between the FPGA and all the connectors, and also low-jitter clock generation capabilities, enabling the board to be a White Rabbit timing receiver. A LPC176x microcontroller is responsible for board management and IPMI functionality.



Figure 2: AFC v3.1 with FMC130 mounted.

The development began in 2011, through a partnership between LNLS and the Warsaw University of Technology (WUT). Initially thought to be based on a stand-alone 18-slot FMC carrier [8], also an open hardware project, in 2012 its specification was changed to a MicroTCA AMC board to allow for reuse of the project in other applications and to leverage the infrastructure of the MicroTCA system.

The conceptual design was originally created by LNLS and evolved through request of comments in public mailing list [9], while the actual schematics and layout of the board were done under contract with the WUT, which also manufactured the first version of prototypes. The original design reused many circuits and parts from other open hardware FMC carriers, for instance SVEC [10] and SPEC [11].

After this first version, other 2 versions and 2 minor revisions of the board were designed by several actors: WUT, Creotech Instruments SA, and, more recently, LNLS. The modifications in these versions added features, corrected bugs, improved manufacturability and updated components part numbers.

In addition to its primary use in the RF BPM system, there are at least two other planned uses at Sirius for the AFC: fast orbit feedback (FOFB) processor board, and timing receiver. The FOFB processor is basically a concentrator of synchrotron beam position data of the whole storage ring and a feedback controller processor and must make use of the several multigigabit interfaces of the AFC. The timing receiver takes advantage of an assembly variation that routes signals from the μ RTM module to the FPGA transceivers, allowing low-jitter clock recovery and precise event signaling. The precise clock reference may then be distributed to the crate controller (MCH), from which it may be routed to all other boards in the crate, while events can

be decoded by the AFC and converted to trigger signals in bused MLVDS lines on the MicroTCA.4 backplane.

A μ RTM optical receiver board [12], also an open hardware project, was used for prototyping the system and improvements in both AFC and SFP μ RTM are being made by LNLS to better suit our timing system.

The current AFC version is 3.1, with an update expected to later this year with minor improvements, updated part numbers and improved manufacturability to better suit Sirius' final production.

FMC Digitizer

The FMC digitizer, as the name implies, is a mezzanine board hosting the ADCs for RF signal digitization. It is designed for direct sampling of RF signals coming from a beam position monitor, using four high-speed ADCs able to accept frequencies around 500MHz. The board may also generate the ADC clock signals in free running mode or locked from an external reference, allowing coherence between sampling and accelerator's RF frequency.

For the initial development of the FMC digitizer, LNLS contacted and asked for quotes for a contract design from several companies. Prices from most companies that accepted doing the open source design were deemed too high by LNLS, although one accepted developing the open-source hardware as long as LNLS guaranteed acquisition of 200 boards beforehand. However, due to the uncertainty of the final number needed by Sirius and to the difficulty to determine the final price of the boards before the design starts, in-house design was preferred.

The board was then designed by LNLS beam diagnostics team in 2012 and then manufactured, with some modifications, by WUT. This process resulted in two versions of the FMC digitizer: FMC130 [13], which employs Linear Technologies' LTC2208 and was mostly designed by LNLS personnel, and FMC250 [14], which employs Inter-sil's ISLA216p25 and is mostly designed by WUT. Some design features, such as VCXO and PLL circuits, the front panel and heat sink, are shared between the boards, while the analog input and digital ADC interfaces are different between the boards.

The first board to be tested by LNLS was the FMC130, with satisfactory results [15]. These digitizers were used for most of the BPM electronics evaluation at LNLS until 2016, when we started evaluating a change to a ISLA216-based design. The main expectations are a reduced operating temperature and a slight improvement in SNR performance (2dB). LNLS team indicated that FMC250's analog input return loss and linearity underperformed when compared with FMC130's, and should be improved [16].

As both boards have a CERN-OHL license, design pieces from LNLS's FMC130 could be directly transported to the Creotech's FMC250 project, speeding up development.

RF Front End

The RF front-end is the analog conditioning module of the BPM electronics, composed of two RF boards and a digital

controller board. The three boards are placed together in a mechanical enclosure as depicted in Fig. 3. To improve stability, the RF channels are switched in pairs, and the signal is then band-pass filtered around 500 MHz, and attenuated or amplified to make use of the most linear region of the ADC. The controller board houses power supplies and a microcontroller for setup and communication.

To enable collaboration with other accelerators, a decision was made to follow the open hardware approach of the rest of the BPM system. The design files for the RFFE, consisting in PCB files for the three electronics boards and CAD files for the mechanical parts are currently available at the OHWR, with a CERN OHL license.

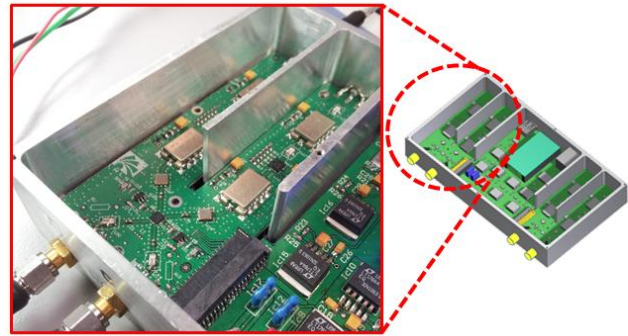


Figure 3: RFFE in its enclosure, RF channels A/C in detail.

SHORTCUTS AND PITFALLS EXPERIENCED BY LNLS

The main advantages LNLS experienced from its open hardware experience were the quicker design time and fast evolution of the projects. The bigger disadvantages were the lack of good documentation and support, specially while developing supporting software, and the difficulties on manufacturing a complex board for the first time.

AFC

The AMC FMC Carrier profited the most, but also presented most pitfalls of all the three parts of the system. Open hardware has allowed white-rabbit support, even before LNLS had enough knowledge to make use of it, simply by taking the datasheet and layout from SVEC and SPEC.

While LNLS was focused on other developments and simply using the AFC, WUT and Creotech has gone forward with the project, adding features such as the μ RTM connections and the making minor revisions to improve and update the board. This continuous update of the project by several agents is one advantage enable by open hardware.

Issues on software and firmware support were also present. MMC firmware was lacking, and a new implementation had to be done by LNLS to better user the board [17]. This is something any in-house project must tackle, but a COTS board user wouldn't expect to deal with it. On the other hand, sharing the design with other laboratories meant we had early support on this development.

The original board documentation - both design files and manual - were severely lacking in quality. The automatically generated BOM from the design file had to be hand-edited for new production. Variations had to be added to the schematic so we could generate the boards according to our necessities. A complete manual for the AFC hardware and software is yet to be done.

A final note must be done that the board layout is extremely complicated for a board this size: the demand to support several different users and interests means the board has more features than any single user may need. While it does not make the board more expensive, this burdens updates and redesigns of the board.

FMC Digitizer

The merge of the best features of FMC130 and FMC250 is a good example of what can be done with open hardware. While reverse engineering and independent development is always possible, by reusing the actual design, the LNLS team could quickly gather the best features of each board.

One issue in the FMC250 board was the trace and clearance widths of the PCB: the designed employed a minimum of 3mil, and only one Brazilian PCB manufacturer could send us a quote meeting a specification with less than a 4mil width/clearance. A detailed analysis have shown to us this limitation could have been avoided, even though it would require vast layout redesign.

This is a common pitfall between open hardware and other shared projects: a set of constraints decided by a designer in a different context may artificially limit the pool of suppliers an open-hardware user is able to use, or impose extra costs on manufacturing or redesign.

RF Front End

This board was completely designed in-house, and its different versions could not take advantage of any open-hardware project. On the other hand, it ended up being the simplest to build and maintain - by restricting its demands to those of Sirius, it could focus in the sole application.

However, inputs from different engineers in a more open collaboration could have resulted in improvements in the board. The Sirius' partner companies have added several minor improvements to the board, specially in its mechanical parts and electronics form factor. Open hardware built in close collaboration with engineers from other laboratories and companies could have benefited the design from the beginning.

CONCLUSION

Open hardware has enabled fast and highly sophisticated developments by the LNLS team, allowing us to bring a complete BPM electronics system to fruition with limited resources. Contact with existing projects also enable fast learning by the team, enabling newer projects and advances in other areas.

The results of this development has also served as base to other systems at Sirius, such as the orbit feedback and timing system, simplifying the support work for future hardware, firmware and software.

Common issues in open projects have also arose, such as difficult in finding local support and documentation failures that demanded extra work from LNLS. These issues are currently being tackled by the diagnostics team with support of partner companies.

ACKNOWLEDGMENT

The authors would like to thank everyone who helped in the OHWR mailing by discussing the projects, as well as all of those who contributed to the open hardware projects mentioned in this paper. In particular, we would like to thank Rafael Baron, currently at ESS, and Fernando Henrique Cardoso from LNLS for their fundamental contribution to the aforementioned projects.

REFERENCES

- [1] A. R. D. Rodrigues *et al.*, "Sirius Status Report", in *Proc. IPAC'16*, Busan, Korea, May 2016, paper WEPOW001, pp. 2811–2813.
- [2] CERN Open Hardware License v1.2, Sep. 2013, <http://www.ohwr.org/licenses/cern-ohl/v1.2>
- [3] Open Hardware Repository, <http://www.ohwr.org>
- [4] J. R. Ackermann, "Toward open source hardware", *U. of Dayton Law Review*, vol. 34, pp.183–222, 2009.
- [5] Open Source Hardware Association, "Open Source Hardware (OSHW) Definition 1.0", <http://www.oshwa.org/definition/>.
- [6] L. Fried, "Open Source Hardware Licenses", May 2011, <http://www.ladyada.net/library/openhardware/license.html>
- [7] AMC FMC Carrier, <http://www.ohwr.org/projects/afc>
- [8] Stand-alone 18-slot FMC carrier, <http://www.ohwr.org/projects/sac-18-fmc>
- [9] Discussion on OHWR mailing list, Jul. 2012, <http://lists.ohwr.org/sympa/arc/bpm/2012-07/msg00025.html>
- [10] Simple VME FMC Carrier (SVEC), <http://www.ohwr.org/projects/svec>
- [11] Simple PCIe FMC carrier (SPEC), <http://www.ohwr.org/projects/spec>
- [12] μ TCA RTM 8 SFP+, <http://www.ohwr.org/projects/utca-rtm-8-sfp>
- [13] FMC ADC 130M 16b 4cha, <http://www.ohwr.org/projects/fmc-adc-130m-16b-4cha>
- [14] FMC ADC 250M 16b 4cha, <http://www.ohwr.org/projects/fmc-adc-250m-16b-4cha>
- [15] S. R. Marques *et al.*, "Status of the Sirius RF BPM electronics", in *Proc. IBIC'14*, Monterey, USA, Sep. 2014, paper WECYB3, pp. 505–509.

- [16] D. O. Tavares *et al.*, “Development of an Open-Source Hardware Platform for Sirius BPM and Orbit Feedback”, in *Proc. ICALEPCS’13*, San Francisco, USA, Oct. 2013, paper WE-COCB07, pp. 1037–1039.
- [17] H. A. Silva and G.B.M. Bruno, “openMMC: An Open-Source Modular Firmware for Board Management”, presented at PCaPAC’16, Campinas, Oct. 2016, paper THPOPRPO04, this conference.

EMBEDDED CONTROL SYSTEM FOR PROGRAMMABLE MULTI-PURPOSE INSTRUMENTS

M. Broseta, J. Avila-Abellan, S. Blanch-Torné, G. Cuní, D. Fernández-Carreiras, O. Matilla, J. Moldes, M. Rodríguez, S. Rubio-Manrique, J. Salabert, X. Serra-Gallifa, ALBA Synchrotron, Cerdanyola Vallès, Spain

Abstract

At the ALBA's Computing Division, we have started the development of a high-performance electrometer, the **Em#** project, as a versatile and full customizable equipment. It is based on a SPEC board (simple PCIe FMC carrier) with customizable FMC cards and an SBC (Single Board Computer), altogether built in a single cost-optimized instrument. The whole device is designed to provide a wide range of functionalities to fulfill unique and complex experiments by means of configuration changes instead of having specific instruments.

Within the controls software development group, we started the development of a full embedded control software, based on a Linux OS that communicates with the SPEC's FPGA using the PCIe bus. This approach enables the integration of complex operations and functions in real time to higher software layers, as well as the local control, setup and diagnostics via an integrated touch-screen display controlled by the I2C protocol. For a user-level control, the system provides an SCPI API (Standard Commands for Programmable Instruments) allowing an easy integration to any control system. This paper describes the design process, main aspects of the data acquisition and the expected benefits during the integration in the Control System.

INTRODUCTION

High accuracy low current readout is an extensively demanded technique used in 3rd generation synchrotrons. They comprise a need both for diagnostics and data acquisition in today's photon labs. In order to tackle the problem of measuring from various sources of different nature and magnitude synchronously, while remaining flexible at the same time, ALBA developed years ago a 4 independent channel electrometer, the Em, based on trans-impedance amplifiers with high resolution ADC converters integrated and an Ethernet communication port.

The new **Em#** is the evolution of the 4-channel electrometer Em widely used in at ALBA since 2011. This new product solves a few limitations and provides new functionalities to make it more versatile and customizable.

ELECTRIC DESIGN

The **Em#** project uses a Simple PCI Express FMC Carrier (SPEC [1]) board to command a FPGA Mezzanine Card (FMC), designed to work as a 4-channel ADC and transfer the processed data to a Single Board

Computer (SBC), the Intel NUC DE3815 [2], using its high-speed serial computer expansion bus (PCIe).

The SPEC, it is a cost-optimized design developed by the CERN under the Open Hardware Licence (OHL) that mainly works as FMC carrier. It is powered by Xilinx Spartan 6 FPGA [3] for custom gateway designs using High-speed serial connectivity, a PCIe interface and an FMC slot.

This hardware configuration allows **Em#** implement its own FPGA control code, to manage a 4-channel ADC converter in the FMC card. The communication and data sharing with the main control software in the SBC are also part of the control code routines implemented in the FPGA. But apart from the SPEC and SBC boards, there are other hardware boards included in this equipment. Figure 1 shows the hardware diagram block of the **Em#** project.

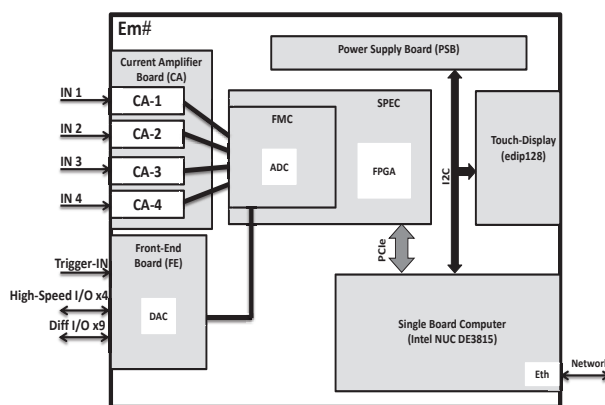


Figure 1: Em# hardware diagram block.

The Current Amplifier board (CA) contains the circuitry needed to communicate and control the 4 current amplifiers (CA-X). The Front-End board (FE) manages the trigger input and the different IO ports (4 High-Speed I/O ports and 9 Differential I/O ports). The Power Supply Board (PSB) supplies the equipment with different voltages needed by each module or board. A Touch-Screen (edip128) monitors the status and lets a general configuration of the equipment.

SOFTWARE DESIGN

The software development has been divided into three software projects, all together distributed in a single software package:

- The Linux OS
- The gateway (FPGA software)
- The main control software in the SBC (ALIN)

The **Em#** features a complex embedded control software based on Linux OS. For that reason, a light Embedded Linux has been customized specially for this equipment, with only the necessary drivers to control the available hardware and with enough functionality to run the required applications. This Embedded Linux has been created using Yocto [4], an open source collaboration project that provides a set of recipes, tools and methods that allow building custom Linux-based systems to be embedded, regardless of the hardware architecture. Among the different recipes or set of recipes used for this project, these are the most significant ones:

- meta-e3815: Recipes to build the Embedded Linux OS distribution for the Intel Atom Processor E3815, used for this project.
- meta-spec: Main recipe used to compile and install the Spec Linux drivers in the corresponding Linux kernel selected
- meta-python: This is the set of recipes that provides python support.
- meta-alba: Contains the set of recipes to install the main control software (ALIN) and its drivers in the SBC

The gateway software [5] runs in the SPEC FPGA. It is written in VHDL and the design focuses on fast acquisition and data sharing between the different modules inside the FPGA as well as with the software running in the SBC. The binary, is reprogrammed in the FPGA every time the system boots. Figure 2 shows the FPGA block diagram.

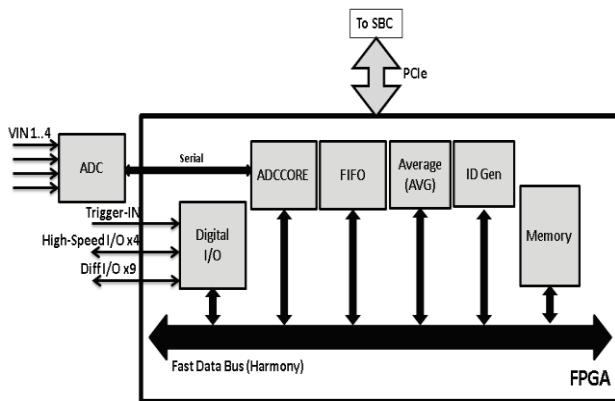


Figure 2: FPGA block diagram.

The **Em#** has been designed to acquire data at 400KSamples/second per channel. The main software is not fast enough to get the 4-channels data acquired at that sample rate via PCI bus. Therefore, the acquisition is carried out by the gateway software through a fast data acquisition bus, designed and implemented to share data between the different FPGA blocks. That fast data bus name is the Harmony Bus. Acquired data is stored in an FPGA memory block. The frames sent through this bus contains: the ID of the block which generates the frame, the data and the timestamp which indicates when the data

was generated. The main software in the SBC configures the different acquisition types and reads the acquired data stored in the FPGA memory. Other slow and low priority data such as the information displayed in the touch-screen, are also transmitted through the PCI bus

In the SBC resides and runs the main software (**ALIN**) that has been designed aiming for high versatility in the application design and easy user control of the equipment. Versatility means that the software is easily adaptable to new features, just modifying the configuration of the FPGA, or to hardware changes. **ALIN** is a multipurpose software customized to work as an electrometer. Regarding the easy user control, it offers both remote and local control interfaces. Remote control is available via telnet (using the SCPI protocol [6]) or via web through a webserver. Local control is available through navigation menus using the touch-screen display.

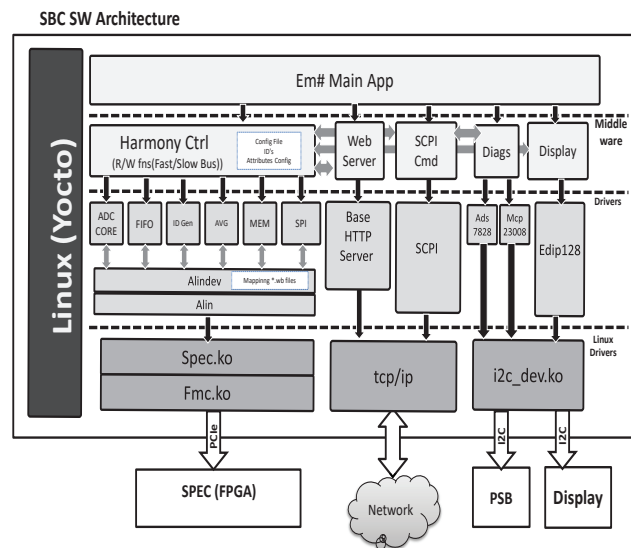


Figure 3: SBC software architecture.

Written in Python has the profits from the clean and straightforward syntax, while still performing well. The Figure 3 shows the software architecture of ALIN divided in different functional layers. The communication between modules in different layers is always from top to bottom, except for the middleware layer where a cross communication is allowed to share data between the different modules. The different layers are, from top to bottom:

- Applications: The main application is in this layer. It allows the equipment control, remote or locally. It starts when the equipment boots and it is responsible to initialize and configure the middleware modules that contain the control and diagnostics functions.
- Middleware: In this layer resides the logic that makes this equipment work as an electrometer, or as any other equipment. It also provides the functionality to interact with the equipment through the touch-screen, or through a communications port.

- Drivers: Software modules to control physical or logical devices are in this layer. There is a driver for each FPGA module, a driver for each I2C device or a driver for the SCPI that contains the protocol to remotely control the equipment.
- Linux drivers: At the bottom, there are the Linux drivers needed for the Em# project. They have been previously compiled for the Embedded Linux distribution. Examples are the SPEC driver to communicate with the SPEC board via PCI or the I2C driver to control the display and the Power Supply Board (PSB).

In the left-bottom side of Figure 3, the **Spec** and **FMC** Linux drivers [7, 8] are the kernel modules to control the SPEC and FMC cards via PCI bus. These Linux drivers are used by others like **alin** and **alinde** which implement read/write operations on the FPGA using the Self Describing Bus (SDB) [9]. This is a framework that helps to self-detect and manage the FPGA contents. It describes a series of structures used to provide metadata about the FPGA logic blocks, allowing the main software to automatically discover and configure them at runtime, via PCI bus. These SDB data structures are divided in records of 256 bytes size, where the first 64-byte are common between records and provide information about the FPGA blocks like the type of record, product, vendor, name, date, version and also the first-last address of the virtual memory space where the block data is located.

On top the alin drivers; there are specific drivers for each FPGA block. They provide functions to initialize write default values or read/write registers. These drivers use an external file that contains the register mapping. This file is auto generated using the same definition file which is used to define the FPGA block in the gateway software.

In the middleware layer, the **Harmony Control** middleware module uses these drivers to implement the electrometer **Em#** functionality. It configures the acquisition that will go through the Harmony bus in the FPGA, starts/stops the acquisition and process the acquired data in the FPGA memory. Acquired data and configuration parameters are shared to the rest of the middleware modules. External configuration of some predefined main software parameters, it is possible through a configuration file, which is loaded every time the system starts or after user request, by a command execution.

Remote control of the equipment is possible through a simple set of ASCII commands. These user control commands are implemented following a standard protocol for programmable instruments; the SCPI protocol. SCPI commands are ASCII textual strings that can contain one or more keywords, many of which take parameters. Responses to query commands are typically ASCII strings. The **SCPI middleware** module contains the list of control and configuration commands and their associated read/write call-back Em# functions. The **SCPI**

driver, used by this middleware module is where the SCPI protocol is implemented.

The **Em#** also includes a web server for remote monitoring and overall control. The web offers a general equipment status, the current and voltages values read from the 4 channels, the configuration of the channels, the status and configuration of the acquisition, the last data acquired, the status and configuration of the 16 I/O ports, general diagnostics, etc. The **Webserver middleware** starts/stops the server using the **Base HTTP Server** [10] python library. The Webserver middleware is also responsible to gather the information to be shown in the web client, generating periodically a *Javascript Object Notation* (JSON) file [11]. Figure 4 shows the typical applications running in the browser client. It is a JavaScript application that uses jQuery library [12] to read the contents of the JSON file and keep the web contents updated. In the opposite way, when a parameter is modified in the browser client, it executes a PHP code in the server side. The webserver middleware captures the data send in the POST PHP method and then executes the corresponding call-back command in the main application.



Figure 4: Web application in browser client.

It is also possible to do a more specific remote control of the FPGA through the tools available via SSH. There is a handful set of tools that help designers to check/configure the status of the FPGA, to get the SDB structures, write binary file or read/write to the FPGA devices or to configure some general parameters of the equipment

Local equipment control is possible through to the touch-screen display. The **Display middleware** module allows the user control through navigation menus, to locally configure or check the status of the equipment via the touch-screen display. The display is programmed by

means of a protocol of high-level language graphic commands via I²C [13]. The **edip128** implements the communication protocol with the display.

ALIN also includes functions to control and self-detect its own diagnostic status. That is done in the **Diagnostics middleware** module. Check the harmony bus stability, self-detect the consumption of the power supplies in the Power Supply Board (PSB) board, among other tasks are some of the diagnostics examples done by this module. To control the PSB board is done using the **Ads7828 (ADC)** and **Mcp23008 (Port-Expander)** I²C drivers.

CONCLUSIONS

There are already many customizable business solutions in the market that include an FPGA, a CPU and an FMC connector, all together on the same equipment. The **Em#** project tries to take the advantage of such solutions while offering a reduced cost. The software project has been designed modular to adapt to any other similar hardware approach, occasionally needing few changes in the driver modules or different configuration files. The control toolkit has been also designed to ensure an easy integration into any control system, regardless of the framework used for the control.

Direct memory access (DMA) is currently not supported in this design and therefore FPGA data is read by the NUC through virtual memory spaces via PCIe. That could be a problem due to the main software not being fast enough to meet the desired acquisition time of 400 KSamples per second. Instead, it can be considered as an advantage because the final solution applied (a FPGA memory block, the Harmony bus and the dynamic ID's) allows keeping a completely separate functionality between FPGA and NUC. The configuration and control resides in the NUC while the data acquisition is mainly in the FPGA. The result is that functionality of the **Em#** is not limited to work only as an electrometer but can be adapted and extended with other flavours in the control applications domain.

REFERENCES

- [1] Simple PCIe FMC Carrier (SPEC). OHWR website: <http://www.ohwr.org/projects/spec/wiki>
- [2] Intel NUC DE3815, <http://www.intel.eu/content/www/eu/en/nuc/nuc-kit-de3815tykhe-board-de3815tybe.html>
- [3] Spartan-6 FPGA Data Sheet, https://www.xilinx.com/support/documentation/data_sheets/ds162.pdf
- [4] Yocto Project, <https://www.yoctoproject.org>
- [5] X. Serra *et al.*, "A Generic Fpga Based Solution for Flexible Feedback Systems", ALBA-CELLS Synchrotron, presented at PCaPAC16, Campinas, Brazil, Oct 2016, paper FRFMPLCO06, this conference.
- [6] "Standard Commands for Programmable Instruments", European SCPI Consortium, May 1999.
- [7] A. Rubini, "SPEC Software Support", CERN, February 2014.

- [8] A. Rubini "FMC Bus Abstraction for Linux", CERN, February 2014
- [9] A. Rubini, W. Terpstra, M. Vange, "Self Describing Bus (SDB) – Specification for Logic cores – Version 1.1", April 2013
- [10] Guido van Rossum and the Python development team, "The Python Library Reference (Release 2.7.12)", Python software Foundation, September 2016.
- [11] Introducing to JSON, <http://www.json.org>
- [12] JQuery 1.9 documentation, <http://www.jquery.com>
- [13] SMBus/I²C Protocol, <http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/plain/Documentation/i2c/smbus-protocol>

GATEWARE AND SOFTWARE FRAMEWORKS FOR SIRIUS BPM ELECTRONICS

L. M. Russo*, J. V. F. Filho, LNLS, Campinas, SP, Brazil

Abstract

The Brazilian Synchrotron Light Laboratory (LNLS) is developing a BPM system based on the MicroTCA.4 standard comprised of AMC FPGA boards carrying FMC digitizers and an AMC CPU module. In order to integrate all of the boards into a solution and to support future applications, two frameworks were developed. The first one, gateway framework, is composed of a set of Wishbone B4 compatible modules and tools that build up the system foundation, including: PCIe Wishbone master; FMC digitizer interfaces; data acquisition engines and trigger modules. The gateway also supports the Self-Describing Bus (SDB), developed by CERN/GSI. The second one, software framework, is based on the ZeroMQ messaging library and aims to provide an extensible way of supporting new functionalities to different boards. To achieve this, this framework has a multilayered architecture, decoupling its four main components: (i) hardware communication protocol; (ii) reactor-based dispatch engine; (iii) business logic, comprising of the specific board functionalities; (iv) standard RPC-like interface to clients. In this paper, motivations, challenges and limitations of both frameworks will be discussed.

INTRODUCTION

Sirius is a new 3 GeV synchrotron light source under construction in Brazil, with a 0.27 nm.rad natural emittance and 518 meters circumference. The beginning of machine installation is scheduled to the end of 2017 [1].

In this context, a BPM electronics system has been specified, designed and developed by the Beam Diagnostics team at LNLS, reaching its final phase of long-term testing and hardware manufacturing in the following year [2].

As the system was developed from scratch, employing new high-performance data acquisition and communication technologies (e.g., MicroTCA.4, FPGA, FMC, PCIe) [3], the need for an FPGA gateway and software infrastructure frameworks emerged. For that matter, it was sought the use of consolidated codebases and collaborative development through an open source approach. Examples of this were the initial collaboration with the Warsaw University of Technology, the use of a community-driven set of repositories aimed at building generic software/gateway modules from the OHWR collaboration [4] and the use of projects such as the ZeroMQ messaging library [5], the CZMQ High-Level C Binding library [6] and the Malamute Messaging Broker [7] which leveraged many years of development.

In the next sections, the requirements and the details of these two frameworks, licensed under the copyleft GPLv3 and LGPLv3 licenses, will be described.

* lucas.russo@lnls.br

GATEWARE FRAMEWORK

The gateway framework consists of a set of modules, mainly written in VHDL, that interconnect with each other and to external interfaces. It can be used as a basis to other designs. The general architecture is as depicted in Fig. 1.

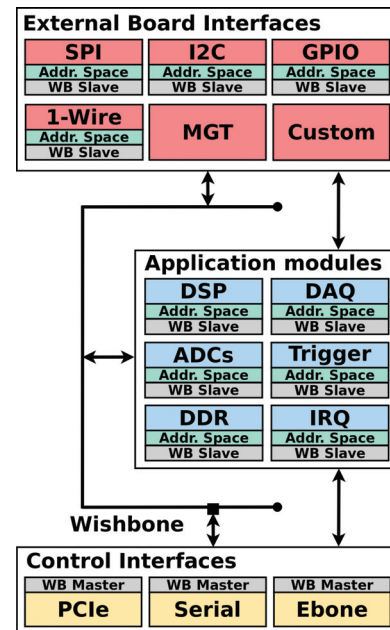


Figure 1: Gateway Framework Architecture.

General Description

The first basic component of the framework is the *Control Interfaces*, as shown in Fig. 1, and it encompasses standard communication interfaces to a controlling node, acting as a Wishbone Master to the gateway side. Currently, 3 types are supported: PCIe Gen1 (for Xilinx Artix7 FPGA) and Gen2 (for Xilinx Kintex7 FPGA), with accompanying linux driver with PIO, single buffer and scatter-gather DMA; RS232 Syscon for simple serial communication supporting auto-baud generation; preliminary support for Ethernet MAC + Etherbone [8] for UDP communication and userspace software library.

The second layer, *Wishbone Infrastructure*, is a set of Wishbone modules and functions that provide the general interconnection between controllable modules (i.e., that can receive/transmit configuration parameters and/or low-bandwidth data) and address space enumeration with SDB [9] support. This is the standard adopted within the framework for consistently interfacing with all *Control Interfaces*.

The third layer, *Application Modules*, is where all of the framework functionalities reside, comprising components from third-parties and LNLS Beam Diagnostics team. A

variety of modules are supplied, ranging from DSP modules (e.g., adders, multipliers, dividers, filters, CORDIC), DDR interface and Interrupt Management, to Data Acquisition, ADC interfaces and Trigger logic.

The forth and last layer, *External Board Interfaces*, is a set of board communication protocols used to interface with Multigigabit standards (i.e., MGT) and to configure or interface with board peripherals, such as: SPI (ADC, clock circuit configuration, etc.), I2C (EEPROM, temperature/voltage/current sensors, etc.), 1-Wire (EEPROM) and GPIO (LEDs, generic interfaces, etc.). Also, there are some custom interfaces implemented, like a parallel ADC interfaces for data acquisition.

Development

The gateway framework is a development and integration effort from a diverse set of modules that took place in the context of the Sirius BPM electronics focusing on creating generic modules, written in portable VHDL/Verilog to avoid vendor lock-in and to encourage reusability. This encouraged that, besides the primary target platform (i.e., BPM electronics), the Sirius MicroTCA.4 timing receiver also started porting its gateway to this framework.

The development is coordinated through the GitHub platform [10] following a simple branch model and fork+pull requests. Currently, the framework is being versioned together with its main application (i.e., BPM), but a separate repository is planned for simpler integration with other projects.

SOFTWARE FRAMEWORK

The software framework, called *HALCS* (Hardware Abstraction Layer for Control Systems) [11], can be defined as a software daemon that abstracts away a given hardware platform and its functionalities by means of a common interface to hardware and generalized set of *specific application modules*. It is written in C99, implementing a simple, yet effective, scalable object-oriented API, following the guidelines in [12]. The framework was thought as a way of simplifying the development and deployment of applications, while keeping it extensible and flexible. The general framework architecture can be viewed in Fig. 2.

General Description

HALCS main components are: *Hardware Abstraction Layer* (yellow solid box in Fig. 2), defining a common interface for all *Board Support* (yellow dashed box in Fig. 2) implementations, such as PCIe and TCP/IP; *Dispatch Engine* (green box in Fig. 2), that receives message requests from upper-layers and safely demultiplexes them according to the selected *Board Support*; *Specific Modules Layer* (blue boxes in Fig. 2), implementing application-specific logic, receiving message requests from external clients through an RPC interface and ordering lower layers to perform a desired set of operations; *Client Interface* (red boxes in Fig. 2), providing an external API for external clients to communicate with the application.

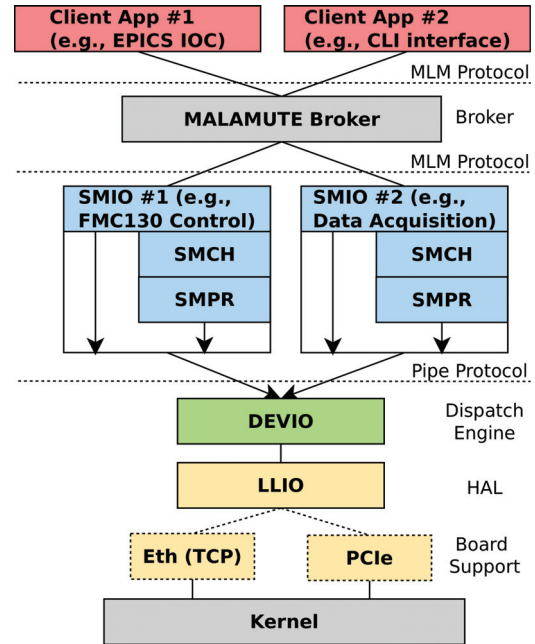


Figure 2: Software Framework Architecture.

The framework also relies on an external standalone messaging broker called *Malamute* [7] to provide reliability, authentication and mailbox messaging pattern. In the future, *HALCS* is planned to use, besides the mailbox pattern, a higher-efficiency, asynchronous, stream pattern (ideal for data acquisition applications) and, possibly, a service pattern for modules implementing logic that can be replicated, like some calculation engine or some data processing logic. Both of these patterns are also available in the broker API.

Hardware Abstraction Layer

The *Hardware Abstraction Layer* is implemented as a standalone software library, called *LLIO*, for Low-Level Input/Output, and acts as a generic interface for hardware operations. The selection of which interface to use (currently PCIe or TCP/IP) is done at compile time and can be easily changed by opting for another implemented set of *LLIO* operations, as long as it follows the same *LLIO* interface.

Dispatch Engine

The *Dispatch Engine*, called *DEVIO* in Fig. 2, is the core of the framework. It acts both as the entry point for a constructed application using the framework and a serialization engine to safely access the hardware without the use of traditional synchronization points (e.g., mutexes, semaphores). Instead, all of the communication is done by sending/receiving messages between layers, improving decoupling, flexibility and avoiding programmer errors in protecting shared resources. Typically, the *DEVIO* layer performs the following steps:

1. Calling thread calls `devio_new()` method for creating a new instance of *DEVIO* layer
2. *DEVIO* registers the selected *LLIO* operations

3. (Optionally) *DEVIO* tries to parse an SDB structure located in hardware to dynamically enumerate its modules as *SMIO* modules
 - 3.1. If successful, spawns each *SMIO* module as a new CZMQ actor, executing its entry point routine and opening an inter-thread communication (inproc), using PAIR-PAIR ZeroMQ sockets (Pipe Protocol in Fig. 2)
4. *DEVIO* opens a control socket to the calling thread, so it can send commands to *DEVIO* at anytime, such as: REGISTER_SMIO, to register a new *SMIO* (executing the actions of step 3.1.) and UNREGISTER_SMIO, to unregister an *SMIO* (executing the opposite actions of step 3.1.)
5. *DEVIO* starts its event-driven reactor engine to wait for commands from the registered *SMIOs* and dispatching the appropriate messages to the *LLIO* layer

Specific Modules Layer

This layer is where all of the business logic resides. It is composed of self-contained modules that implement specific operations and communicates with lower layers by sending/receiving messages through the PAIR-PAIR socket opened by *DEVIO*. For higher layers, it uses the MLM protocol [7] and, on top of that, a simple RPC protocol to export its functionalities to external clients.

In order to register new functions, one would have to fill a description structure informing its opcode, how many and what type of the arguments it receives and the type and size for the return value. After that, the exported function can be implemented by following the function signature: `int (*disp_table_func_fp)(void *owner, void * args, void *ret)`. Finally, all that is left is associating this function to the description structure and registering it in the *SMIO* exported operations. All of these actions are available in the API, through functions.

There are also two optional layers inside *SMIO*, called *SMPR* and *SMCH*, showed in blue boxes Fig. 2. They can be used to implement specific protocols (e.g., SPI, I2C, 1-wire, GPIO) and specific operations on top of it (e.g., Clock-distribution IC using SPI protocol, Programmable Crystal Oscillator using I2C protocol for configuration).

Currently, the *SMPR* layer implements interfaces with the OpenCores SPI and OpenCores I2C gateway modules. These modules are controllable through a simple set of registers and implements the respective protocol. The *SMCH* layer, which is independent of the *SMPR* and only relies on a standard interface to communicate with it, implements various IC interfaces, such as: ISLA216P ADC, AD9510 PLL and clock-distribution, EEPROM, Si57x crystal oscillator, I2C switches, etc. In fact, one can change the *SMPR* protocol of any *SMCH* chip by simply choosing another protocol for it. Any specificities can be controllable through the specific *SMPR* protocol API, while the implementation of both layers remain unaffected.

Client Interface

The *Client Interface* is basically a set of wrapper functions that encapsulates the necessary arguments and return types for each *SMIO* exported function. There is also a generic API for sending/receiving commands to *SMIO* functions that relies on the description structures for checking the type, arity and size of arguments. This layer forms the API in which client programs use to communicate with a given application implemented with *HALCS*.

FUTURE WORK

Currently, *SMIOs* only have the mailbox interface, implementing a synchronous request-reply RPC, available through *Malamute*. In this pattern, each *SMIO* receives a mailbox in the broker and can send/receive messages to that specific mailbox. However, in some cases, like data acquisition, a more suitable pattern is a data stream (i.e., publish-subscribe) in which each client that wants to receive some published data registers to a certain *topic* and any control can be set through the regular mailbox pattern API. This in fact creates 2 protocols that implement specific use cases: a high-performance protocol for large amounts of data data (e.g., data acquisition); low-performance protocol for control and monitoring (e.g., configuration parameters). In order to fix these issues, an event API for asynchronous communication and a stream API for high-performance data transmission is under study and should be implemented soon.

CONCLUSION

Two frameworks for developing interfaces to hardware application boards were presented. Currently, two applications are using the frameworks inside LNLS, namely BPM electronics and MicroTCA timing receiver, but more are envisioned. New features are included with relatively ease and new APIs for high-performance data streaming and asynchronous communication will be implemented over the next months.

ACKNOWLEDGEMENTS

The author of this paper would like to acknowledge the collaborative work of Adrian Byszuk from Creotech Instruments SA, for the development of the FPGA PCIe core and the linux driver, Andrzej Wojeński from the Warsaw University of Technology, for the initial gateway and software codes, OpenCores and Open Hardware (OHWR) collaborations, for an excellent combined set of software/gateway/hardware projects and the initiative to embrace and push forward the development of free and open source projects.

REFERENCES

- [1] A. R. D. Rodrigues *et al.*, "Sirius Accelerators Status Report", in *Proc. IPAC'15*, Richmond, VA, USA, May 2015, paper TUPWA006, pp. 1403–1406.

- [2] S. R. Marques *et al.*, "Status of the Sirius RF BPM Electronics", in *Proc. IBIC'14*, Monterey, CA, USA, Sep. 2014, paper WECYB3, pp. 505–509.
- [3] D. O. Tavares *et al.*, "Development of an Open-Source Hardware Platform for Sirius BPM and Orbit Feedback", in *Proc. ICALEPCS'13*, San Francisco, CA, USA, Oct. 2013, paper WECOCB07, p. 1039.
- [4] Open Hardware Collaboration, <http://www.ohwr.org>
- [5] ZeroMQ Messaging Library Project, <http://zeromq.org>
- [6] CZMQ High-Level C Binding Project, <http://czmq.zeromq.org>
- [7] Malamute ZeroMQ Messaging Broker Project, <https://github.com/zeromq/malamute>
- [8] Etherbone Core Communication Project, <http://www.ohwr.org/projects/etherbone-core>
- [9] Self-Describing Bus Project, www.ohwr.org/projects/fpga-config-space
- [10] Beam Position Monitor Gateway Project, <https://github.com/lnls-dig/bpm-gw>
- [11] Hardware Abstraction Layer for Control Systems Project, <https://github.com/lnls-dig/halcs>
- [12] CLASS ZeroMQ RFC, <https://rfc.zeromq.org/spec:21/CLASS>

A FRAMEWORK FOR DEVELOPMENT AND TEST OF xTCA MODULES WITH FPGA BASED SYSTEMS FOR PARTICLE DETECTORS*

T. C. Paiva, L. A. Ramalho, A. A. Shinoda, Faculdade de Engenharia de Ilha Solteira, UNESP, Brazil
A. Cascadan†, V. F. Ferreira, M. Vaz, Núcleo de Computação Científica, UNESP, Brazil

Abstract

This work describes a framework to develop firmware for ATCA carrier boards with FPGA. It is composed of an ATCA IPMI protocol implementation for environmental monitoring and control, and a companion XVC protocol implementation for remote FPGA configuration and system debugging. A study case is also presented of the development of a setup to validate a Level 1 Tracker Trigger System proposed for CMS at HL-LHC.

INTRODUCTION

The ATCA standard, originally created for the telecommunication industry, has recently aroused the interest of other fields. In the physics community the ATCA is already been used [1, 2], due to the high level of support to system monitoring and control it provides [3]. As an example, the ITER experiment developed a modular control and data acquisition systems designed on ATCA platform [4]. Also, the LBNL synchrotron built an ATCA readout system with an ATCA processor blade which performs image descrambling and formatting [5]. And, recently, the ATLAS Cathode Strip Chamber (CSC) back-end readout system has been upgraded to ATCA environment with high speed links, commercial Pigeon Point IPMC and Timing Trigger and Control (TTC) I/O for synchronization [6].

ATCA standard is also the choice of the AM+FPGA group to the development of a Level 1 Tracking Trigger (L1TT) system for the Compact Muon Solenoid (CMS) operating in the High Luminosity LHC [7]. Sao Paulo Research and Analysis Center (SPRACE), as part of this international collaboration, is in charge of building a demonstration framework to validate those R&D prototypes. It consists of two ATCA shelves with custom ATCA carrier boards, named Pulsar 2b [8]. One of them implements the Data Sourcing System, which is an emulator for the Outer Tracker detector electronics output, and the other contains the Pattern Recognition System being proposed.

The L1TT electronics, which will be physically inaccessible during the LHC runs, requires safe and reliable operation, remote configuration and JTAG tests of all its FPGA devices. These requirements are attended by our framework, as it follows Intelligent Platform Management Interface (IPMI) [9] and Xilinx Virtual Cable (XVC) [10] specifications, and also it uses Pulsar 2b ATCA carrier boards. The follow-

ing will describe the implementation of this system and its application in the demonstration setup.

ATCA AND IPMI

The ATCA is a standard defined by PICMG [3] that specifies a chassis system (shelf) intended to provide a comprehensive and reliable environment for hosting carrier boards and their extensions, such as mezzanines and transition modules. It defines also the concept of Hardware Platform Management (HPM) consisting of a distributed control system, which relies on the IPMI specification. Controllers collect information from different types of sensors spread in the system and take actions, like speeding up cooling fans or shutting down modules, to ensure a safe environment for the electronic boards. HPM also enables hot swap operations to allow replacement of electronic units with the system powered on. There are three types of controllers in this architecture:

Shelf Manager Controller (ShMC): it is the central element in the shelf, located in the Shelf Manager (ShMC) board, that gathers information from the installed hardware, generates alarms and controls power supply and fan speed.

Intelligent Platform Management Controller (IPMC): it is installed on each carrier board and is a local HPM agent, directly connected to ShMC.

Modular Management Controller (MMC): it is the simplest management agent residing in the extension modules of the carrier boards, like Advanced Mezzanine Card (AMC) and Rear Transition Modules (RTM), that is only able to execute basic commands sent by the IPMC.

The HPM management elements connect to each other using the Intelligent Platform Management Bus (IPMB) as a physical layer for the IPMI communication [11]. IPMB-0 stands between the ShMC and IPMC cards and IPMB-L connects IPMC and MMC devices. The Figure 1 shows the controllers and the connection between them inside a shelf.

The starting point of the framework was the Pulsar 2b carrier board with an IPMC card already used for simple tasks but with no support for IPMI operations. The previously chosen proprietary real time operating system, RTX from Keil, prevented the use of open source projects as base of the IPMI solution, like CoreIPM, and it was decided for an implementation from the scratch, with a very minimalist approach. It proved necessary support for the hot swap

* This material is based upon work supported by the São Paulo Research Foundation (FAPESP) under Grant No. 2013/01907-0 and by funds provided by the cooperation agreement with PADTEC S/A under FUNDUNESP Grant No. 2215/2013.

† cascadan@ncc.unesp.br

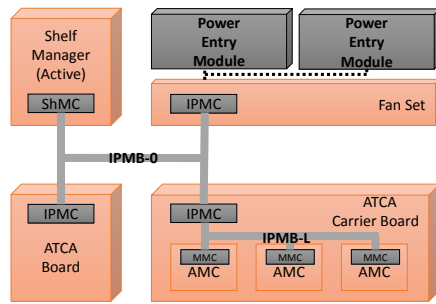


Figure 1: ATCA distributed management consisting of the ShMC, the IPMC and the MMC controllers, and their inter-connections through IPMB-0 and IPMB-L buses.

procedure to allow the ShMC to recognize the board when plugged and monitoring capabilities [12].

The `ipmb_traced` tool was used to sniff the IPMB-0 message exchanging. Carrier boards found in the market served as a reference during the development to harness the understanding about the IPMI specification and to select the minimal set of IPMI commands needed [13].

During extensive tests, the IPMI implementation for the Pulsar 2b IPMC demonstrated its capability to entirely map all the sensors and to effectively handle hotswap operations. Those functionalities allow the ShMC to recognize the carrier board, executing power negotiation and fan speed control in response to the sensor reading gathered [13].

REMOTE ACCESS

The programming and debugging of FPGA components is done through its JTAG interface, and standard ports available on computers must be translated to the JTAG protocol. Once an ATCA backplane interconnects carrier boards, as shown in Figure 2, ATCA network switches can be used to provide remote access. The XVC protocol [14], specified and supported by Xilinx, defines the translation between TCP/IP frames and JTAG signals, which fits very well for this case.

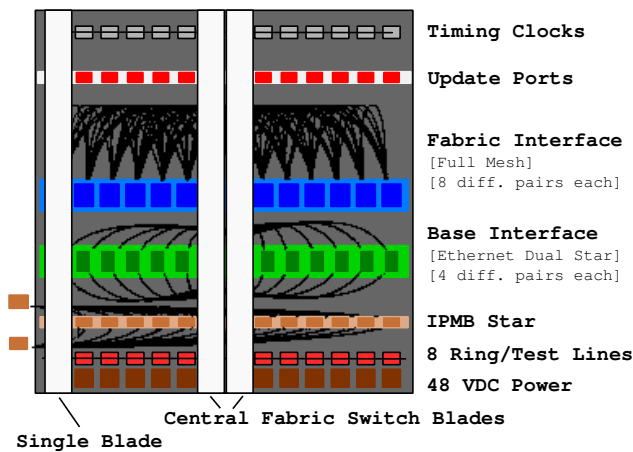


Figure 2: ATCA switch provides external access to FPGA devices on the carrier boards through shelf backplane. [15]

Since the IPMC is connected to the base interface and also is part of the JTAG chain, according to the Pulsar 2b design, the implementation of an XVC service was straightforward. Therefore, the connection between Xilinx device tools and the Pulsar 2b FPGA was made possible by a Vivado Hardware Server instance acting as a gateway for remote programming and debugging.

The ATCA switch also provides fabric interfaces that enable directly communication with the gigabit transceivers of the Pulsar 2b FPGA. This grants flexible write and read channels to the board for data exchanging, as it is the purpose of the IPbus protocol [16], which runs over UDP/IP. Both XVC and IPbus connections implemented for this project are illustrated in Figure 3.

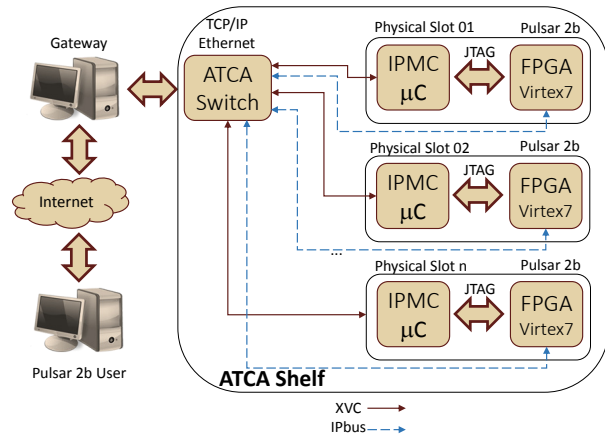


Figure 3: Connections between developer and FPGA using the XVC service and IPbus.

FRAMEWORK APPLICATION

In the L1TT demonstration, the Data Sourcing system deliver data as the real system, with an user interface with IPbus connection to a computer [16], writing and reading data in a dual port memory and executing commands like start and reset.

The computer also controls the Timing, Trigger and Control (TTC) LHC sync system that provides a common clock source and the BC0 (LHC orbit flag 11 KHz) signal to synchronize all boards in different shelves. The data stream from Data Sourcing Shelf to Pattern Recognition Board happens at the same time in all links, as shown in Fig. 4.

The data transfer happens using a wrapper interface, which provides the transceiver user clock, data valid and data input signals. The wrapper is a high speed serial link protocol that can be performed by any differential protocol and encoding system. For the first shelf level tests, it was used the Gt Wizard 64b66b protocol [17] as a wrapper to connect the Data Sourcing Shelf and the Pattern Recognition Board.

We applied a tuning process to improve the performance of the channels to achieve higher speed links without losing reliability. The process was performed using the Xilinx

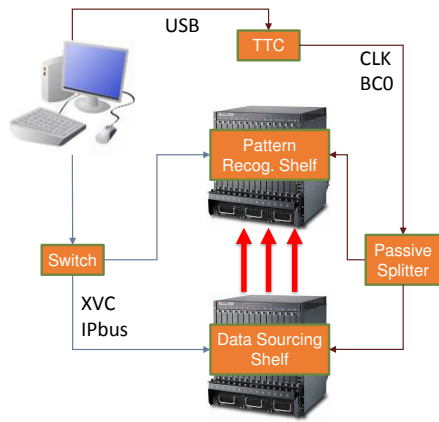


Figure 4: Shelf level test demonstration setup.

IBERT tool, which allows the developer to measure the bit error rate (BER) and eye scans while applying pseudo random binary sequences to stress the channels. Therefore, it is possible to equalize the channels changing their tuning parameters, e.g. TX Differential Swing Voltage, and check their influence in the errors measured.

As result, we found a tuning setup for channels in 8 Gbps that provides a BER in the order of 10^{-12} . The quality could also be checked in the eye scan blue open area in the Fig. 5. The reliability reached meets the requirements of the L1TT demonstration.

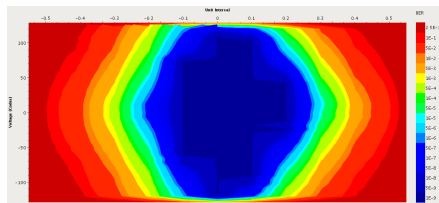


Figure 5: Eye scan of a L1TT demonstration channel.

CONCLUSION

The IPMI service implemented for the Pulsar 2b IPMC enables the carrier board to operate in a safe and optimal condition inside the shelf. When the Data Sourcing System pushes the data rates to the limit, the FPGA devices tend to overheat, which is prevented by the ShMC adjusting the fans speed according to the sensors reading. The ShMC is also able to deactivate the boards in dangerous situations, as in power surge events or temperature excessive rises. With the XVC service, it is possible to remote program and debug FPGA devices in an intercontinental connection. The remote access also allows the collaboration to share the hardware infrastructure available, saving both resources and time.

The Data Sourcing shelf emulates the Tracker detector output, sending off-line simulated pre-trigger data through links synchronized with the help of LHC TTC clock. The

platform proved to deliver event data with a error rate low enough for the tests purpose, at 8 Gbps for each channel resulting approximately 32 Gbps per link. Future work will improve the data rate communication and also will define better constraints for the framework operation, which will assist the L1TT Demonstration in several aspects.

REFERENCES

- [1] V. Bobillier *et al.*, “MicroTCA and AdvancedTCA equipment evaluation and developments for LHC experiments,” in: *Journal of Instrumentation* 11.02 (2016), p. C02022.
- [2] R. S. Larsen, “Recent Progress in Next Generation Platform Standards for Physics Instrumentation and Controls,” in: *18th IEEE-NPSS Real Time Conference (RT)* 1.1 (June 2012), pp. 1–5., n 3.0 *AdvancedTCA Base Specification*. 2008.
- [4] M. Correia *et al.*, “Development of High-Availability ATCA/PCIe Data Acquisition Instrumentation,” in: *IEEE TRANSACTIONS ON NUCLEAR SCIENCE* 63.3 (June 2016), pp. 1620–1624.
- [5] P. Mcvittie *et al.*, “A Readout System for High-Speed CCD Cameras Based on Advanced Telecommunications Computing Architecture,” in: *IEEE Nuclear Science Symposium and Medical Imaging Conference Record (NSS/MIC)* 1.1 (2012), pp. 1623–1625.
- [6] R. Claus, “A new ATLAS muon CSC readout system with system on chip technology on ATCA platform,” in: *Nuclear Instruments and Methods in Physics Research A* 1.1 (2015), pp. 272–274.
- [7] Technical Proposal for the Phase-II Upgrade of the Compact Muon Solenoid. <https://cds.cern.ch/record/2020886/files/LHCC-P-008.pdf>
- [8] ATCA at Fermilab, <http://www-ppd.fnal.gov/ATCA/>
- [9] INTEL, HEWLETT-PACKARD, NEC and Dell, *IPMI Specification v 2.0*, Tech. rep. Intel, 2013.
- [10] Xilinx Virtual Cable product, <http://www.xilinx.com/products/intellectual-property/xvc.html>
- [11] P. Perek and D. Makowski, *Intelligent Platform Management Controller for ATCA Carrier Boards*, Ed. by Lambert Academic Publishing, 1st ed. Saarbrücken: Lambert Academic Publishing, 2011.
- [12] T. C. Paiva, “Remote Development Environment with Reconfigurable Components in the Advanced Telecommunications and Computing Architecture Context,” MA thesis. São Paulo: FEIS UNESP, Sept. 2016.
- [13] L. A. Ramalho *et al.*, “Development of an Intelligent Platform Management Controller for the Pulsar IIB,” in: *2015 IEEE Nuclear Science Symposium and Medical Imaging Conference* 1.1 (2015), pp. 1–4.
- [14] Xilinx Virtual Cables specification, https://raw.githubusercontent.com/Xilinx/XilinxVirtualCable/master/README_XVC_v1_0.txt
- [15] Choosing the Right ATCA Fabric, <http://www.embedded.com/design/connectivity/4009302/Choosing-the-Right-ATCA-Fabric>
- [16] IPbus: a flexible Ethernet-based control system for xTCA hardware.
- [17] High Speed Serial, <http://www.xilinx.com/products/technology/high-speed-serial.html>

UVX CONTROL SYSTEM: AN APPROACH WITH BEAGLEBONE BLACK

S. Lescano, INSA Lyon, Villeurbanne, France

A. R. D. Rodrigues, E. P. Coelho, G. C. Pinton, J. G. R. S. Franco*, P. H. Nallin,
Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Abstract

UVX is a 1.37 GeV synchrotron light source that has been in operation by the Brazilian Synchrotron Light Laboratory (LNLS) since 1997. Its control system, which was completely developed in-house, has received some upgrades lately in order to get around issues from aging, improve performance and reduce maintenance costs. In this way, a new crate controller was designed. It is based on BeagleBone Black single-board computer (SBC) [1], a cheap open hardware and community-supported embedded Linux platform that will be adopted for some control system applications in Sirius [2], the upcoming Brazilian light source. In this paper, we describe an overview of the design and results obtained.

INTRODUCTION

As shown in [3], the control system of our machine is organized in three levels. The lowest one comprises 3U VME-like crates with I/O cards and a local controller module, responsible for managing these cards and communications to the upper level. We name this low-level control system based on crates with I/O cards as LOCO (from Local COntroller).

Since the start of UVX operation, I/O cards are the same. Most of them has only TTL digital pins and analog inputs and outputs with 12 or 16-bit resolution. There are also specific boards for reading Pt100 temperature sensors or counting pulses, for instance. However, LNLS Controls Group developed three generations of local controller boards, as shown in the next section.

UVX LOCAL CONTROLLERS TIMELINE

In a nutshell, the goal while developing a local controller for UVX control system is always to build a reliable, low-cost and general-purpose module to manage the crates. Since the beginning of control system implementation, another important feature is always present in our controller designs: units of a given model run the same program, a universal software that contains routines to read and write all types of I/O boards and performs all possible operations specified by high-level applications. Running local controllers with a unique software simplifies maintenance tasks and the embedded software development process. With an architecture like that, only high-level applications know the equipments controlled by each crate. Also, the application protocol used in communications to controller crates was specified by LNLS engineers and has never suffered drastic changes.

First Generation: Z80 and Serial Communication

First version of UVX local controller board was based on a Z80 microprocessor. Its software, named PSICO, was written in assembly language. Controller's communication interface was designed internally and is based on RS-485. Although these boards are in operation until today, they are treated as a legacy system, as we don't provide updates for the embedded software anymore and many of their electronic components are obsolete.

Second Generation: eZ80 and Ethernet

Aiming the adoption of a widely used communication interface standard (Ethernet), a new version of the local controller board was developed in the early 2000s [4]. Zilog eZ80F91 microcontroller was the hardware platform chosen. The embedded software was rewritten in C and renamed to PROSAC. Some of these boards are still in operation in UVX storage ring systems, despite the lack of updates for this design in the past years.

Third Generation: SBC with FreeBSD

Because second generation local controller was too much dependent on its hardware platform (an eZ80 microcontroller with built-in Ethernet MAC), the design of a new controller was started. At that time, Controls Group engineers wanted to experiment with the emerging world of embedded Linux platforms too.

An industrial-class single-board computer (Advantech PCM-4153F) was picked. Local controller software (PROSAC) was completely rewritten in C, taking into account libraries such as Pthread (POSIX threads) and the presence of an abundant secondary memory, used for storage of waveforms for special applications. Routines to read and write I/O cards were implemented with the mapping of SBC's PC/104 bus into LOCO crate backplane bus lines.

Although we have moved to an environment which is not hard real-time with this design, in practice all requirements for controller operation in UVX were satisfied. Today we have many of these local controllers under operation, notably those which interfaces to UVX storage ring high current power supplies.

Various flavours of Unix and Linux operating systems were tested for use with Advantech SBC. FreeBSD surpassed all the others because tests revealed that it was more responsive while performing synchronized operations over power supplies (beam energy ramp and magnets cycling).

This local controller based on Advantech single-board computer is the most failsafe we ever made. Machine operators always say that.

* guilherme.franco@lnls.br

Motivation for a New Design

We have been replacing local controllers of first and second generation by third generation ones for the past years, in order to achieve better reliability during operation or to substitute controllers that presented some serious fault.

Because of Advantech PCM-4153F elevated cost and also intending to evaluate BeagleBone Black in a real operation environment, we started a new local controller design. Main project policy was to run practically the same software as we do on Advantech SBCs, but using a computer which costs about a tenth. Powered with Texas Instruments AM335x processor, BeagleBone Black comes with a Debian Linux core and other two auxiliary real-time cores, the Programmable Real-time Units (PRUs).

HARDWARE DESIGN

We made a simple “carrier PCB” design for BeagleBone Black, just placing in a Eurocard-sized board digital buffers between BeagleBone Black GPIO pins and the LOCO bus signals on crate backplane. The board also has a counter, used during synchronized operations over power supplies (energy ramp and magnets cycling), a 7-segment display for status indication (just as it was since the first generation of local controller boards) and a reset monitoring circuit. Figure 1 shows the design, mounted with panel and handle.



Figure 1: The new local controller prototype.

EMBEDDED SOFTWARE OVERVIEW

General-purpose embedded software for the local controller based on BeagleBone Black is an adaptation of Advantech SBC PROSAC. Porting the software to the new hardware platform involved some tasks.

First of all, we modified the low-level routines used to read and write data through the 8-bit LOCO bus. In the previous design, LOCO bus was accessed through SBC's PC/104 bus (ISA). With BeagleBone Black, bus crate is managed with its GPIO pins. Although we have moved to a quite different hardware approach, the corresponding software changes are minimal. Only a few lines of a C source code file had to be changed.

Moreover, we made minor changes in the code, adapting it from FreeBSD to Linux.

PROSAC Threads Structure

PROSAC runs four threads, described below.

- Main thread: launches the other threads and periodically updates a 7-segment display, which shows local controller status.
- Reader thread: reads continually all inputs from the I/O cards, storing these values in RAM memory.
- Networker thread: deal with client I/O operations through a TCP/IP socket. When the client only wants to read the inputs of the cards, the last values obtained by the reader thread (stored in RAM) are retrieved.
- Interrupter thread: this thread is active when the controller is performing synchronized operations over power supplies. Operation of this thread assumes that controller has in RAM memory a waveform, which is point-by-point traversed each time it receives a trigger pulse from UVX timing system.

Interrupter Thread Detailed

Since the third generation of UVX local controllers, interrupter thread operation is based on a hardware counter, which accumulates the number of pulses received from the timing system since the last actuation and can be read or reset by software. This implementation strategy was adopted because neither FreeBSD nor Linux are real-time systems.

For UVX power supplies synchronized operation, we consider that if the controller can actuate between two successive pulses of the timing system, then it works in a good way. So interrupter thread should always read a value equals to 0 or 1 from the hardware counter while polling its status. A reading greater than 1 means that the controller missed at least one in-time actuation. While designing a new controller, we always try to keep the number of lost pulses equals to zero or as small as possible during synchronized operations.

TESTS AND THREADS TUNING

With the BeagleBone Black “carrier board” prototype, we first tested our embedded software to make sure that main, reader and networker threads were operating correctly. During these first tests, board's 7-segment display worked properly and PROSAC could read all I/O cards currently used in machine operation. Also, tests performed with a standard test client showed that all commands defined in our communication protocol between operations room and embedded controllers worked perfectly.

First tests of synchronized operation of the controller exhibited many situations of loss of pulses. These tests were performed in workbench with pulse trains of most common frequencies used in UVX.

One could think about using BeagleBone Black PRUs to solve the issue. But this was promptly discarded, as it would require a complete restructuring of PROSAC code. In fact, we intend to use the PRUs only for new software developments. And here we only wanted to port an existing software to a new hardware platform.

In order to evaluate and improve interrupter thread performance under different trigger frequencies, we considered a series of tests exploring PROSAC configuration parameters and important aspects of the embedded operational system (Linux), such as its kernel configuration and scheduling policies and priorities.

Once traditional Linux kernel is not capable of meeting hard real-time requirements [5], our first idea was to apply PREEMPT_RT patch [6] to the kernel. Besides, scheduling policy of all PROSAC threads was set to the real-time option SCHED_FIFO, and their priorities were defined so that the interrupter thread had the biggest priority. Therefore, it could use the CPU without being preempted by any other thread until it explicitly yields the processor. Additionally, we set BeagleBone Black CPU frequency to 1 GHz (maximum allowed), modifying its default CPU frequency scaling configuration. Unfortunately, this proposal didn't lead to expected results, and controller continued to loss timing system pulses despite all these efforts. With the traditional Linux kernel, SCHED_FIFO scheduling policy wasn't able to decrease the number of lost pulses too.

Given the failure of PREEMPT_RT patch combined with a real-time scheduling policy approach, our second try was to keep the traditional Linux kernel and use its Completely Fair Scheduler (CFS), while running BeagleBone Black CPU at 1 GHz. In opposition to the scheduler of the first proposed solution, which allows a process to retain the processor as long as it wants, CFS tries to give a fair amount of CPU usage to each process or thread, taking into account a parameter called nice value.

Threads with smaller nice values are allowed by the scheduler to consume more processing time. For this reason, setting interrupter thread niceness to the minimum (-20) grants more CPU time for this thread only when it effectively needs, and does not starve the other threads out completely. Furthermore, we fixed main thread niceness to the maximum allowed (+19), since it requires no important processing at all (it just refreshes the 7-segment display), and reader thread niceness to +15, as it spends the majority of its processing time waiting for inputs.

Using this approach, we varied networker thread nice parameter and the number of I/O cards connected to the test crate in order to investigate the impact of periodically bus readings and client TCP/IP requests over system's overall performance. In general, results were very good. Number of lost pulses changed with the number of I/O boards in the crate. Variations on networker thread nice value didn't influence it. The only disadvantage of this configuration is that crate boards reading rate was degraded during synchronized operation. For instance, we obtained reading rates as small as 7 Hz in some tests.

In order to increase the number of input readings during synchronized operation, a new solution was proposed. We kept nice values of interrupter, main and reader threads as before, set niceness of networker thread to +10 and changed the way reader thread works. When under normal operation, reader thread competes for CPU time with all the others. Under synchronized operation, it is disabled, and a complete reading of all inputs is performed after

every controller actuation, inside interrupter thread. As a result, we achieved a reading rate equal to the frequency of received pulses, without increasing the number of lost pulses. Table 1 summarizes obtained results for two trigger frequencies. Tests were performed with 2 I/O cards on crate and a TCP/IP client requesting 1000 readings per second.

Table 1: Interrupter Thread Performance

Frequency (Hz)	Total pulses	Lost pulses
512	5,376,065	8 (0.0001 %)
1000	6,050,225	18 (0.0003 %)

CONCLUSION

Tests showed that BeagleBone Black can serve as a hardware platform for UVX local controllers. The new design has an acceptable performance during synchronized operation, comparable to that of the previous one. We have already put in operation two local controllers based on BeagleBone Black. Their functioning is satisfactory. A batch of printed circuit boards was ordered for design replication.

We are also considering to use another SBC in UVX controller modules: BeagleBone Green, which is fully compatible and cheaper than BeagleBone Black. The main difference between these two boards is the lack of a video output interface on BeagleBone Green. Since a video interface is completely useless for our applications, this is not a problem. We have tested in workbench the new UVX local controller prototype with BeagleBone Green too, and results obtained are the same.

REFERENCES

- [1] BeagleBoard.org, <http://beagleboard.org>
- [2] J. P. S. Martins *et al.*, "Sirius control system: design, implementation strategy and measured performance", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 456-459.
- [3] J. G. Franco *et al.*, "LNLS control system", in *Proc. ICALEPCS'99*, Trieste, Italy, Oct. 1999, pp. 651-653.
- [4] J. G. R. S. Franco *et al.*, "Upgrading the LNLS control system from a proprietary to a commercial communications environment", in *Proc. EPAC'04*, Lucerne, Switzerland, Jul. 2004, pp. 530-532.
- [5] R. Love, "Process scheduling", in *Linux Kernel Development*: Addison-Wesley, 2010, pp. 64-65.
- [6] Real-Time Linux, <https://wiki.linuxfoundation.org/realtime/start>

openMMC: AN OPEN SOURCE MODULAR FIRMWARE FOR BOARD MANAGEMENT

H. A. Silva*, G. B. M. Bruno, LMLS, Campinas, Brazil

Abstract

openMMC is an open source firmware designed for board management in MicroTCA systems. It has a modular architecture providing decoupling between application, board and microcontroller-specific routines, making it useful as a base for many different designs, even those using less powerful controllers. Despite being developed in a MicroTCA context, the firmware can be easily adapted to other hardware platforms and communication protocols. The firmware is based on the FreeRTOS operating system, over which each monitoring function (sensors, LEDs, Payload management, etc) runs its own independent task. The OS, despite its reduced footprint, also provides numerous tools for reliable communication among the tasks, controlling the board efficiently.

INTRODUCTION

LMLS Beam Diagnostics team is currently developing the Sirius' Beam Position Monitor (BPM) electronics and has adopted the MicroTCA.4® standard by PICMG [1] in its boards designs [2].

The main board on the electron BPM system is the AMC FMC Carrier (AFC) [3], which is a general purpose FPGA board that hosts up to two mezzanine cards in FMC form factor. Those smaller cards carried by AFC can be implemented to have many different applications (e.g. fast digitizers, SFP modules, data pre-processing modules, RS485 communication). In the BPM application, fast digitizer FMCs will be used in order to read the BPM analog signals.

AMC boards, as the application boards are called in the MicroTCA system, must have implemented a Module Management Controller (MMC), which usually is a microcontroller responsible for monitoring the board health and acting as a communication channel between the system manager and application using Intelligent Platform Management Interface (IPMI) [4].

openMMC was created to be an open source (using GPLv3 license) modular and generic firmware, easily portable to other platforms. It runs over FreeRTOS, which gives the developer a wide set of tools to implement complex monitoring functions or advanced hardware control. Given its modular independent structure, it is possible to use the firmware in applications outside MicroTCA environment with little effort, changing the communication protocol in its lower layers, for example.

The project development is being versioned in a GitHub repository [5], using pull requests and issues tracking as its main collaboration tools.

* henrique.silva@lnls.br

MOTIVATION

The development of openMMC started after some unfruitful tests with the available open source MMC implementations. Those firmwares were developed to run on specific target boards, requiring a substantial effort in order to port them to AFC's hardware and begin a functional evaluation process.

After some attempts, it was clear that porting the code basically meant to rewrite it from scratch, given that its low level driver and application functions were deeply intertwined.

The hardware flexibility offered by MicroTCA was not being accompanied by its MMC firmware architecture, since each board implementation had to recreate the managing firmware. openMMC was thought to be the hardware independent firmware that could meet this need.

FreeRTOS

FreeRTOS [6] is a popular open source real time operating system for embedded controllers that has already been ported to a wide range of CPU architectures.

It features a preemptive scheduler that allows the application code to run multiple tasks in parallel with a single core. The scheduler decides which task will run based on its priority. More important tasks are always executed first, whilst blocking the lower priority ones. If one or more tasks are on the same priority level, a round-robin time slice method is applied, ensuring that all of them are executed within its time limits.

Most of OS-native tools used in communication between tasks are also implemented on FreeRTOS (e.g. semaphores, software timers, queues). The developer can also use some unique functions provided, such as direct-to-task notifications, event groups and co-routines.

Except for task creation, all tools on FreeRTOS can be stripped from its compilation, reducing resource usage, thus enabling use of cheaper and lower-power microcontrollers.

FreeRTOS project uses a modified GPLv3 license, which allows the user to implement its application on top of the OS without having to publish proprietary code [7].

FIRMWARE STRUCTURE

The firmware was structured in order to be easy to upgrade and port to different boards and controllers. Therefore four different abstraction layers are implemented: *Application*, *Hardware Abstraction*, *Port* and *Driver*, arranged as in Fig. 1.

Application

The Application layer holds high-level tasks responsible for deciding which action will be taken based on the in-

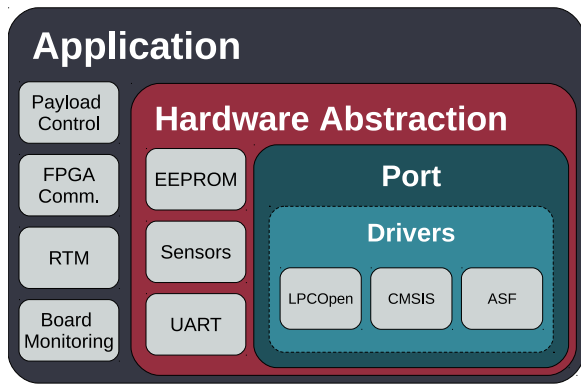


Figure 1: openMMC firmware structure.

formation provided by the lower level layers. Most of the functions implemented on this level are state-machine based, regularly checking and actuating on the board status.

Hardware Abstraction

In the Hardware Abstraction Layer (HAL) lies all the functions that interfaces with the board peripheral hardware, making the internal calls to the IC registers transparent to the caller for example. Since any board can benefit from a module in this layer, they are all stored together inside "modules" folder.

An exception in the HAL is the IPMI Protocol module, that does not target any specific peripheral hardware. Instead, it is responsible to manage IPMI protocol messages internally, decoding packets, asserting checksums and building responses accordingly.

The following modules were already implemented and tested in LNLS' AFC board:

- IPMI Protocol
- Watchdog Timer
- SCANSTA111 JTAG Switch
- ADN4604 Clock Crossbar Switch
- AD84XX DAC
- AT24MAC EEPROM
- 24xx64 EEPROM
- Hotswap Handle
- LM75 Temperature Sensor
- MAX6642 Temperature Sensor
- INA220 Voltage and Current Sensor
- HPM Upgrade
- PCA9554 I/O Expander
- Si57X Oscillator
- UART Debug interface

Port

The Port layer serves as a connection between *Driver* and the upper layers. Its purpose is to mask all functions provided by the controller drivers so that the upper layers only call generic functions defined in this level, as demonstrated in Fig. 2.

This layer presents a few restrictions regarding its structure. Developers in charge of implementing new hardware

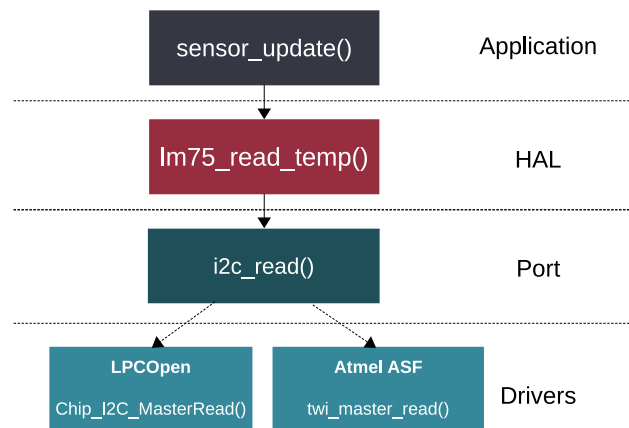


Figure 2: openMMC sensor update function call sequence.

drivers must follow the port layer functions' signature in order to maintain compatibility throughout the firmware.

Currently, there is only one port maintained in mainline openMMC, which targets LPC17 family chips. Development of a port to ATxMega128 CPU has been initiated by GSI [8].

Drivers

The lowest layer is where the microcontroller peripheral drivers are implemented, accessing directly the hardware to control the outputs.

BOOTLOADER

Having a small footprint makes it possible to store more than one image of the running firmware, easing the system's upgrade process. Taking advantage of this property, a Bootloader scheme was developed in order to manage the controller's memory sections and reserve a portion of the ROM to store an updated firmware. An example of the internal memory division in a LPC1764 controller can be seen in Fig. 3.

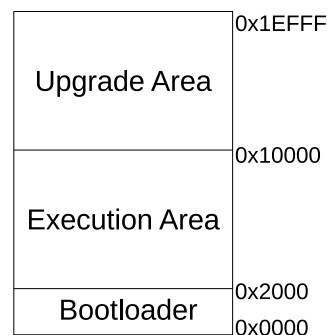


Figure 3: LPC1764 ROM organization with openMMC.

The new firmware is sent to the MMC broken in small packages using HPM update protocol [9], defined by PICMG. The controller then writes page by page the new firmware in the reserved ROM section and also adds an update flag to the bootlader, indicating that this firmware is the more recent version and should be moved to the proper section in order to be executed.

The bootlader is always executed first in power-up sequence, checking whether there is a new firmware to write into execution section. If not, it just moves the program execution flow to the starting point of the user code area.

BUILD AUTOMATION

CMake is a software designed to automate build systems [10], widely used in projects where the compilation chain has more than one build target and needs to manipulate multiple sources. It implements its own scripting language that allows the user to fully customize each build without having to worry about sources dependencies.

Using this tool with openMMC allows developers to fit the firmware to the board hardware just by adding new modules in the compilation list present on each board port folder. The user is also able to set new compilation flags or link new libraries almost effortlessly.

BOARD PORTING

openMMC also allows the user to easily port the firmware to a different board with other peripheral hardware configuration while keeping the target controller.

Since all peripheral hardware drivers are implemented as modules in the *Hardware Abstraction Layer*, only the `CMakeLists.txt` file inside the board folder have to be changed. In this file, all modules needed in each hardware configuration are selected based on a list parameter. When CMake is called to generate the automatic Makefiles, it parses this argument and includes all selected modules sources and dependencies in the compilation chain.

CODE SIZE

openMMC has a small memory footprint, which makes it perfect for embedded environments that traditionally have limited resources. Table 1 presents openMMC's average RAM usage and code size in ROM after compiling in both minimal and standard configurations for AFC BPM board, using GCC's optimization flag `-Os`.

Table 1: openMMC Resources Usage

Configuration	RAM [kB]	ROM [kB]
Minimal	5.54	20.91
Standard	8.07	29.69

CONTINUOUS INTEGRATION

Every modification in the application layer must be tested across all boards and controllers ports to assert that the firmware remains compatible.

Validating that the firmware compiles with every variant option becomes very difficult as the ports count increases. Using a Continuous Integration tool allows the maintainer to run dedicated tests after each commit on the main repository and automatically reports if the latest change has made any port unusable. Travis C.I. [11] uses a simple script written in its own language to install the needed toolchains. Tracking

the latest commits on the repository branches, allows it to perform build tests on them, ensuring that the latest modification is compatible with all board and controller ports.

DOCUMENTATION

The firmware documentation is written using Doxygen style [12]. This way, the documentation format is completely automated and can have many different output formats such as HTML, LaTeX, XML, RTF, etc.

By hosting the firmware code in GitHub repository, one is able to use a feature called GitHub Pages, in which a small website for generic use can be hosted simply by adding a new branch named *gh-pages* in the main repository. openMMC documentation has been uploaded to GitHub pages, but it is still under development [13].

Using this tool simplifies code documentation update, since it just needs to be regenerated by Doxygen after each change and pulled to its respective GitHub Pages branch.

ACKNOWLEDGMENTS

The authors would like to thank K. Macias from Creotech Instruments SA for the extensive firmware debug and P. Miedzik from GSI for the initial discussions in the firmware structure definition and by suggesting FreeRTOS as a base for the project.

REFERENCES

- [1] *PICMG MicroTCA Specification*, <https://www.picmg.org/openstandards/microtca>
- [2] D. O. Tavares *et al.*, "Development of an Open-Source Hardware Platform for Sirius BPM and Orbit Feedback", in *Proc. ICALEPCS'13*, San Francisco, October 2013, p. 1039, 2013.
- [3] AMC FMC Carrier Project, <http://www.ohwr.org/projects/afc>
- [4] *Inteligent Platform Management Interface*, <http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-home.html>
- [5] openMMC GitHub Repository, <https://github.com/lnls-dig/openMMC/>
- [6] FreeRTOS Project Page, <http://www.freertos.org/>
- [7] FreeRTOS modified GPLv3 license, <http://www.freertos.org/license.txt>
- [8] GSI MMC implementation repository, <https://github.com/qermit/JAMMCI/>
- [9] Hardware Platform Management Overview, <https://www.picmg.org/openstandards/hardware-platform-management/>
- [10] CMake Project, <https://cmake.org/>
- [11] *Travis Continuous Integration tool*, <https://travis-ci.org/>
- [12] Doxygen Project, <http://www.stack.nl/~dimitri/doxygen/>
- [13] openMMC Documentation at GitHub Pages, <http://lnls-dig.github.io/openMMC/>

IMPLEMENTATION OF A PRECISION LOGARITHMIC AMMETER

W. R. de Araujo*, M. R. Bissiano, G. Paulino, D. Galante, LNLS, Campinas, SP, Brazil

Abstract

A precision ammeter is in development for the acquisition of sensor signals such as photodiodes, gold mesh (by photoelectron effect) and ionization chambers. One of the problems of conventional ammeters is the automatic scale selection, which hinders many measurements performed in ample energy ranges. The ammeter in development is based on a different methodology than present on most commercial systems, using a logarithmic amplifier. This choice can provide a logarithmic response output in the range of pico to milli-amperes. The electronic board is in development by Brazilian Synchrotron Light Laboratory (LNLS), and it is being installed and tested at the Toroidal Grating Monochromator (TGM) Beamline.

BASIC OF LOGARITHMIC AMPLIFIER

A logarithmic amplifier (LogAmp) is a device that can express the output as a logarithmic function of the input, either in electrical current or voltage. This device is commonly used in telecommunications for compression of voice and video signals, and here it is explored its potential for measuring and monitoring currents produced in light sensors. [1]

The most common LogAmp uses the exponential curve of a bipolar junction transistor (BJT) to convert an input current in a logarithmic output voltage. This class of LogAmp usually operates on unipolar inputs and it is very sensitive to temperature variations. The most simplified model is composed by a NPN transistor connected at the negative feedback of an operational amplifier, as shown outlined in Figure 1. The transfer function of this circuit is:

$$V_{out1} = -V_{BE} \approx V_t \ln\left(\frac{I_{IN}}{I_s}\right) \quad (1)$$

Where I_s is the reverse saturation current and V_t is the thermal voltage. The first is a constant dependent of the construction parameters, materials, and, as consequence, temperature of operation. The thermal voltage is a universal parameter of transistors, and depends only on the temperature. The approximation shown above can be used due to the fact that the input current in the OpAmp is approximately zero.

A first solution to reduce the dependence of temperature is introducing a second LogAmp, which acts like a reference to the first through a subtracting circuit, as shown in the Figure 1.

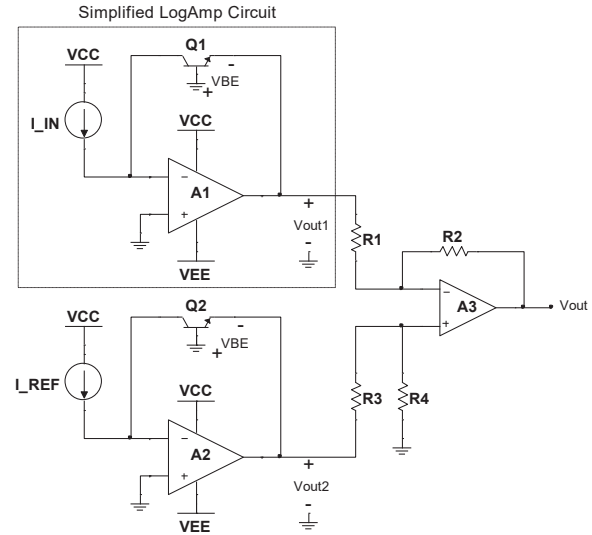


Figure 1: A referenced current LogAmp is shown above. A simplified model of a LogAmp circuit is outlined. The subtraction of two simple LogAmps with the third OpAmp (A3) eliminates the temperature dependence generated by I_s .

The general transfer function of the above circuit is given by:

$$V_{out} = V_{LOG} = V'_y \ln\left(\frac{I_{IN}}{I_s}\right) - V'_y \ln\left(\frac{I_{REF}}{I_s}\right)$$

$$V_{LOG} = V_y \log\left(\frac{I_{IN}}{I_z}\right) \quad (2)$$

Where V_y is a function of V_t and the gain attributed of the resistor in the subtractor circuit, and I_z is a conversion name to I_{ref} . This approach is able to eliminate the temperature dependence caused by I_s because the transistors have the almost identical properties and are in close thermal contact for proper cancellation. However, V_y is proportional to V_t , and so, it is still sensitive to temperature variations. By adding subsequent temperature-compensation circuitry this dependency is virtually eliminated (normally, an additional OpAmp amplifier stage with a resistive temperature detector [RTD], or similar device, is incorporated as part of the gain) [1]. Several commercial integrated circuits have embedded some type of temperature compensation which can stabilize the final gain V_y of the system.

DEVELOPMENT OF THE CONCEPT

For the development of the Logarithmic Ammeter (Log-Ammeter), it was designed an electronic based on a commercial LogAmp IC, with a range of 200 dB. This circuit has an internal reference current of 100 nA and a temperature stabilizing embedded in the chip. The system

* william.araujo@lnls.br

was designed to produce an output voltage from -1 V to 1 V for an input from 1 pA to 10 mA. To elaborate all the tests, it was used an experimental setup like depicted on Figure 2.

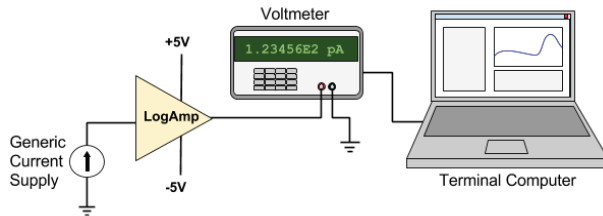


Figure 2: Schematic of the experimental setup for LogAmp tests. All of the experiments were developed in this configuration. The communication between the Terminal Computer and the Voltmeter is made by the IOCs from the EPICS system. The Generic Current Supply can be any device; a commercial supply, a photodiode or a voltage power supply in series with output resistors.

First it was necessary to characterize the experimental parameters V_y and I_z . Utilizing a Femtoamp Remote Sourcemeter 6430 from Keithley as a current supply, it was possible obtain the curve I-to-V characteristic of the LogAmp (Figure 3). Manipulating the Eq. (2), it was obtained the linear relation:

$$V_{\text{Log}} = A \log(I_z) + B \quad (3)$$

Where $A = V_y$ and $B = -V_y \cdot \log(I_z)$. Then, varying the input current of the LogAmp, it was obtained the curve presented on Figure 3. From its, determine the parameters $V_y = (0.2015 \pm 0.0006)$ V/decade and $I_z = (115 \pm 8)$ nA could be determined.

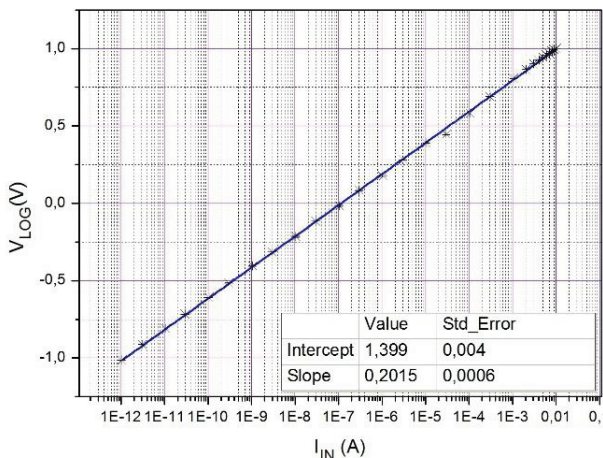


Figure 3: Relation IIN-to-VLOG of the LogAmp. The slope value corresponds to the parameter V_y , and from the Eq. (3), $I_z = (115 \pm 8)$ nA.

After this characterization by a controlled input current, it was possible to determine a calibration equation of the LogAmp, and so, monitoring currents from other experiments. For the next steps of the tests, it will be assumed the absolute values from the parameters, $V_y = 0.2015$ V/decade and $I_z = 115$ nA.

TESTS MADE AT THE TGM BEAMLINE

Initially, the proposed to create a logarithmic amplifier is for the use in the beamlines in the Brazilian Synchrotron Light Laboratory (LNLS). The second part of the experiments consists in make tests at the Toroidal Grating Monochromator (TGM) Beamline using the LogAmp with the aim to validate the technique.

The TGM beamline operates in the Vacuum-Ultraviolet (VUV) energy range, from 3 eV to 330 eV through three grades [2]. The present results were performed by the grade 1 (3 eV to 13 eV).

Energy Scan

A basic test of the TGM Beamline is a scan varying the energy on the monochromator. The monochromatic synchrotron radiation focuses on a diagnostic photodiode and energy is changed by moving the diffraction grating, producing a characteristic profile of the response of photodiode/efficiency of the grating. For this test, the current supply was replaced by the diagnostic photodiode. The values of energy and voltage were registered by the terminal computer from Experimental Physics and Industrial Control System (EPICS). A datasheet was created. It was compared the performance of the LogAmp and a commercial ammeter normally used on the beamline. For each instrument, fifteen scans were recorded and it was made the average of the curves. The results are shown in the Figure 4.

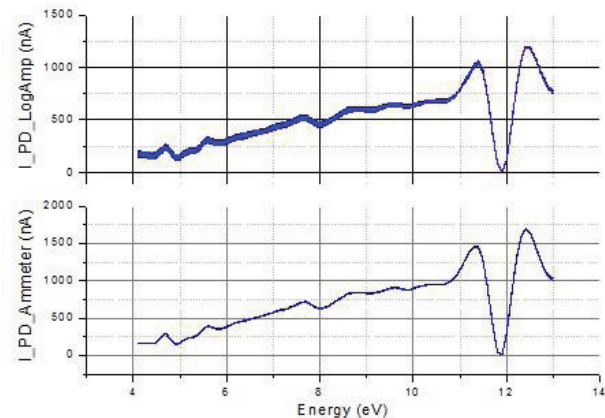


Figure 4: Energy scan for the 1st grating (3 eV to 13 eV). Each graph refers to the average curve of an instrument with standard error. The upper curve is the profile obtained by the use of the LogAmp, while the lower curve was obtained by the commercial ammeter.

From Figure 4 it is possible to see that the curve obtained from the LogAmp has a profile many similar to the obtained from the commercial ammeter. The noise, although small, present in the graph is due to the absence of any kind of filter in the input, which can make appear some fluctuations in the measured current. A difference of the values observed in the two curves occur because the curves was obtained for different alignments of beamline and ring current.

Total Electron Yield

After the scans in energy, the LogAmp system was used to measure a standard sample of TGM Beamline, to prove its concept in a real experiment. The Total Electron Yield (TEY) is a system that allows the indirect measure the absorbed photon flux as a function of its energy by measuring the reposition current driven to the sample after photoelectrons ejection when hit by the incident VUV and X-ray beam [3]. A current is generated on the sample holder and it is measured in an ammeter. An aluminum foil was fixed at the sample holder and the energy was varied like in the previously scan. Like the energy scan test, it was made TEY scans for a commercial ammeter and the LogAmp. Were realized seven scans for each instrument. The average curves are shown in the Figure 5.

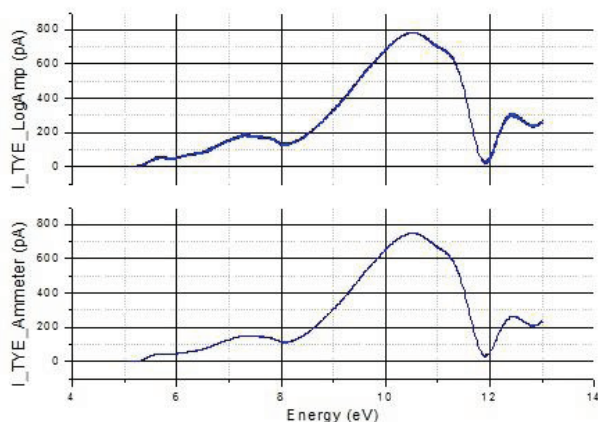


Figure 5: TEY scans for the grating 1 (3 eV to 13 eV). Each graph refers to the average curve of an instrument with standard errors. The upper curve is the profile obtained by the use of the LogAmp, while the low curve was obtained by the commercial ammeter.

For the TEY tests, it is possible to see that the values obtained is more precisely than the other case. The profile of the curves is practically the same, and the small difference between the curves are caused by a natural misalignment of the beamline and the decrease of the current in the storage ring. The TEY test is important to show how efficiently is the LogAmp in the picoampere range.

Motor Alignment

The last test realized at TGM Beamline intended to show how the LogAmp can cover a large range of currents without changing its scale. This test consists of use the LogAmp to monitor the signal in the diagnostic photodiode when the position of the first mirror is varied. The Figure 6 shows the current in the photodiode as a function of the motor position.

This graph shows the capacity of the LogAmp to vary 7 decades without change the scale of measurement. This property is really interesting because one of the mainly problem of the commercial ammeters is the switch time between two scales of measure.

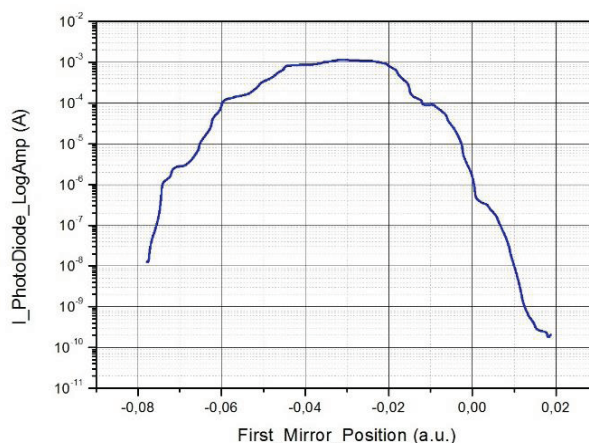


Figure 6: Current in the photodiode in function of the zenithal position of the first mirror. The first mirror position has an arbitrary unit.

CONCLUSION

The use of a logarithmic amplifier proved to be a good option to current measurements. The motor control test shows the capacity of very large range of currents without the necessity of change the scale. This is an advantage of the LogAmp when compared to other types of ammeters, which would allow the detection of events which vary the measured value some decades. The energy scans performed at the TGM Beamline show the performance of the LogAmp in a real experiment, proving the capacity, usability and efficiency of the LogAmp to be used at beamlines, even for measuring very low currents on real experiments of photo-absorption.

Simultaneously to the development of the LogAmp, it was implemented an algorithm of an Analog Digital Converter (ADC) to convert the output voltage of LogAmp in a digital current measure. The chosen ADC is in a Field-programmable gate array (FPGA) which contains a fourth order Σ - Δ modulator of 24-bits resolution for high precision measurement applications, with analog input in bipolar operation. The FPGA will be used to implement data transmission over Ethernet interface with EPICS driver. It will have a trigger input for synchronism with external events which allows to do measure as Fly-Scan [4], for example. Operation tests were proceeded with the ADC powered by a voltage source and also a rapid test of integrating ADC and LogAmp, both were successful.

The next steps in the development is the study filter design to stabilize the signals in the LogAmp, and the integration of amplifier and ADC and the with the terminal computer directly [5].

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Verônica Teixeira (TGM Beamline/LNLS), Prof. M.Sc. Afonso Alonso (FEEC/UNICAMP), Prof. Dr. Rangel Arthur (FT/UNICAMP), Eng. Douglas B. Beniz (LNLS) and Eng. Giovanni Franchi (Age Scientific) due to their great support in this project.

REFERENCES

- [1] “Integrated DC Logarithmic Amplifiers,” *Maxim Engineering Journal*, vol. 56, pp. 3-9, 2005.
- [2] TGM Beamline, <http://lnls.cnpem.br/beamlines/uvsoftx/toroidal-grating-monochromator-tgm/tgm>
- [3] Total Electron Yield, <http://lnls.cnpem.br/beamlines/xafs/equipments/tey>
- [4] G. B. Z. L. Moreno, F. P. O’Dowd, H. H. Slepicka, R. Bongers, and M. B. Cardoso, “On-the-fly scans for fast tomography at LNLS imaging beamline,” in *Proc. ICALEPCS’15*, Melbourne, Australia, Oct. 2015, paper THHB3O03.
- [5] M. Clifford, “Fundamental Principles Behind the Sigma-Delta ADC Topology,” Analog Devices Technical Article, 2016.

OPERATION EXPERIENCE AND MIGRATION OF I/O CONTROLLERS FOR J-PARC MAIN RING

N. Kamikubota[#], S. Yamada, K.I. Sato and N. Yamamoto, J-PARC, KEK & JAEA, Japan
T. Iitsuka, T. Aoyama and S. Yoshida, Kanto Information Service, Tsuchiura, Japan
H. Nemoto, ACMOS Inc., Ibaraki, Japan

Abstract

In 2008, when we started beam commissioning of J-PARC MR, I/O controllers (EPICS IOC) were consisted of about 80 pieces of VME-bus computers and a few PC-based controllers. The total number of IOCs was 80-90.

In 2016, we have non-VME IOCs: a) Yokogawa F3RP61 (Linux-based CPU with PLC I/O modules), b) virtual ioc (EPICS IOC on a virtual machine), and c) commercial micro-server. The total number of IOCs is around 170.

Histories and characteristics of VME and non-VME IOCs are described in this report. In addition, based on operation experience since 2008, reliability of VME-bus computers is discussed.

INTRODUCTION

J-PARC (Japan Proton Accelerator Research Complex) is a high-intensity proton accelerator complex, located in Ibaraki, Japan. It consists of three accelerators: a) 400-MeV Linac (LI), b) 3-GeV Rapid Cycling Synchrotron (RCS), and 30-GeV Main Ring (MR) [1-3]. The control system of J-PARC was developed using the EPICS (Experimental Physics and Industrial Control System) toolkit [4-5].

J-PARC MR started beam operation in 2008. Since then, the beam power of MR has been improved year by year. In June, 2016, the power reached 425kW (Figure 1). We continue beam commissioning toward the design goal, 750kW, and more [6].

To support commissioning activities, the MR control system has been often required to add new beam-diagnostic devices, or follow upgrade of existing power supplies, and so on, in a limited period. Thus, flexibility and extension ability are very important.

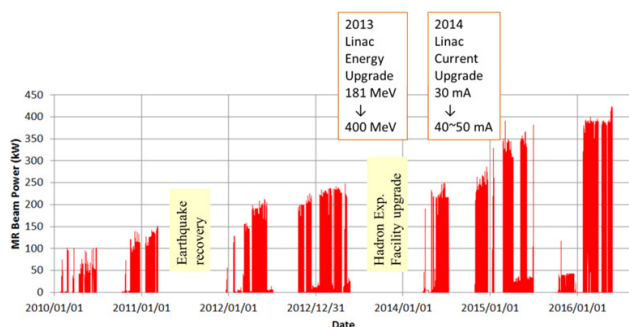


Figure 1: Improvement of MR beam power until 2016.

[#]norihiko.kamikubota@kek.jp

CHROICLE OF IOC IN J-PARC MR

Initial Design of IOC for J-PARC MR

The control system for J-PARC MR was constructed in 2006-2008. At the time, a VME-bus was understood to be robust and highly reliable, compared with other platforms. Thus, we selected VME-bus computer as a primary I/O controller (hereafter IOC). About 80 pieces of VME-bus IOCs were introduced in 2008.

However, in our history, non-VME-type IOCs were also introduced. Even in 2008, a PC-based controller was introduced. Then, a PLC-type Linux-based CPU in 2009, an IOC running on a virtual machine in 2011, and a commercial micro-server ("Saba" IOC) in 2015, were introduced. The numbers of MR IOCs between 2008 and 2016 are summarized in Figure 2.

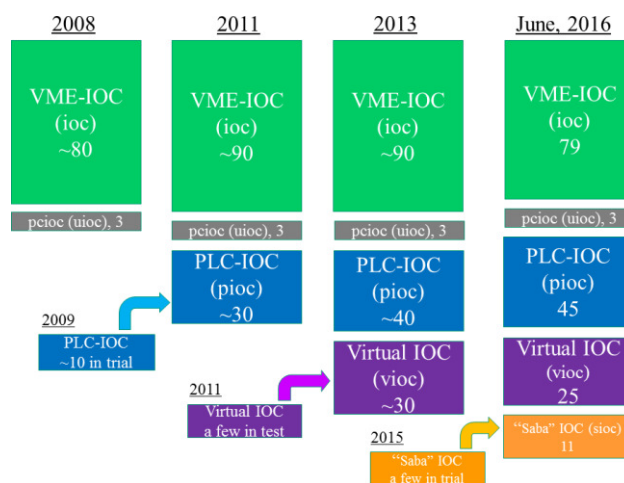


Figure 2: Numbers of MR IOCs between 2008 and 2016.

VME-IOC and PC-based IOC

In 2008, three (later four) models of VME-bus computers were used in MR operation: (a) the model, VMIC V7807 (after 2009, V7865), for DAQ of network-based digitizers (Yokogawa WE7000) [7], (b) the model, Sanritz SVA041, for controlling ladder-logic PLC (Yokogawa FA-M3) [8], and (c) the model, VMIC V7700, for VME-bus timing modules [9]. The specs of four models are given in Table 1. For all the models, we selected Intel-based chip, in order to run Linux OS. Front-end devices underneath VME-IOCs are shown in Figure 3.

One model of vacuum controllers has no external control interface but a RS485 port. It was not supported

by VME-IOC. Thus, we decided to introduce a “microIOC”, a customized PC-based controller provided by Cosylab [10]. A model with eight RS485 ports was introduced (Figure 4).

Table 1: VME-bus CPU modules in J-PARC MR

CPU Model	Device	Spec
VMIC V7807	Digitizer (TCP/IP)	Pentium M, 1.8GHz 1.0/1.5GB memory
VMIC V7865	Same as above # After 2009	Core Duo T2500 2C2T 2.0GHz / 3GB
Sanritz SVA041	Ladder PLC (TCP/IP)	ULV Celeron M 600MHz / 512MB
VMIC V7700	timing modules (VME-bus)	Celeron M 400MHz / 512MB

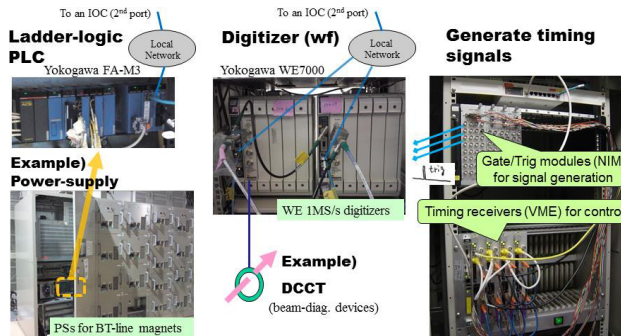


Figure 3: Front-end devices underneath VME-IOCs.

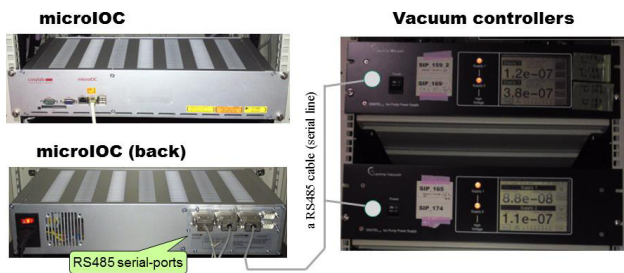


Figure 4: A microIOC and vacuum controllers.

PLC-IOC

In 2008, porting EPICS into F3RP61 (a Yokogawa’s Linux-based CPU for PLC) was carried out by KEK and RIKEN [11-12]. As in Figure 5, this enabled us a “PLC-IOC”, which uses robust PLC I/O modules without ladder programming and a separated VME-IOC. After 2009, many PLC-IOCs, each consists of a F3RP61 and suitable I/O modules, have been introduced in J-PARC MR [13-14]. A PLC-IOC for octapole magnets is a typical example (Figure 6).

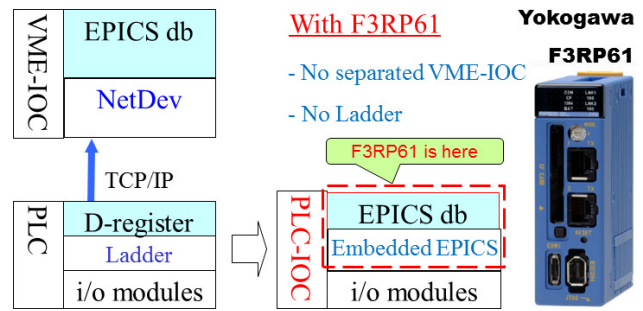


Figure 5: Scheme of PLC-IOC with a F3RP61.

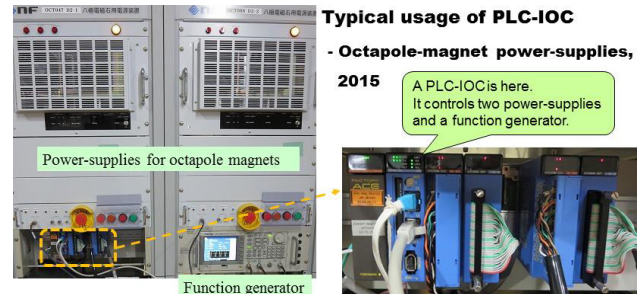


Figure 6: A PLC-IOC to control octapole magnets.

Virtual IOC

The “virtual IOC” is an EPICS IOC running on a virtual machine. Since 2011, virtual IOCs have been used for network-based devices, and soft-IOCs of management purposes. Three blade-type servers, with Scientific Linux 6.7 and KVM, have been used as host machines [15-16].

Example of virtual IOCs in MR are shown in Figure 7. The left is a network traffic monitor using the SNMP protocol. The right provides an automatic tuning of digitizer scales at parameters changes. In the right case, a virtual IOC communicates with three “real” IOCs.

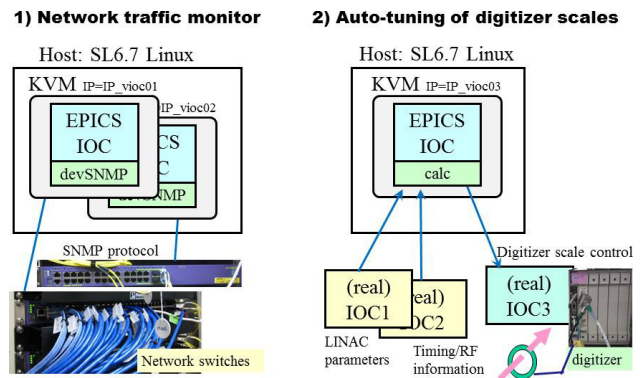


Figure 7: Two examples of virtual IOCs in J-PARC MR.

“Saba” IOC

The “Saba Taro” is a commercial micro-server. It is designed for administrative purposes (i.e. DNS, Web, DHCP, etc.) used in a small office, or for a host of IoT devices [17]. A “Saba” IOC is an EPICS-IOC, using a

“Saba Taro” of the model “Pinon Type-P” [18]. A “Saba” IOC is much cheaper and much smaller, however, much powerful than a VME-IOC. Figure 8 shows the spec of “Saba Taro” and some of VME-IOCs in comparison.

A VME-IOC for network-based devices can be replaced by a “Saba” IOC. In 2015, we carried out a reliability test of a few “Saba” IOCs in MR field areas. They ran without any trouble over a year. In 2016, we start a multi-year plan to replace VME-IOCs by “Saba” IOCs, gradually in the coming years (Figure 9).


Spec. comparisons between VME-IOCs and “Saba” IOC				A “Saba” IOC, EPICS running on a micro-server
種別	CPU	Memory	OS	
サバ太郎 Saba Taro	Celeron J1900 2~2.42GHz/4.C	8GB	SL6.8 SSD-Boot	
V7807 (VME SBC)	Pentium M 1.8GHz	1GB ~1.5GB	SL6.3 Network-Boot	
SVA041 (VME SBC)	Celeron M 600MHz	512MB	SL6.3 Network-Boot	

Figure 8: A “Saba” IOC and its spec.

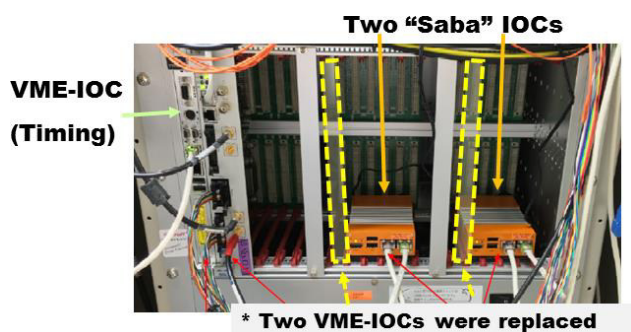


Figure 9: Two VME-IOCs were replaced by “Saba” IOCs.

DISCUSSION

Number of J-PARC MR IOCs in 2016

In June, 2016, numbers of various IOCs used in J-PARC MR operation are summarized in Table 2.

Table 2: List of IOCs in June, 2016 in J-PARC MR

IOC type	Number	Interface (Purpose)
VME-IOC (V7807, V7865,SVA041)	54	Network (digitizer, ladder-PLC)
VME-IOC (V7700)	25	VME-bus (Timing)
PC-based IOC	3	RS485 (Vacuum)
PLC-IOC	45	Basic signal I/O
Virtual IOC	25	Misc.
“Saba” IOC	11	Network

Reliability of VME-bus Computer

During 2008-2011, VME-IOCs worked well without troubles. Since 2011, each time after a scheduled power outage, a few VME-IOCs did not run. On-board memory

card was broken (Figure 10). After replacing the broken card by a spare, the VME-IOC worked again. Memory cards on the models SVA041 and V7807 were broken, however, no failure on other models. Number of broken cards at each power outage is shown in Figure 10.

Later, we knew that all broken memory cards are the products in the middle of 2007. We introduced 33 (22) pieces of SVA041 (V7807) in 2007. Until 2016, 20 of 33 memory cards on SVA041 (7 of 22 on V7807) were broken, respectively. More detailed analysis on this issue is given elsewhere [19].

In 2015, Micron, a DRAM company, announced a defect in DRAM chips which were shipped before December, 2010 [20]. Figure 11 is an essential part of the announcement. Observed our problem is well understood.

It is worth noting that no trouble on main VME boards of the models, SVA041 and V7700, since 2008. In addition, VME boards introduced not in the middle of 2007 have no memory card failure. However, this experience arise a suspicion for reliability of VME-bus computers, and encourages us to use “Saba” IOCs more.

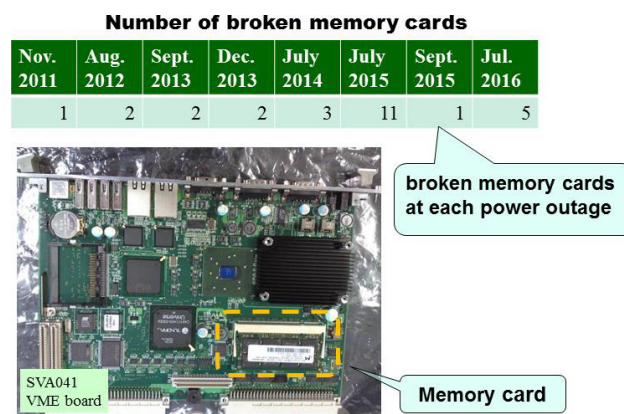


Figure 10: Memory card failures during 2011 and 2016.

Part of Announce CSN-37 by Micron, jan.2015

This customer service note describes a potential issue in certain legacy products that, under certain usage conditions over extended time periods, may result in the inability of a small percentage of the devices to properly power on after a power cycle event. The issue affected a limited subset of 95nm DDR1 and DDR2 products manufactured before December 2010. As has been previously discussed publicly, Micron has worked proac-

The root cause of the Micron memory component failure over time in certain use conditions is the degradation of a single transistor on the silicon chip. On a small percentage

Figure 11: Part of the announcement by Micron.

CONCLUSION

Various types of I/O controllers of J-PARC MR are reviewed. Based on the operation experiences between 2008 and 2016, reliability of VME-bus type I/O controller is discussed.

We appreciate all the MR staff members for their continuous efforts to maintain and update I/O controllers.

REFERENCES

- [1] J-PARC website: <http://j-parc.jp/index-e.html>

- [2] S. Nagamiya, "Introduction to J-PARC", *Prog. Theor. Exp. Phys.* 2012 02B001.
- [3] T. Koseki and K. Hasegawa, "Present Status of JPARC - After the Shutdown due to the Radioactive Material Leak Accident -", in *Proc. IPAC'14*, Dresden, Germany, paper THPME061, pp.3374-3375.
- [4] <http://www.aps.anl.gov/epics/>
- [5] N. Kamikubota *et al.*, "J-PARC Control toward Future Reliable Operation", in *Proc. ICALEPCS'11*, Grenoble, France, Oct. 2011, paper MOPMS026, pp. 378-381.
- [6] S. Igarashi, "Recent Progress of J-PARC MR Beam Commissioning and Operation", in *Proc. HB2016*, Malmo, Sweden, July 2016, paper MOAM6P60, pp. 21-26.
- [7] M. Takagi *et al.*, "Linux Support of WE7000 EPICS Driver for J-PARC MR", in *Proc. of 3rd Annual Meeting of Particle Accelerator Society of Japan*, Sendai, Japan, Aug. 2006, pp.448-450.
- [8] J. Odagiri *et al.*, "Development of EPICS Device/Driver Support Modules for Network-based Devices", in *Proc. of 3rd Annual Meeting of Particle Accelerator Society of Japan*, Sendai, Aug. 2006, pp.925-927.
- [9] N. Kamikubota *et al.*, "Operation Status of J-PARC Timing System and Future Plan", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, paper WEPGF121, pp.988-991.
- [10] <http://www.cosylab.com/microioc/>.
- [11] A. Uchiyama *et al.*, "Development of Embedded EPICS on F3RP61-2L", in *Proc. PCaPAC'08*, Ljubljana, Slovenia, Oct. 2009, paper WEX03, pp. 145-147.
- [12] J. Odagiri *et al.*, "Application of EPICS on F3RP61 to Accelerator Control", in *Proc. ICALEPCS'09*, Kobe, Japan, Oct. 2009, paper THD005, pp. 916-918.
- [13] M. Takagi *et al.*, "Control of the J-PARC Slow Extraction Line based on Embedded EPICS", in *Proc. ICALEPCS'09*, Kobe, Japan, Oct. 2009, paper WEP076, pp.549-551.
- [14] S. Motohashi *et al.*, "Data Acquisition System of Beam Loss Monitors of the J-PARC Main Ring", in *Proc. ICALEPCS'09*, Kobe, Japan, Oct. 2009, paper WEP070, pp. 537-539.
- [15] N. Kamikubota *et al.*, "Virtual IO Controller at JPARC MR using XEN", in *Proc. ICALEPCS'11*, Grenoble, France, Oct. 2011, paper WEPMU039, pp. 1165-1167.
- [16] N. Kamikubota *et al.*, "Experience of Virtual Machines in J-PARC MR Control", in *Proc. ICALEPCS'13*, San Francisco, CA, USA, Oct. 2013, paper MOPPC131, pp. 417-419.
- [17] http://www.pinon-pc.co.jp/hp/?page_id=102
- [18] S. Yamada, "Deployment of a tiny fanless server as IOC in J-PARC Main Ring", in *Proc. of 13th Annual Meeting of Particle Accelerator Society of Japan*, Chiba, Japan, Aug. 2016, MOP092, in press.
- [19] N. Kamikubota *et al.*, "J-PARC MR's Experience on VME-bus Computers in the Last Decade", in *Proc. of 13th Annual Meeting of Particle Accelerator Society of Japan*, Chiba, Aug. 2016, MOP095, in press.
- [20] https://www.micron.com/~media/documents/products/customer-service-note/csn37_95nm_legacy_dram.pdf

TIMING AND SYNCHRONIZATION AT FRIB*

M. Konrad[†]

Facility for Rare Isotope Beams, Michigan State University, East Lansing, MI 48824, USA

Abstract

The Facility for Rare Isotope Beams (FRIB) requires a timing system for distributing a common time base for time-stamping data as well as for triggering actions of multiple devices distributed over the machine. Three different technologies are used to accomplish these goals. An event system based on commercial off-the-shelf hardware made by Micro-Research Finland provides time-stamps and triggers to more than 500 fast data-acquisition devices like beam diagnostics electronics, LLRF controllers, and machine protection nodes. This system is also used to distribute FRIB's complex beam pulse patterns with event rates of more than 50 000 events per second. For many hundred devices which require a lower timing accuracy like programmable logic controllers and computers the Precision Time Protocol (PTP) is used. Additionally the Network Time Protocol is used for legacy devices that do not support PTP, yet. We describe the architecture of the FRIB timing system and how the different timing subsystems are synchronized. We also describe how FRIB's beam pulse patterns are generated.

INTRODUCTION

FRIB [1] is a project under cooperative agreement between US Department of Energy and Michigan State University (MSU). It is under construction on the campus of MSU and will be a new national user facility for nuclear physics. Its driver accelerator is designed to accelerate all stable ions to energies >200 MeV/u with beam power on the target up to 400 kW [2]. The FRIB driver linac requires a timing system for distributing a common time base for time-stamping data as well as for triggering actions of multiple devices distributed over the facility. Commissioning of the front-end as well as the core components of the timing system is currently underway. The remaining parts of the accelerator and the timing system are planned to be commissioned over the next two years.

ARCHITECTURE

The client devices connected to the FRIB timing system have very different timing requirements. They can be divided into the three accuracy classes listed in Table 1. Due to the large number of timing clients in each of the classes the use of different technologies helps to reduce cost. Devices in the "high" accuracy class are connected to an event system that broadcasts its events over a fiber network. This system is based on commercial off-the-shelf products by Micro-Research Finland [3]. For the "medium" accuracy class

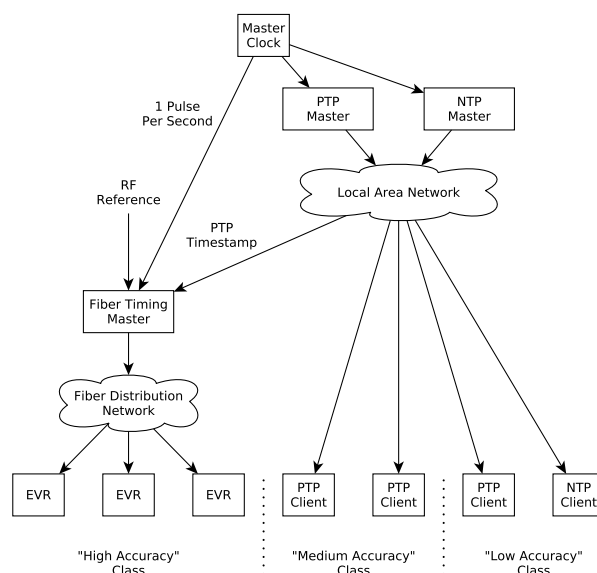


Figure 1: The FRIB timing system relies on three different technologies: A fiber-based event system as well as the Ethernet-based Precision Time Protocol (PTP) and Network Time Protocol (NTP). The three subsystems are synchronized to a common master clock.

the Precision Time Protocol (PTP version 2 as defined by the IEEE 1588-2008 standard) is used. In the last years this protocol has gained a lot of popularity for industrial automation applications and is widely supported by state-of-the-art programmable logic controllers. Additionally the Network Time Protocol (NTP) is supported. Compared to NTP the more modern PTP offers features like more efficient multicast messaging, cleaner handling of leap seconds as well as fault tolerance. This is why PTP is preferred over NTP even for clients in the "low" accuracy class. NTP is used only for devices that do not support PTP.

The event system, the PTP timing system and the NTP timing system can be considered three independent subsystems of FRIB's timing system. All three subsystems are synchronized to a common master clock to ensure data which has been time-tagged by devices connected to different timing subsystems, can be correlated (see Fig. 1).

EVENT SYSTEM

The event system is the most complex and most critical part of the FRIB timing system. In the following we will give a more detailed description of this subsystem.

* Work supported by the U. S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661

[†] konrad@frib.msu.edu

Table 1: Accuracy Requirements of Timing Client Devices at FRIB. Devices that have low accuracy requirements but require triggers are considered part of accuracy class “high”.

Accuracy Class	Typical Client Devices	Triggers Required	Estimated Number of Clients
“High” (≤ 100 ns)	Fast beam-diagnostics equipment, LLRF controllers	Yes	≈ 500
“Medium” (≤ 1 ms)	Programmable logic controllers, motor controllers	No	≈ 100
“Low” (≤ 1 s)	Computers	No	Hundreds

Table 2: Time Structure of the Beam during the Power Ramp up to 400 kW. The arrows correspond to a linear ramp.

Time (s)	Pulse Rate (kHz)	Pulse Length (μ s)
0–30	2 \rightarrow 25	0.6
30–60	25	0.6 \rightarrow 5
60–100	25	5 \rightarrow 20
100–500	25	20 \rightarrow 39.4

Synchronization with RF Systems

The RF master oscillator is synchronized to the timing system (see Fig. 2). An Event Receiver (EVR) card connected to the event system generates a one pulse-per-second signal which is used to discipline a low-noise Rubidium frequency standard. This device provides a 10 MHz signal that is used as a reference signal for a phase-locked oscillator with an output frequency of 10.0625 MHz. This frequency is distributed to all low-level RF controllers which derive their operating frequency (depending on the type of cavity 40.25 MHz, 80.5 MHz, 120.75 MHz, or 322 MHz) from this reference signal using an internal phase-locked oscillator. In a similar way the clock signal for the Event Generator (EVG) is derived from the RF reference signal. This approach ensures low jitter of the event system relative to the RF reference frequency.

Time Structure of the Beam

The FRIB driver linac requires a wide variety of beam-pulse patterns. During nominal 400 kW operation it is operated with a pulse rate of 100 Hz and a duty cycle of 99.5%. The average beam power can only be ramped up slowly to 400 kW to reduce thermal stress to the target. The ramp-up procedure consists of four phases during which the pulse rate and pulse width are increased as defined by Table 2. As soon as a phase has been completed the next phase starts automatically. After 500 s the power ramp-up is complete and the system proceeds to nominal operation.

A variety of pulsed operation modes have been defined for commissioning and beam tuning. Some of them can generate either a single shot triggered by the operator or periodic pulses with adjustable frequency and pulse length (within certain constraints).

FRIB’s timing patterns need to be expected to evolve over time. For example the power ramp-up procedure might get

optimized once measurements of the actual target temperature are available.

Due to the complexity of FRIB’s time structure we have decided to let the timing master broadcast “beam on” and “beam off” events. Thus timing clients do not need to have any knowledge about the pulse pattern. No large memories are required for storing pulse patterns in each timing client and no distribution system for pulse patterns is needed. On the other hand the large number of events that need to be played out in real time during the power ramp-up make this approach challenging on the timing master side.

Streaming of Event Data

The EVG provides two hardware sequencers that are implemented in its FPGA. They allow the EVG to play out a predefined series of events in real time. Each sequencer consists of a memory that holds up to 2048 time-stamp – event code pairs as well as control logic that compares the time-stamp of the next event in the list to the current time and sends out the event code if a match is detected. For most other accelerator facilities that are using the MRF timing system so far (mainly synchrotron light sources) 2048 time-stamp – event code pairs is sufficient but this is not enough to store the millions of time-stamp – event code pairs required for FRIB’s power ramp-up. To work around this limitation both sequencers are used alternately ensuring continuous operation. The sequence of events is broken into 10 ms chunks containing one machine cycle each. These chunks are loaded into the sequencers alternately. Even during the power ramp-up mode which requires up to $\approx 25\,000$ “beam on”/“beam off” event pairs per second the maximum number of events per machine cycle is ≈ 500 which leaves enough headroom for additional diagnostics events and other extensions.

During initialization both sequencers are configured for single-shot operation. A multiplexed counter generates a 100 Hz trigger signal that starts both sequencers in case they are armed. The “start of sequencer 1” interrupt is set up to trigger filling and arming of sequencer 2 (in this order). The “start of sequencer 2” interrupt triggers filling and arming of sequencer 1. To start the process, the events for the first cycle are loaded into sequencer 1 and it is armed.

Note that the sequencers are configured for single-shot operation which means they need to be re-armed before they will play out a sequence again. Only one of them is armed at a time and the other one will thus ignore the trigger. Also note that this behavior also makes the system fail in a

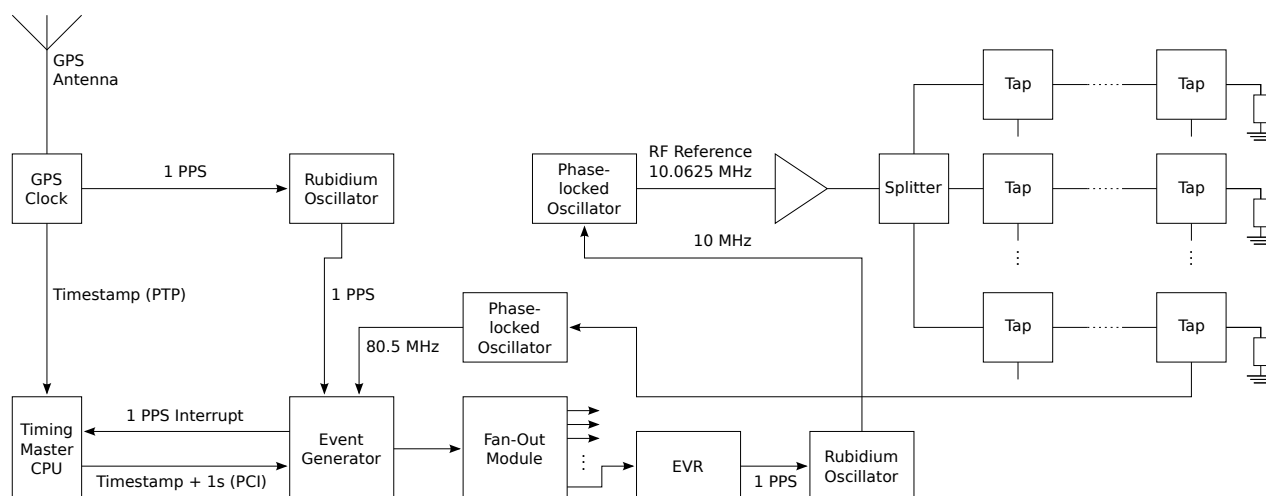


Figure 2: Synchronization of the event timing system to the GPS master clock as well as synchronization of the RF reference oscillator to the event system.

clean way: In case the computer crashes the EVG will finish playing out the loaded machine cycles. When finished the sequencers will not be re-armed and the EVG automatically stops broadcasting events. Breaking up the event stream into machine cycles ensures that the beam is always off when the EVG stops broadcasting events.

The streaming of the event data has been implemented on the timing master EPICS IOC using the mrfioc2 driver [4]. Recent code contributions to this driver by FRIB and Paul Scherrer Institut have added support for the cPCI-EVx-300 hardware series and the start-of-sequencer interrupts. Leveraging these features, continuous data streaming to the EVG's sequencers can now be implemented on the EPICS database level. Development of C/C++ code is not required.

Note that each time-stamp – event code array needs to be computed and downloaded into the EVG in less than 10 ms. This real-time constraint is being met reliably by our EPICS IOC running on a standard Intel CPU. With Debian GNU/Linux we are using the same operating system that is running on all other FRIB controls computers. However for this application a real-time kernel is being used.

Beam Scheduler

The time structure of the beam (“beam schedule”) is calculated in software by the EPICS Input/Output Controller (IOC) depending on the beam mode and the parameters selected by the operator. Depending on the beam mode the number of pulses as well as the pulse width are either fixed or provided by the operator directly or calculated dynamically for each machine cycle (e.g. during the power ramp-up). This behavior is implemented in the EPICS database to simplify online troubleshooting and to facilitate potential future modifications. For each machine cycle an aSub record generates the time-stamp – event code array based on the the number of pulses and the pulse width. This data is then downloaded into the hardware sequencers. The aSub record automatically distributes the beam pulses evenly over the

allowed period to spread out the heat load evenly on the rotating target.

A finite-state automaton implemented in the EPICS state-notation language defines the sequence of states for each beam mode (e.g. the different phases of the power ramp-up). It also handles exceptions like a trip of the machine protection system.

SUMMARY

By using the MRF timing system, PTP, and NTP, the FRIB timing system is able to provide time-stamps and triggers to many hundreds of clients at low cost. By synchronizing the three subsystems data time-stamped by different subsystems can still be correlated. The RF reference frequency is also synchronized to the timing system.

The event stream for the fiber-based timing system is generated in real time by the EPICS IOC of the timing master. The event data is then streamed into the EVG for real-time distribution. This allows the timing master to play out event sequences of arbitrary length and complexity without the need for custom firmware or the client devices having knowledge about the time structure of the beam. Changes to the time structure of the beam require only changes to the EPICS record database of the timing master making this solution very flexible.

REFERENCES

- [1] FRIB, <http://www.frib.msu.edu>
- [2] J. Wei *et al.*, “FRIB Accelerator: Design and Construction Status,” in *Proc. 13th Int. Conf. on Heavy Ion Accelerator Technology. (HIAT'15)*, Yokohama, Japan, September 2015, paper MOM1102, pp. 6–10.
- [3] Micro-Research Finland Oy, <http://mrf.fi/>
- [4] mrfioc2 device driver for Micro-Research Finland Hardware, <http://epics.sourceforge.net/mrfioc2/>

PROCESSING SPE FILES FROM PRINCETON INSTRUMENTS DURING DATA ACQUISITION IN LNLS

D. B. Beniz[†], Brazilian Synchrotron Light Laboratory, Campinas, Brazil

Abstract

DXAS (*Dispersive X-ray Absorption Spectroscopy*) [1] beamline of LNLS (*Brazilian Synchrotron Light Laboratory*) uses a Princeton Instruments CCD, PyLoN [2], to acquire spectra of materials under analysis. Such detector produces an SPE binary file which can be read by a Python script, WinspecUtils.py [3], extracting intensities information on a 2D matrix for each acquired frame and then processing them as a NumPy [4] array.

Using that, a procedure to analyse partial data while the experiment is being performed in DXAS beamline was developed in Python language for their experiments. In this article we will focus on XMCD (*X-ray Magnetic Circular Dichroism*) analysis, describing its motivation and main aspects of its implementation.

MOTIVATION

XMCD experiments in DXAS involve a complex setup of equipment. Characteristics of materials under analysis also contribute to a succeed experiment. The fact is that only after hours of spectra acquisition and more hours of data analysis such results are achieved, and then it could be too late to go back and change something on experiment environment or even in the sample itself.

With that in mind, a way to pre-analyse XMCD results during the experiment could save time and effort, helping to make a decision to make something different and maybe remove, or mitigate, injurious interferences on the experiment even before the end of scheduled time of beam usage by the researcher.

The option by Python language to elaborate scripts to achieve such aim was the natural decision in LNLS since it is being recently used in the laboratory at almost all their beamlines to orchestrate the control of devices during experiments, using an internally developed Python package called Py4Syn [5], which is also available for external community. Looking on the Internet by a Python tool that read SPE files, the first step on data analysis, WinspecUtils.py was found and its test was pretty satisfactory. So, we decided to adopt it and develop a procedure to process the data array extracted from the SPE file during the spectra acquisition.

DATA FLOW TO CALCULATE XMCD

Each XMCD cycle is a combination of eight spectra, or frames, which are acquired by PyLoN CCD when the sample were submitted to a magnetic field with a specific sequence of directions, as here: [+ - - + - + -], where “+” represents the positive direction of magnetic field and

“-” the negative one. To guarantee the synchronization between the applied magnetic field direction and the moment where a spectrum is acquired by the CCD, all the process is controlled by the same script, also in Python, using Py4Syn, which operates the power supplier connected to the magnetic coil and send the electric pulse to trigger the CCD acquisition.

At the end of a cycle, that is, every time a set of eight spectra are acquired, the XMCD is calculated. Figure 1 illustrates the data flow across the main data processing boxes.

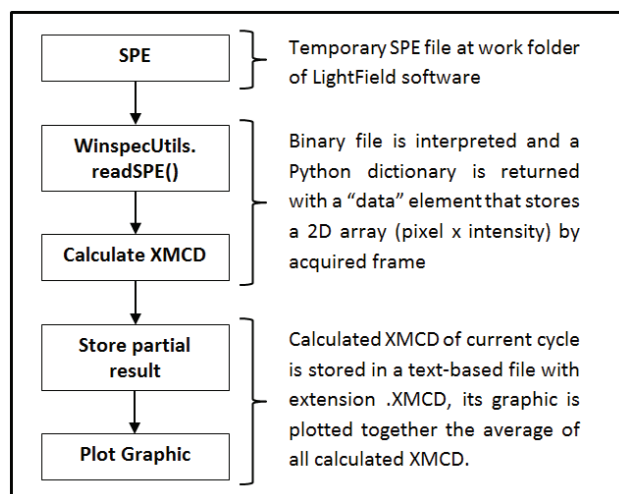


Figure 1: Data Flow to Calculate and Plot XMCD.

WinspecUtils.readSPE()

Based on the well formatted SPE binary file, and Princeton Instruments definition of each type and length of data on its header, readSPE() method of WinspecUtils.py script unpack each piece of data from it and return a Python dictionary with this structure:

```

spedict = { 'data':[],
            'IGAIN': pimaxGain,
            'EXPOSURE': exp_sec,
            'SPEFNAME': spefilename,
            'OBSDATE': date,
            'CHIPTemp': detectorTemperature,
            'COMMENTS': comments,
            'XCALIB': xcalib,
            'ACCUMULATIONS': accumulations }
  
```

The most important to us at this moment is the ‘data’ element. It will store in the first dimension a set of spectra, or frames, and of each spectra a 2D matrix of pixel x intensity.

[†] douglas.beniz@lnls.br

Maximum number of pixels for PyLoN CCD is 2048, and the intensities for each pixel depends on some variables like beam intensity, beamline monochromator selected energy, distance of CCD to the sample, among others.

Calculate XMCD

With the information of each spectrum of a cycle it is possible to calculate the XMCD. Its formula is presented by Eq. (1), which indicates the difference between the averages of calculated absorbance for each group of acquired spectra, where “NORM+” and “NORM-” indicate, respectively, normalized spectra acquired when magnetic field was positive and when it was negative. Eq. (2) presents the calculation of absorbance (μ), where “I” is the radiant flux transmitted by the material under analysis, and “I0” is the radiant flux it receives.

$$XMCD = \langle \mu_{NORM+} \rangle - \langle \mu_{NORM-} \rangle \quad (1)$$

$$\mu = \ln \left(\frac{I}{I0} \right) \quad (2)$$

As the I0 in DXAS beamline can only be acquired by removing the sample from the front of the beam, it needs to be collected before the dichroism experiment, and generated SPE of I0 is one of the inputs for the calculation. WinspecUtils.readSPE() method is also used to gather its data.

NumPy functions are used to facilitate the mathematic calculation of those intermediate and final results manipulating the data directly on arrays.

Storing Partial Results

The calculation results, that is, the XMCD values for all pixels (which are calibrated to correspondent energies in another process that is not presented here), are then stored in an ASCII text file with extension .xmcd. Such file contains information of energy (obtained from pixels and a calibration formula), calculated XMCD, extracted I, extracted I0, calculated μ and calculated μ^{NORM} .

Such files are identified as partial results.

Plotting Graphics

At every cycle processing, with calculation of correspondent XMCD, right after save a file with the partial results, a graphic with current cycle result and an average of all calculated cycles is presented using Matplotlib [6] package of Python. Figure 2 show an example of a graphic for a processed XMCD of an individual cycle and the total average of all previous cycles.

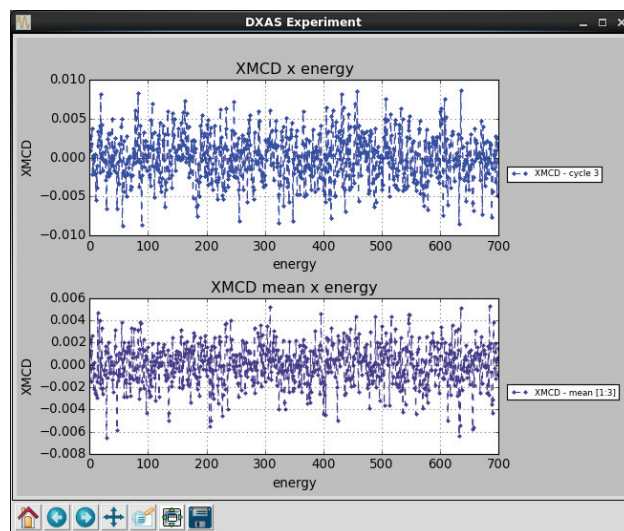


Figure 2: XMCD Graphic of one cycle and all the average.

This way, researchers can follow the behaviour of the material they are analysing and can quickly react if the partial results are totally incoherent with that they presumably should observe.

Finally, at the conclusion of the experiment, a set of ASCII text files is generated with all the calculations, like XMCD and absorbance (μ), for all cycles organized in columns.

REFERENCES

- [1] DXAS beamline of LNLS, <http://lnls.cnpem.br/beamlines/xaafs/beamlines/dxas>
- [2] Princeton Instruments PyLoN, <http://www.princetoninstruments.com/products/PyLoN-CCD>
- [3] WinspecUtils.py, https://github.com/kaseyrussell/python_misc_modules/blob/master/WinspecUtils.py
- [4] NumPy, <http://www.numpy.org>
- [5] Py4Syn, <http://py4syn.readthedocs.io/en/latest>
- [6] Matplotlib, <http://matplotlib.org>

HIGH LEVEL SOFTWARE FOR THE COMMISSIONING OF THE EUROPEAN XFEL

Lars Fröhlich*, Christopher Behrens, Bolko Beutner, Maria Elena Castro-Carballo, Winfried Decking, Olaf Hensler, Raimund Kammering, Torsten Limberg, Sascha Meykopff, Matthias Scholz, Jan Szczesny, Josef Wilgen, DESY, Hamburg, Germany
Enrico Allaria, Giuseppe Penco, Elettra-Sincrotrone Trieste, Basovizza, Italy

Abstract

The European X-Ray Free-Electron Laser (XFEL) will generate extremely short and intense X-ray flashes from the electron beam of a 2.1 km long superconducting linear accelerator. The commissioning and operation of the accelerator relies heavily on high level software for the automatization of measurements and procedures. The paper gives an overview of the ongoing work and highlights some new measurement techniques.

INTRODUCTION

The European X-ray Free-Electron Laser (XFEL) is a research facility that has been constructed in collaboration between the European XFEL Facility GmbH¹ and DESY² in Hamburg, Germany [1–4]. The main component of the facility is a superconducting linear accelerator (linac) that delivers an electron beam with particle energies up to 17.5 GeV and average beam power up to ~600 kW into several long undulator sections. In these sections, the electrons generate extremely brilliant X-ray pulses at wavelengths down to 0.05 nm. These light pulses are distributed to several beamlines and end-stations for photon science experiments. Extensive infrastructure, including a cryogenic plant, has to be operated to allow the XFEL to work. An overview of the control system architecture for the entire facility is given in [5]. The 130 MeV injector has already been fully commissioned with beam, and the commissioning of the entire machine will start soon.

The XFEL is a system of considerable complexity, and operating it smoothly requires a high degree of automatization. We therefore aim to offer high level abstractions for all important machine parameters and to develop user friendly tools for typical physical and technical tasks in the control room. This paper gives a brief overview of our evolving control system landscape, reports on our experience from the commissioning of the injector, and highlights a few applications that have enabled us to perform unprecedented measurements of the beam emittance across the bunch train with a new measurement techniques.

* lars.froehlich@desy.de

¹ European X-Ray Free-Electron Laser Facility GmbH, Albert-Einstein-Ring 19, 22761 Hamburg, Germany

² Deutsches Elektronen-Synchrotron DESY, Notkestraße 85, 22607 Hamburg, Germany

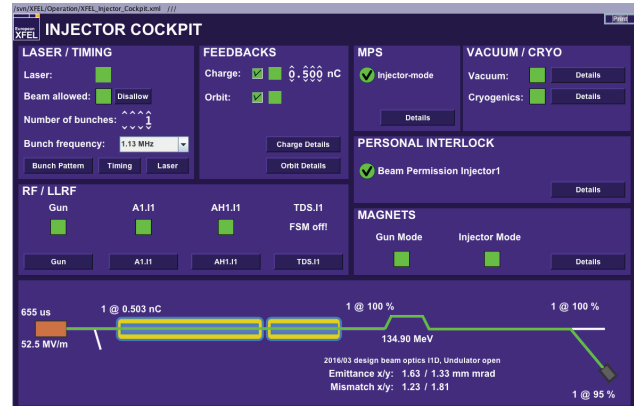


Figure 1: Screenshot of the “injector cockpit” *jddd* panel.

FUNDAMENTALS

No less than four main control system protocols are in use at the XFEL: DOOCS [6, 7], EPICS [8, 9], TINE [10, 11], and Karabo [12]. A lot of work has already been invested to improve the interoperability of the first three protocols [13], so that the problem of network communication across protocol boundaries is less daunting today than it was several years ago. Most of the remaining cross-protocol effort is focused on creating an interface with Karabo.

We are operating a central *configuration database* as a network service. It stores a complete list of beamline components and associated information such as calibration data. This helps to avoid inconsistencies in the configuration of distributed servers [14].

In large parts, XFEL controls follow a “rich server–thin client” philosophy. We are trying to implement high level abstractions and advanced data processing at the server level so that many user interfaces do not need to be programmed but can be configured with our *jddd* [15, 16] user interface builder (Fig. 1). Even applications requiring more complex interaction or more advanced plotting capabilities become simpler to write and easier to maintain with this approach.

Almost all of the control system servers for the machine are written in C++ and deployed on Linux systems. We have created a multitude of libraries to facilitate easy access to various accelerator components and to the central database, for numerical and image analysis tasks, for the calculation of optical functions, and for particle tracking. GUI applications are deployed on MacOS, Windows, and Linux desktops and written in Matlab, Java, or Python. Toolboxes and libraries for these languages are available as well.

VIRTUAL XFEL

Before installation in the real machine, many control system components of the XFEL can be tested in the *Virtual XFEL*, a deep simulation of the accelerator that runs on dedicated hardware with its own separate timing system [17]. Beam positions, charge readings, magnet currents, and other data are generated by fake front-end servers and passed to the original middle layer and data acquisition software for processing. A physics simulation performs particle tracking in real time based on the setting of the virtual magnets and feeds the beam positions back to the fake front-end servers, closing the loop. The result is a virtual machine that feels almost like a real one—beam cannot be transported properly until a reasonable magnet file has been loaded, changes in beam energy reveal spurious dispersion. We are continually improving the simulation in order to provide a test bed even for complex beam dynamics tasks such as beam-based undulator alignment.

INJECTOR COMMISSIONING EXPERIENCE

From December 2015 to July 2016, we commissioned the injector of the European XFEL with beam. Although it is only ~50 m long, it contains almost all of the subsystems of the entire machine—most notably (Fig. 2): an RF gun, a small spectrometer for the 5–6 MeV beam, one cryomodule with eight accelerating 1.3 GHz cavities (A1), one cryomodule with eight decelerating 3.9 GHz cavities (AH1), a laser heater chicane with a small undulator, an optics matching and diagnostic section with a transverse deflecting RF structure (TDS), four fast kickers and four scintillation screens, and a dump line to safely dispose of the 130 MeV electron beam.

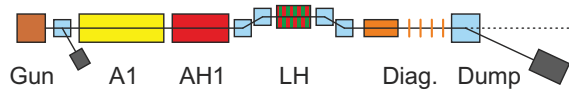


Figure 2: Schematic of the XFEL injector.

The commissioning relied heavily on high level software from the very start. For example, a middle layer server allowed controlling magnets by physical parameters such as deflection angles, fields, and particle energies instead of setting currents directly [14], and trajectory, transmission, and beam loss displays used pre-processed data from the data acquisition system (DAQ). The startup, conditioning, recovery, and shutdown of the RF stations was handled by finite state machines implemented as middle layer servers. Support for finding the optimal phase of the RF wave in the gun and in the cryomodules was also available through middle layer servers and standalone tools from the very beginning.

A number of generic multi-purpose utilities proved extremely useful for the commissioning of technical systems and for the automation of common tasks and measurements. These include tools for data recording, scans of arbitrary

control system channels, and correlation finding, as well as utilities for performing configurable checks on subsystems and for executing sequences of control system commands.

Beam-based feedbacks for bunch charge and trajectory could be tested soon after the commissioning of the standard diagnostics systems of the injector. They are based on the same architecture as the ones used at FLASH [18].

A good knowledge of the beam energy is essential for having control of the beam optics. From the very start, we could rely on the setting of the gun and the injector dump dipole for energy measurements. Later on, we commissioned a multi-layered system of energy information servers that provides the best possible energy information for the entire machine:

- A *LLRF energy gain server* gathers the expected energy gain for each RF module in the linac, taking acceleration phases and the possible disabling of modules into account.
- One or more *beam energy measurement servers* perform beam-based energy measurements in dispersive sections of the machine. This already worked well in the laser heater chicane and will be set up in many more places along the accelerator.
- An *energy profile server* scales the data from the LLRF energy gain server to the measured energies, providing a consistent energy profile for the entire machine in realtime.

Of course, there is also tool support for adjusting the magnetic lattice to this energy profile with few mouse clicks.

An *Elegant* [19]-based optics server [20] with a complete optics model of the machine provided a powerful engine for optics calculations as a network service from day one. It was later extended with matching capabilities and now serves as the backbone of most software related with beam optics. As an example, a simple *orbit response tool* proved most helpful for the cross-validation between the optics model and the actual magnetic lattice. By measuring the response of beam positions to the variation of corrector strengths, polarization errors and swapped planes can easily be diagnosed. In addition, the same tool can be used to determine the offset of quadrupoles by varying their strength, and measure dispersion by varying the beam energy.

EMITTANCE MEASUREMENTS AND OPTICS MATCHING

The beam exiting the RF gun is subject to strong space charge forces and eludes modelling in the usual framework of linear optics. Therefore, we follow the standard approach for linacs and match the beam to a design optics only after acceleration to ultrarelativistic energies in A1 and AH1, where linear optics can be used again. In order to perform this *matching*, the Twiss parameters (α , β , γ , and the emittance ϵ) of the beam need to be measured first. Because good control of the beam optics is essential for a smooth operation of the machine and because the emittance is an eminently important parameter for the FEL process itself, we have

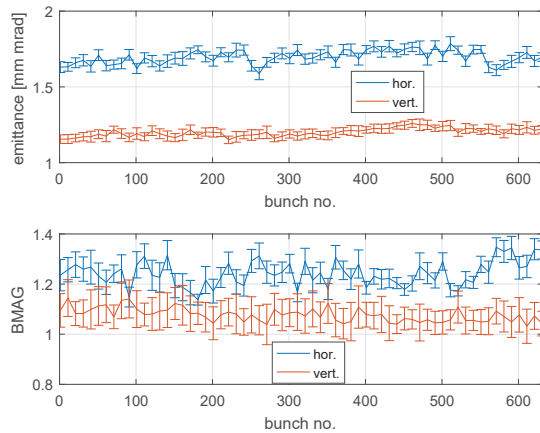


Figure 3: Normalized projected emittance and mismatch amplitude (BMAG) along the bunch train.

invested a lot of work into several independent measurement methods:

- The *four-screen on-axis* method acquires beam images from four scintillation screens with a fixed betatron phase advance between them. These four screens intercept the beam in its normal path, and have to be inserted and removed one by one to obtain images. The method is very robust and produces a measurement of the projected emittance in about three minutes.
- The *four-screen off-axis* method is similar, but faster: The screens are only inserted half-way into the beam pipe, so that they do not intercept the normal bunch train. Four fast kicker magnets are then fired to deflect individual bunches onto these off-axis screens. An emittance can thus be measured in few seconds and without interrupting the normal operation of the accelerator.
- The *quadrupole scan* method uses only a single screen and changes the phase advance by varying the strength of an upstream quadrupole. Because of its versatility, this method provides a good way of cross-checking the results of the other tools in multiple locations. We use an adapted version of the tool from the FERMI FEL. The method can also be extended into a multi-quadrupole scan that varies several magnets simultaneously to achieve better resolution; we have used this approach to good effect, but have yet to implement it in a non-expert tool.

The setup with kickers and off-axis screens allows us to measure emittances frequently. Automated measurements in intervals of few minutes could easily be implemented to monitor drifts of the beam optics over time. The kicker setup also made it possible for the first time to examine the evolution of the emittance over an entire bunch train; Figure 3 shows the horizontal and vertical emittance over a train of 650 bunches along with the mismatch amplitude (BMAG) with respect to the design optics. The method also integrates well with the use of the transverse deflecting cavity—by streaking individual bunches, slice emittances

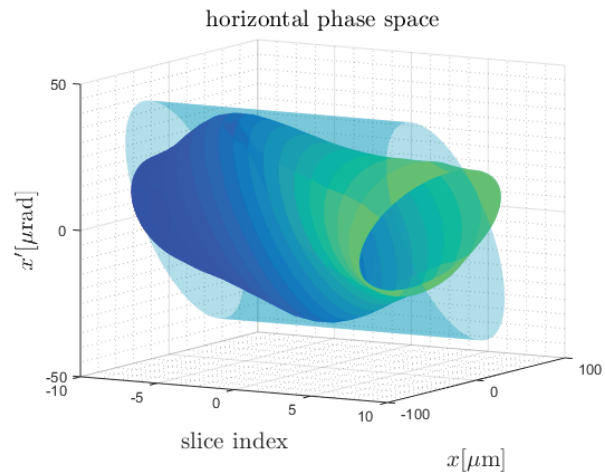


Figure 4: Horizontal phase space of the beam, measured with the four-screen off-axis method and transverse deflecting cavity.

can easily be measured (Fig. 4), and individual slices can be matched to a target optics.

CONCLUSION AND OUTLOOK

The European XFEL is a machine of considerable complexity, not only due to its sheer size and number of individual components, but also because of the intricacies of its pulsed mode of operation, of its beam distribution system, and of many subsystems. We are trying to reduce this complexity by offering more high-level, physical abstractions for all important machine parameters directly in the control system. This effort goes hand in hand with the development of user friendly tools for typical tasks in the control room.

Our experience with high level controls during the commissioning of the XFEL injector was generally a very positive one. The early availability of middle layer servers for optics calculations, magnet parameters, beam energies etc. made many commissioning tasks easier than expected. Based on this foundation, several measurement algorithms for emittances (both projected and slice) and Twiss functions could be developed and cross-checked successfully, which are now in the standard toolbox of every machine operator. This has given us an unprecedented amount of control over the injector optics—down to the matching of individual slices. We are now looking forward to the commissioning of the entire linac in the near future.

REFERENCES

- [1] A. Aghababayan et al., “The European X-Ray Free-Electron Laser technical design report”, DESY 2006-097, DESY, Hamburg, Germany, 2007.
- [2] W. Decking et al., “European XFEL post-TDR description”, XFEL.EU TN-2013-004-01, European XFEL GmbH, Hamburg, Germany, 2013.
- [3] W. Decking et al., “Status of the European XFEL”, in *Proc. FEL’15*, Daejeon, Republic of Korea, Aug. 2015, paper WEA04.

- [4] E. Cartlidge, “European XFEL to shine as brightest, fastest x-ray source”, *Science*, 354, 22–23, 2016.
- [5] A. Aghababayan et al., “The large scale European XFEL control system: Overview and status of the commissioning”, in *Proc. ICALEPCS’15*, Melbourne, Australia, Oct. 2015, paper MOA3O02.
- [6] A. Aghababayan et al., “The accelerator control system at DESY”, *ICFA Beam Dynamics Newsletter*, 47, pp. 139–167, 2008.
- [7] The DOOCS website, <http://doocs.desy.de/>.
- [8] L. R. Dalesio et al., “The experimental physics and industrial control system architecture: past, present, and future”, *Nucl. Instr. and Meth. A*, 352 179–184, 1994.
- [9] The EPICS website, <http://www.aps.anl.gov/epics/>.
- [10] P. K. Bartkiewicz and P. Duval, “TINE as an accelerator control system at DESY”, *Meas. Sci. Technol.*, 18, pp. 2379–2386, 2007.
- [11] The TINE website, <http://tine.desy.de/>.
- [12] B. C. Heisen et al., “Karabo: An integrated software framework combining control, data management, and scientific computing tasks”, in *Proc. ICALEPCS’13*, San Francisco, USA, Oct. 2013, pp. 1465–1468.
- [13] P. Duval et al., “Control system interoperability, an extreme case: Merging DOOCS and TINE”, in *Proc. PCaPAC’12*, Kolkata, India, Dec. 2012, pp. 115–117.
- [14] L. Fröhlich et al., “Magnet server and control system database infrastructure for the European XFEL”, in *Proc. ICALEPCS’15*, Melbourne, Australia, Oct. 2015, paper WEPGF006.
- [15] E. Sombrowski et al., “jddd: A tool for operators and experts to design control system panels”, in *Proc. ICALEPCS’13*, San Francisco, USA, Oct. 2013, pp. 544–546.
- [16] E. Sombrowski et al., “A HTML5 web interface for JAVA DOOCS Data Display”, in *Proc. ICALEPCS’15*, Melbourne, Australia, Oct. 2015, paper WEPGF150.
- [17] R. Kammering et al., “The virtual European XFEL accelerator”, in *Proc. ICALEPCS’15*, Melbourne, Australia, Oct. 2015, paper TUD3O04.
- [18] R. Kammering and C. Schmidt, “Feedbacks and automation at the Free Electron Laser in Hamburg (FLASH)”, in *Proc. ICALEPCS’13*, San Francisco, USA, Oct. 2013, pp. 1345–1347.
- [19] M. Borland, “elegant: A flexible SDDS-compliant code for accelerator simulation”, LS-287, Advanced Photon Source, Argonne, USA, 2000.
- [20] S. Meykopff, “An optics-suite and -server for the European XFEL”, in *Proc. PCaPAC’14*, Karlsruhe, Germany, Oct. 2014, pp. 52–54.

AUTOMATION OF THE MAGNETIC FIELD MEASUREMENTS OF THE AIR COILS BY MEANS OF THE MOVING WIRE SYSTEM

M. Yakopov[†], S. Abeghyan, S. Karabekyan, J. Pflueger, European XFEL Hamburg, Germany
Z. Zhao USTC/NSRL, Hefei, Anhui, People's Republic of China

Abstract

To ensure self-amplified spontaneous emission process the undulators, of a used for this, must not deflect the electron beam from its orbit. The possible deflection of the electron beam, introduced by undulator, must be corrected by means of two air coils. These air coils which are installed from both sides of the undulator, must eliminate not only the deflection angle, but also the displacement between electron beam trajectory and the orbit. For European XFEL 182 air coils are necessary. The magnetic field of each air coil was measured, to determine a conversion coefficient used by the control system. To minimize the measurement time an automated procedure has been developed and implemented. This paper describes the measurement setup, technical implementation method and automation procedure.

INTRODUCTION

At both ends of each undulator, used by European XFEL [1], gap dependent steering errors may occur due to unavoidable magnet imperfections. These errors lead to the errors in the 1st and the 2nd Field Integral. One of the important tasks during the commissioning of the undulators in the XFEL magnetic hutch is the exact measurement of the gap dependency of upstream and downstream kicks as a function of the gap. For the compensation of these errors an air coil corrector is foreseen on either end. The serial production of the air coils has been successfully done by an industrial supplier. There are 91 air coils with 12 mm gap and 91 air coils with 32 mm gap, due to geometry of the vacuum chamber, where they supposed to be installed. In order to precisely operate air coils, both software and hardware solution have been integrated into the undulator control system [2], which is based on the Beckhoff automation technology. The software requires the conversion coefficients between currents and steering strengths of the different coils. For precise determination of the conversion coefficients of all air coils an automated test stand has been built based on the moving wire (MW) technique [3]. The conversion coefficients obtained in this way have been implemented in the undulator control system. In addition the distribution of the first field integral across the air coil gap, a crosstalk of the vertical (B_y) component of produced magnetic field to the horizontal (B_z) component and vice versa have been measured. These measurements were a part of the commissioning procedure of the air coils.

AUTOMATED MEASUREMENT SETUP

Description of the Setup

The centerpiece of the measurement setup is the MW measurement system [4]. It is an automated system aimed to the measurements. A LabVIEW program is used to control the measurements, perform the data analysis and manage the data storage. The MW system is operated by a XPS – Newport controller which communicates with the control PC through the TCP/IP network protocol. The measurement setup also includes a water cooling system for the air coils. The functional diagram of the test setup is shown in Fig. 1. A hardware controller has been created for switching ON and OFF the power supplies of the air coils. The controller was operated by XPS via a GPIO interface. The current setting of the two power supplies has been adjusted manually to 1 A and controlled by Keithley multimeters with $\pm 5 \mu\text{A}$ accuracy for both B_y and B_z magnetic field components.

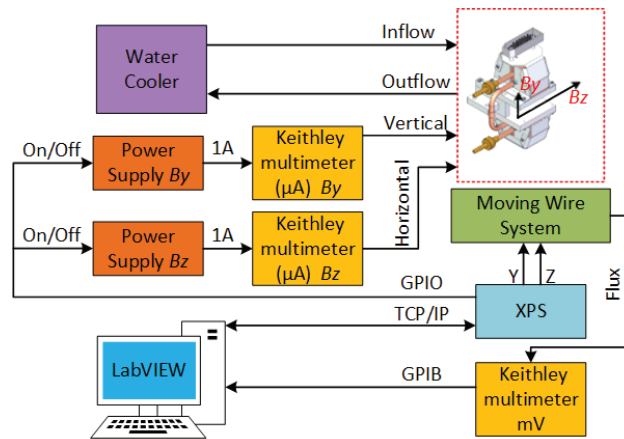


Figure 1: Functional diagram of the magnetic field measurement test setup of the air coils.

Functionality of the setup

The distribution of the first field integral of B_y and B_z components across the gap of the air coils have been measured using the developed Labview software. Before starting the measurements, the air coils have been installed on a specially designed holder, and the front edge of the wire has been positioned at the geometrical middle point of the air coil gap, in X and Y directions. Figure 2 shows an air coil installed on the MW bench ready for measurements.

[†] mikhail.yakopov@xfel.eu

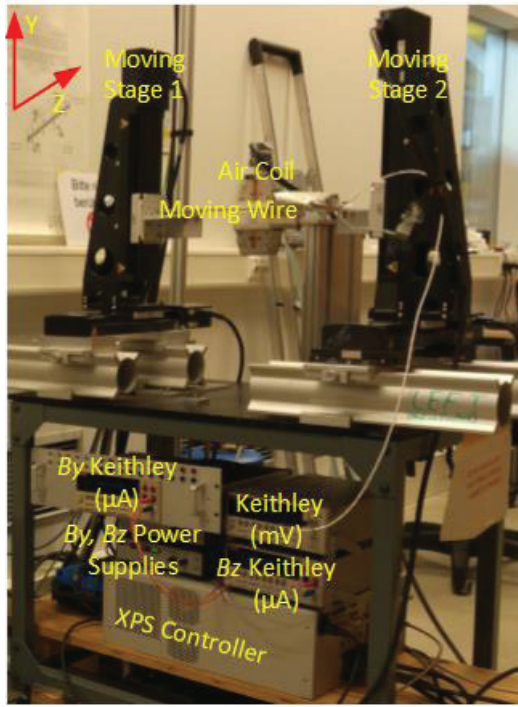


Figure 2: View of the moving wire setup used for the air coil measurements.

Once proper alignment of the moving wire with respect to the gap of the air coil is done, the automatic measurement software may be started.

MEASUREMENT PROCEDURES

There is a slightly different procedure implemented to measure the distributions of the first field integrals for air coils with 32 mm and a 12 mm gap. The difference is in the alignment procedure of the MW, in the vertical direction. In case of 32mm air coil this alignment is done automatically by means of so called *By* scan. To start the measurements, the proper air coil needs to be selected in the GUI of the control program. By clicking the start button all settings will become active and the measurement process will run automatically with the following steps:

1. Powering of the *By* coil;
2. Measuring the first field integrals with 1mm steps in the range of -10 mm to 10 mm along Y direction from the estimated middle point;
3. Un-powering the *By* coil, and eventually moving the wire into a new initial position.

The *By* field integral, as a function of Y direction, is shown in the left part of Fig. 3. The minimum indicates the true magnetic centre of the air coil. This position is taken as a new initial coordinate. The second and eventually next iteration of measurements is done for verification purpose. The final initial position of the magnetic centre is shown in the right part of Fig. 3.

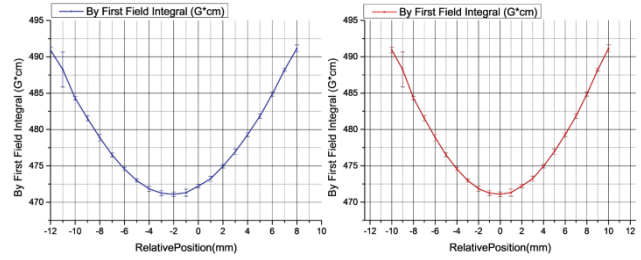


Figure 3: *By* scan of 32 mm air coil. Left -before and right - after the automatic alignment

Once this is done, the next steps will start automatically. This step actually uses the same principle that is implemented for the measurements of the *Bz* component. This step is done for both, 12 mm and 32 mm air coil measurements. In this step only *Bz* air coil is powered. The travel range is from -20 mm to 20 mm along the Z direction. The deviation between the maximum measured value and the initial position of the wire needs to be found. Afterwards the MW is aligned with respect to the centre of the air coil in the Y and Z direction. The following steps characterize the coil properties, especially the cross-talk between the coils. The third step includes the measurement of the first field integral of the *By* and *Bz* components in the median plane while both coils are powered. Travel range is from Z = -20 mm to 20 mm with 1 mm steps. The fourth step repeats the previous measurements without powering the *Bz* power supply, to measure only the value of *By* air coil without the contribution of the *Bz* air coil. The fifth step repeats the previous one by measuring only the *Bz* air coil without powering the *By* air coil. The sixth step repeats the same measurements with unpowered air coils. This measurement gives the *By* and *Bz* values of the ambient magnetic field, which need to be subtracted to get the real values produced by air coils. As it was already mentioned, all measurements have been done with the powering of the air coils with 1 A current. Once the measurements are done the final table and plots will be generated automatically. All obtained data, shown as a function of Z direction in the median plane. Table 1 gives an overview over the measured data.

Table 1: Overview over the Measured Data

Signals (G*cm)	Description
BY_BY1_BZ1	<i>By</i> with powered both <i>By</i> and <i>Bz</i> coils
BZ_BY1_BZ1	<i>Bz</i> with powered both <i>By</i> and <i>Bz</i> coils
BY_BY1_BZ0	<i>By</i> with powered <i>By</i> coil only
BZ_BY0_BZ1	<i>Bz</i> with powered <i>By</i> coil only
BY_BY0_BZ0	Y - vertical ambient magnetic field
BZ_BY0_BZ0	Z - horizontal ambient magnetic field

ANALYSIS OF DATA

To visualize and analyse the data another LabVIEW program was developed. The program generates the plots and visualizes them. The user interface of the program

allows to display any combination of the measured data. In the Fig. 4 all 8 measurement results are selected.

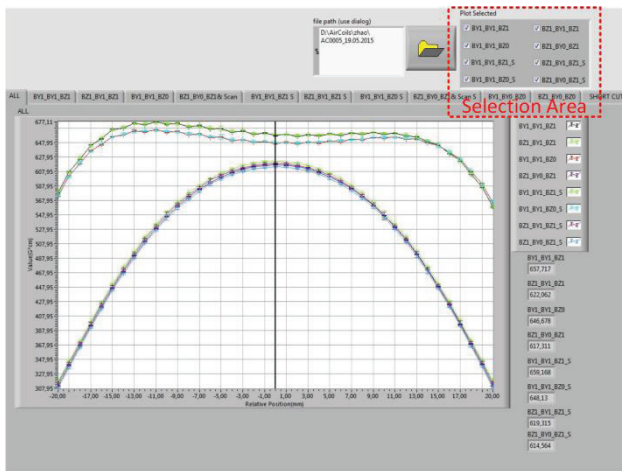


Figure 4: Graphical user interface of the program for the data analyses.

MEASUREMENT RESULTS

The main result for the undulator control system is the value of the first field integral of both B_y and B_z components, measured in the centre of the gap of the air coil. The ambient magnetic field must be subtracted. An example of the measured B_y component for 12 mm gap air coils is shown in Fig. 5.

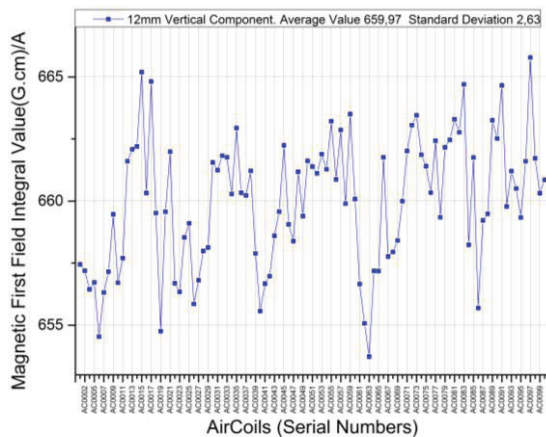


Figure 5: First field integral in Y direction at Y, Z=0 of entire amount of air coils with 12 mm gap. The ambient magnetic field is subtracted.

In the same way the B_z component of the air coils with 12 mm gap as well as both components of the air coils with 32 mm gap have been analysed.

IMPLEMENTATION OF THE CONVERSION COEFFICIENTS

The conversion coefficients, which are necessary to produce magnetic field value according to the set value, have been implemented in the undulator control system.

The air coils control algorithm uses the following equation:

$$\text{AppliedValue}[A] = |\text{SetValue}[Tmm]| * 32768 * B \quad (1),$$

where B is the conversion coefficient, which converts the applied current to the field integral for a specific type of the air coil. The following values have been used for calculating the corresponding transformation coefficients for the undulator control system (see Table 2):

Table 2: Average Values of Final Coefficients for 12 mm and 32 mm Air Coils

Coefficients (Tmm/A)	Standard Deviation
$B_{yAve12}=0,65997$	0,00263
$B_{zAve12}=0,61855$	0,00227
$B_{yAve32}=0,47724$	0,00135
$B_{zAve32}=0,40862$	0,00146

CONCLUSIONS

The MW setup allowed to make relatively fast commissioning of the complete set of the air coils with the accuracy better than 1 G cm (0.001 T mm) [4]. A standard deviation < 0.4% of all measurements has been achieved. An average B_{Ave} coefficients were considered sufficient to use during the commissioning of the undulator control system. These coefficients have been implemented into the undulator control system. Nevertheless, all measured data are available for each air coil, and could be used in case of any specific requirement.

REFERENCES

- [1] Y. Li, B. Faatz, J. Pflueger, "Undulator system tolerance analysis for the European x-ray free-electron laser", *Physical Review Special Topics – Accelerator and Beams* 11, 100701, 2008.
- [2] M.Yakopov, J. Pflüger, "Connection of air coils to the XFEL Control System", WP71_2013_11.
- [3] J. Pflüger, "Moving Wire Measurements", WP71/2013/05.
- [4] F. Wolff-Fabris, M. Viehweger, Y. Li, J. Pflüger "High accuracy measurements of magnetic field integrals for the european XFEL undulator systems" *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, Volume 833, 11 October 2016, p. 54.

DIAGNOSTICS AT JINR LHEP PHOTOGUN BENCH

M. A. Nozdrin*, N. I. Balalykin, V. F. Minashkin, G. D. Shirkov, JINR, Dubna, Russia
S. Weisse, DESY, Zeuthen, Germany

Abstract

The photoinjector electron beam quality strongly depends on the laser driver beam quality. For laser beam diagnostics, a “virtual cathode” system was realized at the photogun bench located in the Veksler and Baldin Laboratory of High Energy Physics (LHEP) of the Joint Institute for Nuclear Research (JINR). The system allows one to image laser beam profile at the cathode. For imaging, the AVINE software suite developed in DESY Zeuthen is used. Equipment for emittance measurement using the slit method was installed. The original emittance calculation software EmCa was created and tested with laser beam.

INTRODUCTION

Photogun bench of JINR LHEP [1, 2] aims to develop and improve “transmissive” photocathodes to increase quantum efficiency, lifetime and to decrease vacuum requirements. “Transmissive” photocathode is the development of the “Hollow” photocathode conception developed at JINR [3, 4]. It consists of micron-sized metal mesh or quartz/sapphire plate with thin-film coating: either metal or semiconductor.

Beam diagnostics, both electron and laser, is important in order to investigate cathode characteristics. To accomplish this, a set of diagnostics subsystems is installed at the bench.

BENCH EQUIPMENT

Main bench elements (Fig. 1) are DC photogun with the maximum voltage of 30 kV, focusing magnet with correction windings, diagnostics and driver laser. Bench beamline vacuum is less than 10^{-8} torr. A faraday cup is used for electron bunch charge measurement. Laser pulse energy is measured by an Ophir Nova II power/energy meter equipped with PE25 pyroelectric sensor.

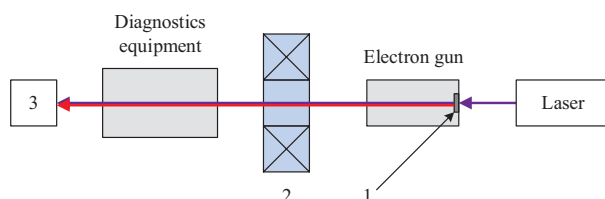


Figure 1: Photogun bench scheme: 1 - “transmissive” photocathode; 2 - focusing magnet with correction windings; 3 - beam dump.

Lasers

The main laser driver currently is LS-2134 by LOTIS TII (Minsk, Belarus). Also LS-2151 model was used,

LS-2132UTF model is being commissioned now. As next step, the new—photoinjector—bench is being constructed. It will use the unique laser system [5, 6] by IAP RAS (Nizhny Novgorod, Russia) and will have a maximum electron energy of 400 keV. Key parameters of the aforementioned laser systems are listed in Table 1.

Emittance Measurement Equipment

Emittance is one of the main particle beam parameters, its measurement is one of the key photocathode investigation tasks. Therefore, a slit emittance measurement system is being developed at the bench. It consists of the following equipment:

- slit mask, 1 mm thick tungsten plate with 9 50 μ m slits located at 3 mm distance from each other;
- vacuum chamber with 2-position pneumatic actuator where the slit mask is installed;
- scintillator screen;
- high sensitivity CCD camera Prosilica GC1380 for beam imaging;
- compressor with 7 atm maximum pressure for pneumatic actuator operation.

Imaging Equipment

A high sensitivity Gigabit Ethernet CCD camera AVT Prosilica GC1380 [7], together with a lens Kowa LM50JCM, is used at the bench. Camera parameters are given in Table 2. Two external trigger inputs are provided: isolated (trigger latency 5 μ s, jitter ± 0.5 μ s) and non-isolated (trigger latency 3.7 μ s, jitter ± 20 ns). As potential differences are not expected on the local bench setup and due to absence of powerful noise sources, the non-isolated channel is used.

Table 1: Lasers Parameters

Parameter [units]	LS-2134	LS-2151	LS-2132	IAP RAS
Wavelength [nm]	266	266	266	260–266
Micropulse length [ps]	15,000	75	5,000	8–12
Micropulse energy [μ J]	15,000	5,000	30,000	1,85
Macropulse length [μ s]	—	—	—	800
Micropulses in macropulse	—	—	—	8 000
Macropulse repetition rate [Hz]	—	—	—	10

* nozdrin@jinr.ru

Table 2: Prosilica CCD Camera Parameters

Sensor	2.3" Sony ICX285AL CCD
Resolution	1360 (H) × 1024 (V)
Pixel Size	6.45 μm × 6.45 μm
Frame rate	20,2 fps
Power Requirements	Less than 3,3 W (5-25 VDC)
Hardware Interface	IEEE 802.3 1000baseT
Software Interface	GigE Vision Standard 1.0
Digitization (ADC)	12 bit

SOFTWARE

Vimba SDK Based Software

AVT Vimba SDK [8] includes Vimba Viewer software which can be used for basic imaging. However, for advanced diagnostics purposes, the software lacks functionality. Hence, development of a custom application using the above mentioned SDK was started. Microsoft Visual Studio 2010 and C++ programming language were used for development. The following functions are provided:

- identify and connect to camera using its unique number;
- asynchronous video acquisition and displaying;
- real time displaying of the intensity distribution in operator specified cross-sections.

At this stage of development an offer from DESY Zeuthen colleagues to use DESY-developed Advanced Video and Imaging Network Environment (AVINE) software [9] was received. As AVINE has necessary features already implemented and provides a well-tested interface to AVT Prosilica cameras, especially of GC type, it is a good match for our use-case. In this regard, the development of our own software was postponed.

ELECTRON BEAM DIAGNOSTICS

Setup

Current electron beam diagnostics setup is shown at Fig. 2 (1 - LS-2134 laser, 2 - electron gun, 3 - focusing magnet, 4 - emittance measurement station, 5 - camera). The yellow arrow indicates the scintillating screen (phosphor and potassium glass were used) position.

Laser Light Issue

The main problem for electron beam imaging is the simultaneous laser beam at the scintillating screen, which also becomes a part of the video image. This issue is redoubled due to low electron energy. The separation of electron and laser beam is not possible in the current setup. The following mitigation scenarios are being considered:

- usage of different screen materials (i.e. YAG and P-22 phosphor);
- electron beam deflection with a dipole magnet;
- foil, transparent for electron beam and opaque for laser.

Transverse Emittance

Currently, the setup for vertical plane transverse emittance calculation is developed and tested with the laser beam. De-

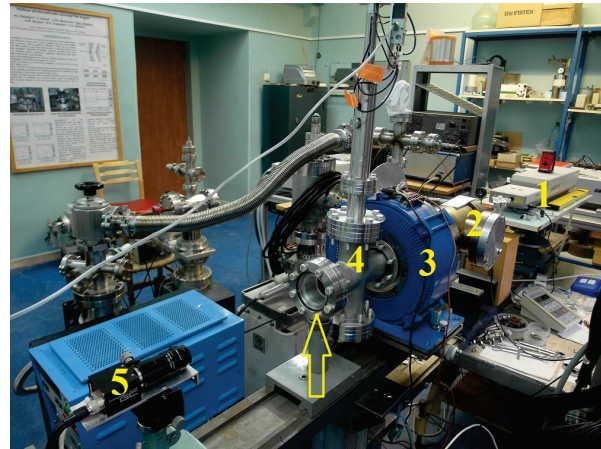


Figure 2: Electron beam diagnostics setup.

pending on the electron beam test results, either the pneumatic actuator will be changed to a 3-position one (with two masks with mutually perpendicular slits) or the pepper-pot method will be used.

EmCa software (Fig. 3) was developed for emittance calculation out of images (sequences) previously saved with AVINE Video Client 3. Microsoft Visual Studio 2010 and C++ programming language were used. A formula derived from [10] is used for calculation.

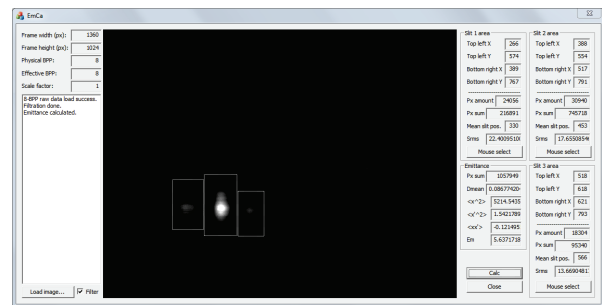


Figure 3: EmCa software.

Algorithm is the following:

1. Open image file. After file has been loaded, image and its parameters (size, physical and effective bits per pixel amount, scale factor value) are displayed. Currently, *.imm format of AVINE (no compression) with 8 bits per pixel (BPP) resolution is supported.
2. Cut-off filtering. If this option is enabled, all pixels with values 0 to 15 will be set to black (value 0).
3. Beamlet selection. Beamlets are manually selected by mouse: press button and select box. Program was tested using a laser beam image showing 3 beamlets, hence beamlet amount is fixed to 3 so far.
4. Emittance calculation. After Calc button has been pressed, Emittance is being computed.

The following program development is planned for future:

- support for arbitrary amount of beamlets;
- beam image in the phase space;
- 16 BPP resolution and AVINE *.imc format (zlib compression) support;
- automatic beamlet selection;

- two-plane transverse emittance calculation (after hardware realization).

LASER BEAM DIAGNOSTICS

Virtual cathode setup was realized at the bench. The equipment setup and a sample beam profile image are shown at Fig. 4.

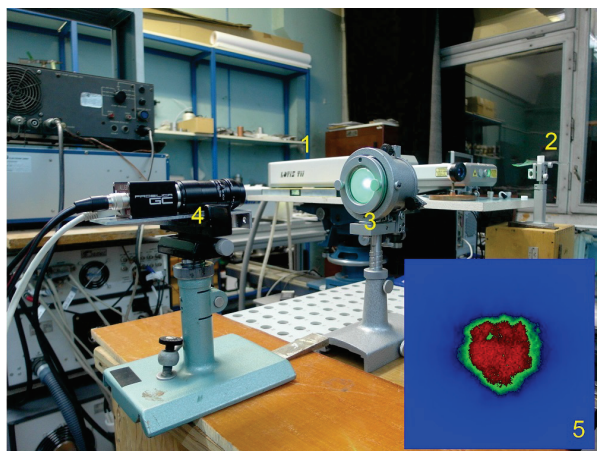


Figure 4: Virtual cathode equipment: 1 - laser, 2 - mirror, 3 - phosphor screen, 4 - camera, 5 - laser beam profile at the virtual cathode.

Stability measurements of the laser beam position and rms size were done. Result for beam position is given at Fig. 5. As one can see, the beam trends to horizontal shift. Further investigations are planned to understand this effect. Similar to beam position, rms size has been measured for 30 minutes. A drift could not be observed. Beam rms size was stable at 1.64 mm in x ($\sigma = 0.011$) and 1.55 mm in y ($\sigma = 0.013$).

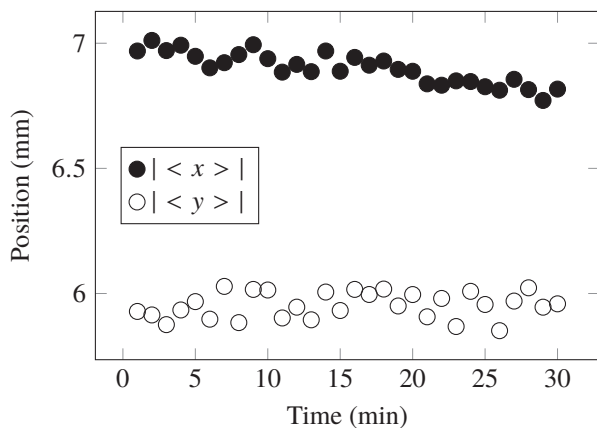


Figure 5: Stability measurement for beam position.

CONCLUSION

Virtual cathode setup was realized at JINR LHEP photogun bench. Imaging is realized using the AVINE construction toolkit developed at DESY Zeuthen. Equipment for slit emittance measurement was installed; software for emittance calculation was developed and tested with laser.

First priority issue is the separation of electron and laser beam at the screen using either a window being transparent for electron beam while being opaque for laser beam or a magnet-based electron beam deflection. Afterwards, realization of the 2-plane emittance measurement with the slit/pepper-pot method is foreseen.

REFERENCES

- [1] N. I. Balalykin, V. F. Minashkin, M. A. Nozdrin, G. D. Shirkov, and V. G. Shabrato. "Researching the Characteristics of Photo- and Thermoemission Cathodes." In: *Physics of Particles and Nuclei Letters* 5.7 (2008), pp. 605–608.
- [2] N. I. Balalykin et al. "JINR LHEP photoinjector prototype." In: *Physics of Particles and Nuclei Letters* (2016). To be published.
- [3] M. A. Nozdrin et al. "Hollow photocathode conception for e-gun." In: *Proceedings of the XXII Russian Particle Accelerator Conference. RUPAC2010* (Protvino, Russia, Sept. 27–Oct. 1, 2010). 2010, pp. 59–61.
- [4] M. A. Nozdrin et al. "Hollow photocathode prototype for e-gun." In: *Proceedings of the 10th European Workshop on Beam Diagnostics and Instrumentation for Particle Accelerators. DIPAC2011* (Hamburg, Germany, May 16–18, 2011). 2011, pp. 242–244.
- [5] N. Balalykin et al. "JINR Powerful Laser Driver Applied for FEL Photoinjector." In: *Proceedings of the 5th International Particle Accelerator Conference. IPAC2014* (Dresden, Germany, June 15–20, 2014). 2014, pp. 2906–2908.
- [6] Ekaterina I. Gacheva et al. "Laser Driver for a Photoinjector of an Electron Linear Accelerator." In: *IEEE Journal of Quantum Electronics* 50.7 (July 2014), pp. 522–529.
- [7] GC1380 / GC1380C. *Technical Manual*. 700018A. Allied Vision Technologies Canada Inc. Sunnyvale, California, Aug. 26, 2010.
- [8] Vimba 1.4 — The SDK for Allied Vision cameras. Allied Vision Technologies GmbH. <https://www.alliedvision.com/en/products/software.html>
- [9] Stefan Weisse, David Melkumyan, and Philip Duval. "Status, Recent Developments and Perspective of TINE-powered Video System, Release 3." In: *Proceedings of the 13th International Conference on Accelerator and Large Experimental Physics Control Systems. ICALEPCS2011* (Grenoble, France, Oct. 10–14, 2011). 2011, pp. 405–408.
- [10] Min Zhang. *Emittance Formula for Slits and Pepper-pot Measurement*. FERMILAB-TM-1988. Oct. 1996.

ORBITKORREKTUR, A JAVA CLIENT FOR TRANSVERSE ORBIT CORRECTION IN PETRA-III

G. K. Sahoo[#], F. Brinker, F. Wedtstein, DESY, Hamburg, Germany

Abstract

PETRA-III is a 3rd generation synchrotron light source dedicated to users at 14 beam lines with 30 instruments in the Max von Laue Hall and 10 beamlines in newly constructed Extensions in North and East which are under installations. PETRA-III is operated with several filling modes such as 60, 480 and 960 bunches with 100 mA or 40 bunches with 90 mA at an electron beam energy of 6 GeV. Transverse orbits are corrected to a reference orbit, which is based on BBA measurements taking into account requirements of the beamlines and results of the dispersion correction. Histograms from 244 BPMs are displayed by means of this Java Client OrbitKorrektur. The orbits may be corrected using the Effective Correction Method with a few correctors or can be corrected using the SVD Method with proper singular Eigen values using theoretical response matrices from an Optic Server of used Optics. In this application additional features are implemented for better observation and analysis of orbits. Furthermore it can also be used to do many additional jobs, such as showing corrector set currents, loading golden and reference orbits, local bumps, displaying the first turn data during commissioning etc.

INTRODUCTION

PETRA-III [1] is a 3rd generation synchrotron light source commissioned with electron beam energy of 6 GeV and 100 mA stored current at betatron tune values of 37.12 and 30.28 in 2009. It has a large circumference of 2304 m which is considerably larger than any existing light source around the world. The machine consists of arcs and several straight sections. The so-called long straights which have a length of 108 m are located in the North, East, South and West. In between two long straights are arcs and a short straight section with a length of 64.8 m. The magnetic structure is a simple FODO lattice. The part that extends from the middle of one long straight to the middle of the adjacent short straight is the basic building block of the machine. Since this section is just one eighths of the machine it is called an octant. The magnetic arrangement of one octant is mirror reflected at the middle of the short straight. Electrons are injected in the South-East (SE) and travel clockwise around the machine. The octant extending from North-East to East was modified breaking the fourfold symmetry and is called as new octant (built in the building *Max von Laue Hall*). It consists of nine Double Bend Achromat cells (DBA). Eight of them provide space for one 5 m or two 2 m long insertion devices (ID).

The two 2 m IDs are inclined towards each other by 5 mrad. This scheme allows operating two independently tunable undulators in a single straight section with beam paths sufficiently separated for individual beam line optics. The ninth straight section is suitable for the installation of an insertion device up to a length of 20 m. The horizontal beam emittance is 1 nm.rad while a coupling of 1% amounts to a vertical emittance of 10 pm.rad. The machine is dedicated to users for experiments from 14 beam lines with 30 end-stations in the Max von Laue Hall. The storage ring has been further modified in 2014 in the North and East quadrants to incorporate 10 new beam lines including a Super Luminescent beam line from dipole radiation. PETRA operates with several filling modes, such as 480 and 960 bunches with a beam current of 100 mA or in the Timing mode with 40 or 60 bunches with 90 – 100 mA. The new light source needs careful magnet alignments, orbit stabilization and closed orbit and dispersion corrections. A prerequisite for the small vertical emittance are tight tolerances on the spurious vertical dispersion that is mainly created by the misalignment of the sextupoles. The closed orbit deviations result from field errors arising from magnetic element positioning errors. The most severe effects come from misalignment of quadrupole magnets, where the resulting dipole field is proportional to both gradient and alignment errors. The closed orbit distortions have been simulated with the alignment tolerances and field errors. Since the integrated quadrupoles strength in the new octant is two to three times higher than in the old octants the alignment requirements are accordingly tighter.

For the initial design condition and without magnet errors the particles move along the so-called “Design Closed Orbit” with designed energy. Generally this orbit is centred in the magnets and Beam Position Monitors (BPMs). The offsets of the BPMs adjacent to quadrupoles are measured with a Beam Based alignment (BBA) to allow the beam passing through the centre of the quadrupoles. Variations of the initial conditions, alignments and magnet field errors lead to a deviation from the “Design Closed Orbit” and oscillate around a distorted closed orbit. Off energy particles take shorter or longer paths around this closed orbit as per the deviation from the designed energy. Transverse orbits are corrected to a reference orbit, which is based on BBA measurements taking into account requirements of the user beamlines and results of the dispersion correction.

[#] Gajendra.Kumar.Sahoo@desy.de



Figure 1: The OrbitKorrektur Java Client as a GUI.

ORBITKORREKTUR JAVA CLIENT

The Orbitkorrektur program is a Java Client (Fig. 1) used for slow orbit correction of PETRA-III using the Singular Value Decomposition (SVDcor) [2] or Most Effective Orbit Correction (Effcor) Methods [3]. The details of these methods or mathematical procedures are not described here. They are not part of the client program but of the Optics server. The program is a complex structure of many supportive Java applications, and communicate with many different types of servers using the TINE [4] network, Libera [5] server for BPM orbit readings, REFORBIT server for golden orbit, Central Magnet Server (MCS) for magnet currents etc. The main application is a Device Manager where all initialization is done with P3-OrbitKorrektur.xml file and are linked with AcopTransport [6] applications. Two Histograms are used to display of any combination of horizontal/vertical orbits and corrector currents. One of the monitor can be programmed to show Turn by Turn data. The prime function of the program is to correct the orbits to golden orbit, which is the resultant of BBA based orbit and BPM offsets or to any reference orbits as saved earlier. This is performed by using 244 BPMs and 210 horizontal and 189 vertical correctors. The details of Twiss Functions and the theoretical response matrices are available in the optic server as used at the time of correction calculations and applications. During the orbit correction one can opt for a global or a local orbit correction by selecting a range for the BPMs. Similarly, one can choose the type of correction scheme whether it is Effcor or SVDcor. After the calculation of the corrector strengths, the Corrector Table is populated with corrector names and the calculated strengths in terms of magnet set currents in Amperes. During normal operation for users the BPM Table is shown with state of BPMs in Fast Orbit Feed Back (FOFB) system. This client has many features and uses the subsystems as explained below.

The Device Panel

This is the main class and is used for updating the data variables through Update (Observable o, Object obj). The data are connected with TINE asynchronous links via

initialization files in Device Manager. In the first initialization ACOP-Observables are defined and send to proper Histograms for display which are then further controlled by the operator by mouse adapter. The orbits are displayed in micro/millimeters. The histograms can also show corrector strengths in units of Amperes. Provision is made to show the status and betatron functions of the BPMs and correctors. One can decide to enable or disable single BPMs or correctors for the orbit correction.

Reference Files

The reference files are managed by the FILEIO class for loading and saving the machine status as a file by the help of the File Sequencer [7]. The File Sequencer maintains the reference and golden orbits files. These orbits are displayed in the Histograms over orbit display panel. This procedure helps in importing the golden orbits at undulator sections from MATLAB.

Device Data for BPMs and Correctors

The BPMs get recent data from the Libera server in nanometers [8]. The beta functions, dispersion functions and phase advances are available for all elements from the optics server. This makes available of response matrices and preset corrector strengths from optic server for selected orbit to be corrected at BPM positions and wait for Orbit Correction to be executed.

Orbit-Korrektur

The Orbit Correction (Orbit-Korrektur) is mainly associated with two functions. At first the new corrector strengths are calculated for the selected orbit. In a second step a chosen fraction of the calculated correction is applied to the correctors. The correction is calculated by the optics server using the theoretical response matrices for the actual optics. Depending upon the type of correction SVDcor/Effcor, the range of BPMs to be used, and the optics, the server returns the calculated strengths of the magnets to be applied for orbit correction to golden orbit/reference orbit etc. The most recent orbit correction can be undone if required. The process of setting the magnet currents to the magnets using the Central Magnet Server are carried out by PS.Bumpfahren /PS.Zielfahren [9] protocols which insure that the changes are applied simultaneously.

Optic Server

The optic server is used to store all types of magnetic optics used in operation of PETRA-III. The loaded optics is assigned by the PETRA Global optics parameter. This server plays an important role to solve the orbit correction procedure or bump protocols with supplied BPM orbits and return correctors strengths. In the server different service routines are sorted by equipment functions and properties. These are described in the following subsections. Mostly these are invoked by Remote Procedure Calls (RPC).

The data of this server begin always with a header structure which contains general data like a machine identifier, actual optics, and an error string. The result of the call is zero for successful execution or errors code otherwise.

Equipment Function OPTICS This function gives information of the optics with many defined properties to get Twiss functions, elements names etc.

Equipment Function ORBCOR This function is utilized for orbit correction with many defined properties to get orbits, kicks, currents, corrector names etc.

Equipment Function BCOEFF The bump service calculates the coefficients for closed bumps of 3 or 4 correctors and allows storing and reading arbitrary/special bumps. The coefficients corresponding to a bump are either fixed or recalculated for the actual optics.

Equipment Function K2I2K Under this property all routines are collected to transfer magnet strength to power supply currents and vice versa. The data are transferred by an array of circuit data structures used for bumps. Together with the information of machine and year it is possible to convert the values for a given energy.

Bumps

This program can create 3 – 4 corrector bumps at selected azimuthal position. But in practice this function is less used as another better capable program to play with bumps in PETRA control room is available.

First Turn, Single or Multiple Injection

These classes are in verge of testing for fully functionality.

Display and Alarm Reports

As an observer, this program keeps track of FOFB operation during Top Up mode. It displays the readiness of this client for slow orbit correction or status of FOFB and Undulator Gaps. It also shows the variation of injection magnets pulse amplitudes as well as the synchronization of BPMs with system/machine clocks. These alarms help in proper utilization of this client.

Libraries Used in OrbitKorrektur

The following libraries are used in the main OrbitKorrektur program:

MstApp11.jar, DeviceManager.jar, Optics.jar, orbitUtil.jar, acopbeans.jar, caj-1.1.5.jar, Tine.jar, CamInstrumentation.jar, Fileutilities.jar, Jca-2.3.2.jar,

Jdoocs.jar, Jmf.jar, Logutility.jar, Logviewer.jar, OrbitUtility.jar, Printlikemstapp.jar, Seqserverlib.jar, Sequencer.jar, simpleTINEWrapper.jar, svnkit.jar, TangORB-6.0.7.jar, Timing.jar.

CONCLUSION

The orbit correction program is successfully used for slow orbit correction of PETRA-III since 2009 and used in upgrade of PETRA-III magnetic lattice with new extensions in North and East. This is capable of loading different optics with low and high beta functions for user's choice at undulator sections and with different beam energies such as 5 or 6 GeV by using 244 BPMs and 210 Horizontal /189 Vertical corrector magnets. The orbit correction can be carried out locally with a few BPMs or globally with all available BPMs utilizing the Effective Correction with a few correctors or SVD with proper selection of Eigen values in the plane of correction. Only, the relevant features are discussed here where the program has many more capabilities to perform.

ACKNOWLEDEMENTS

The authors gratefully acknowledge and would like to thank Rainer Wanzenberg and Reinhard Bacher for their constant encouragements and guidance.

REFERENCES

- [1] K. Balewski, W. Brefeld *et al.*, "PETRA III: A new high brilliance synchrotron radiation source at DESY", in *Proc. EPAC'04*, Lucerne, Switzerland, Jul. 2004, pp. 2302-2304.
- [2] W. Press, B. Flannery *et al.*, "Numerical Recipes", 1st Ed., Cambridge University Press, Cambridge, 1987.
- [3] B. Autin and Y. Marti, "Closed orbit correction of A.G. machines using a limited number of magnets", CERN ISR MA/73-17, 1973.
- [4] P. K. Bartkiewicz, P. Duval, "TINE as an accelerator control system at DESY", *Measurement Science and Technology*, 18 (2007) pp 2379-2386.
- [5] Instrumentation Technologies, <http://www.i-tech.si>
- [6] J. Bobnar, I. Kriznar *et al.*, "The ACOP family of beans: A framework independent approach", in *Proc. ICALEPCS'07*, Knoxville, Tennessee, USA, Oct. 2007, pp. 138-140.
- [7] Jürgen Maass and Pedro Castro-Garcia, DESY, private communications.
- [8] G. Kube, K. Wittenburg *et al.*, "Overview of the diagnostics systems of PETRA III", in *Proc. EPAC'08*, Genoa, Italy, Jun. 2008, pp. 1323-1325.
- [9] Marcus Walla, DESY, private communications.

FAST ORBIT FEEDBACK AT DELTA*

P. Hartmann, A. Althaus, D. Rohde, D. Schirmer, G. Schuenemann[†], P. Towalski[†], T. Weis, S. Khan
DELTA, TU Dortmund University, Germany

Abstract

At the electron storage ring DELTA, studies of a fast orbit feedback integrating Libera Electron and Bergoz MX-BPM electronics were conducted. A review of the project and its results is given.

INTRODUCTION

Global orbit feedback systems that correct orbit distortion in a frequency range between 1 and 500 Hz have become a standard at modern synchrotron light sources [1]. They are typically based on FPGA-driven beam position monitoring (BPM) electronics as for example the I-Tech Libera Brilliance [2].

At the 1.5 GeV synchrotron light source DELTA [3] at the TU Dortmund University, the storage ring as well as the booster synchrotron are mainly equipped with Bergoz MX-Beam Position Monitors [4]. At strategic positions in the storage ring, ten MX-BPMs have been replaced by I-Tech Libera Electron/Brilliance BPM electronics which are capable of measuring the beam position turn-by-turn. The Libera BPM electronics are shipped with a digital interface that integrates seamlessly into DELTA's control system EPICS [5] while the MX-BPMs provide the measured beam position as an analog voltage that is at present digitized by 12-bit analog-to-digital converters (ADC) and fed over CAN bus into the EPICS control system of the accelerator. A slow control system based global orbit feedback (latency > 1 s) compensates for slow beam motion caused by insertion device drifts and thermal drifts.

Fast beam motion

Beam motion at frequencies above 1 Hz has been identified by turn-by-turn measurements from Libera BPM Electronics (see Fig. 1). While high-frequency distortion is mainly caused by the line frequency and its harmonics, distortions at 5.3 Hz, 10.6 Hz and 12.5 Hz were identified as girder resonances using acceleration sensors on top of magnets and on the floor (see Fig. 2) [6].

LOCAL ORBIT FEEDBACK

Suppression of fast beam motion has been investigated in a first step with a fast local orbit feedback based on two I-Tech Libera BPMs and four correctors (see Figs. 3 to 5) [6]. The Libera devices use a code developed at the Diamond Light Source called the Diamond Communication Controller (DCC) [7]. This code runs on a Virtex-II-Pro FPGA contained in the Libera BPM electronics. It distributes measured

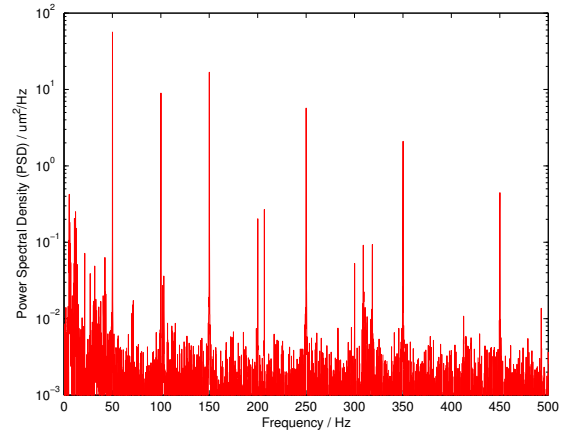


Figure 1: Vertical beam motion at BPM42.

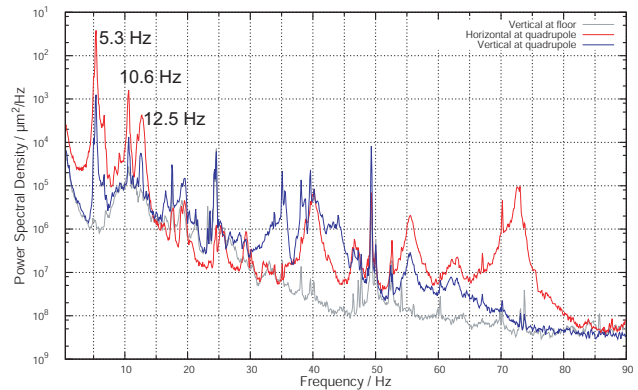


Figure 2: Magnet motion due to girder resonances.

BPM position data at a rate of 10 kHz between all participants linked to a dedicated optical fiber network.

In order to compute orbit corrections and to apply them to fast power supplies and corrector magnets, we implemented the DCC on a Virtex-II-Pro FPGA mounted on the Digilent XUP-Virtex II Pro evaluation board (XUPV2P) [8]. This board was connected to the fiber network and was thus able to receive BPM data at a 10 kHz data rate. In addition, we developed software for orbit correction based on a PI regulator and a driver for fast magnet power supplies (see Fig. 4).

With an optimum set of parameters we were able to reach an upper frequency limit for orbit correction slightly above 350 Hz (see Fig. 5). Distortion at higher frequencies is amplified instead of being damped. This frequency limit is attributed to latency of the beam measurement and regulation process. Beam motion at 50 Hz was damped to -20 dB of the original amplitude, while at 150 Hz the damping still was -10 dB.

* Work supported by BMBF, FKZ 05P09PERB5 and Forschungszentrum Jülich, contract no. COSY/FAIR-114

[†] Affiliation changed

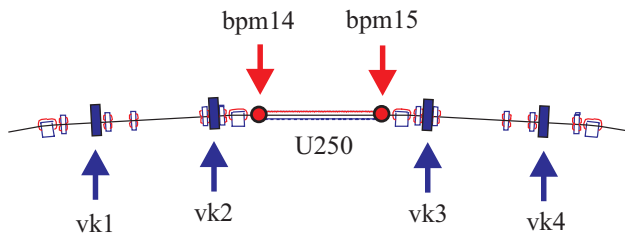


Figure 3: Local orbit feedback for the FEL undulator (U250).

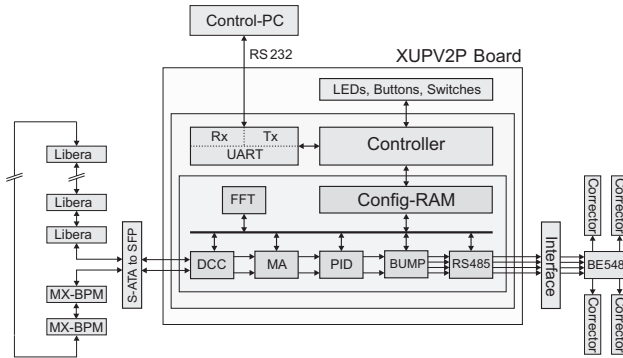


Figure 4: Schematic of the software running on the XUPV2P for the local orbit feedback.

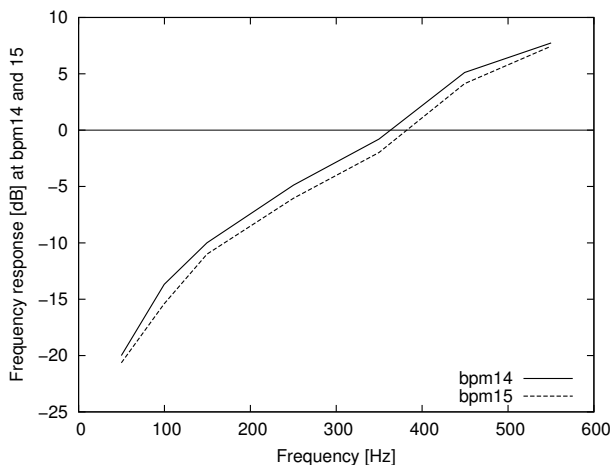


Figure 5: With an optimized set of parameters a frequency limit of about 350 Hz was reached.

BPM EXTENDER

At the same time, we developed the 'BPM Extender 3000', a digital frontend for up to four Bergoz BPMs (see Fig. 6) [9]. The Extender is also based on the XUPV2P evaluation board. An ADC board and other peripheral electronics were added (see Fig. 7). The four Bergoz BPMs are clocked at the highest specified rate of 40 kHz, resulting in position information at a rate of 10 kHz.

This fast data stream is passed to the DCC that distributes it over the DCC network. The link of the XUPV2P to the fiber network was established by connecting the three SATA ports and one extension port of the XUPV2P to SFP jack connectors. The SFP connectors are equipped with fiber

connectors. An Extender 3000 is thus capable of having at maximum 4 neighbours in the DCC optical fiber network.

The original DCC had to be modified to match the requirements of the XUPV2P. For example, the XUP-DCC serves four BPMs while the original DCC serves only one. In addition, the on-board SATA clock of the XUPV2P running at 75 MHz had to be replaced by a 106.25 MHz clock in order to integrate the XUPV2P into the Libera DCC network.

Besides the distribution of this BPM data over the DCC, the data is downconverted to a data rate of about 10 Hz by averaging over 1024 samples. We implemented Linux on one of the internal PowerPC-405 processors of the Virtex II Pro FPGA. A kernel driver was developed that makes the 10 Hz data stream from the FPGA core available through a linux device file. The control system software EPICS including ASYN- and StreamDevice drivers [10], [11] was then compiled and installed. ASYN device support reads data from the linux device file and provides it as EPICS records in the control system network. We were able to achieve a statistical error of less than 0.5 μm for the horizontal and vertical position data of all four Bergoz MX-BPMs [9].

XUPV2P Board

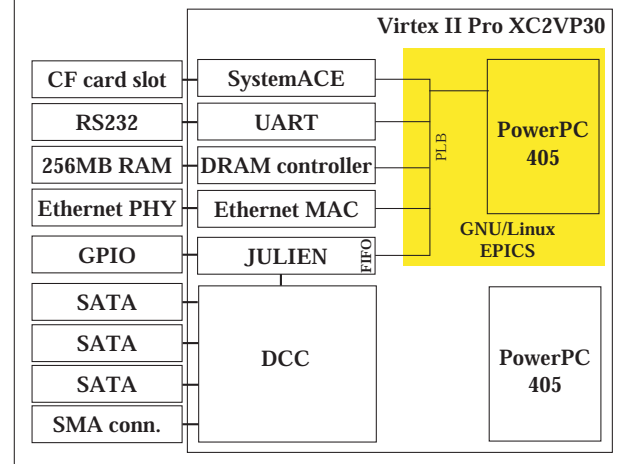


Figure 6: Schematic of the soft- and hardware on the XUPV2P board in the Extender 3000.

GLOBAL ORBIT FEEDBACK

In order to set up a fast global orbit feedback, all MX-BPMs were connected to BPM Extenders using their debug connectors. This way they were still able to provide data in parallel on the 'classical' data path for standard beam operation. The initial network topology was chosen as a ring with one diagonal connection. The time window for the DCC was chosen to be 60 microseconds. Within this window, all BPM data has to be distributed to all participants of the fiber network.

A few 20 years old MX-BPMs were not able to be clocked at 40 kHz. We were able to return them to the manufacturer for hardware upgrade and recalibration.

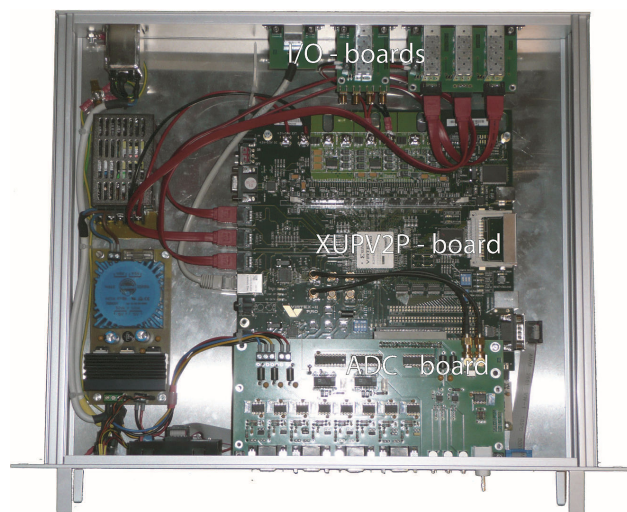


Figure 7: View of the XUPV2P board, the ADC board and the I/O boards inside the Extender 3000.

For orbit correction, several Extender boxes, distributed around the ring, were connected to fast power supplies using RS485 links (see Fig. 8). Horizontal and vertical correctors magnets were developed and installed at strategic positions, based on the ring optics. Promising first tests took place. The main problem turned out to be the synchronization of the DCC data between the Extended MX-BPMs and the Libera BPMs, especially when the Libera sample clock was tuned [12]. Also some Extender boxes had synchronization problems among themselves. Currently, the code for the Extenders is being rewritten from scratch.

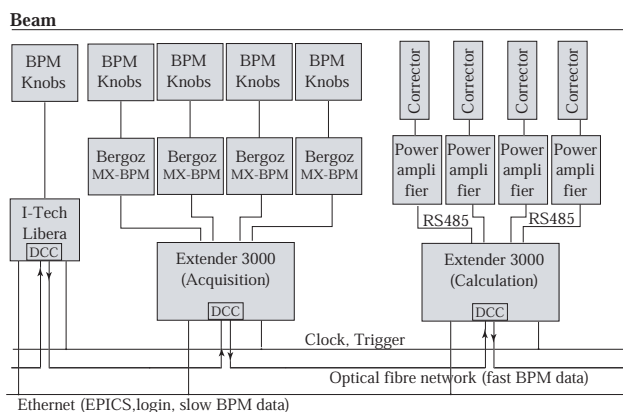


Figure 8: Functional groups of the fast orbit feedback at DELTA. Bergoz MX-BPMs, equipped with BPM-Extenders, deliver fast orbit data in parallel to Libera BPMs. The data are picked up by BPM-Extenders that calculate and apply orbit corrections to the electron beam.

CONCLUSION

At DELTA we have set up a local orbit feedback that allows us to damp orbit distortion up to a frequency of 350 Hz. The development of the Extender 3000 integrates Bergoz BPMs into the DCC communication network. A global orbit feedback was set up but showed synchronization problems between the Extender boxes and the Libera BPMs. The FPGA code is currently being rewritten.

ACKNOWLEDGEMENT

We appreciate help from N. Koch and B. Krumm from the electronics department of the Faculty of Physics, G. Rehm, M. Abbott and I. Uzun (Diamond Light Source), N. Hubert (Synchrotron SOLEIL) and D. Zimoch (PSI).

REFERENCES

- [1] N. Hubert, in *Proc. IPAC'16*, Busan, Korea, p. 3139.
- [2] <http://www.i-tech.si>
- [3] T. Weis *et al.*, in *Proc. RuPAC'06*, Novosibirsk, Russia, p. 138.
- [4] Bergoz Instrumentation, "Multiplexed Beam Position Monitor User's Manual", St. Genis Pouilly, France.
- [5] Leo R. Dalesio *et al.*, *Nuclear Instr. Meth. A* 352, p. 179
- [6] P. Towalski, Master Thesis, TU Dortmund, Germany, 2009.
- [7] I. S. Uzun *et al.*, in *Proc. ICALEPCS'05*, Geneva, Switzerland, P02.030-2.
- [8] https://reference.digilentinc.com/_media/virtex-ii:xupv2p_user_guide.pdf
- [9] G. Schünemann, Master Thesis, TU Dortmund, Germany, 2008.
- [10] <http://www.aps.anl.gov/epics/modules/soft/asyn/ASYN>
- [11] <http://epics.web.psi.ch/software/streamdevice/>.
- [12] M.G. Abbott *et al.*, in *Proc. ICALEPCS'09*, Kobe, Japan, p. 694.

HIGH-LEVEL APPLICATION DEVELOPMENT AND PRODUCTION INFRASTRUCTURE AT TRIUMF

E. Tikhomolov, Y. Bylinskiy, A. C. Morton*, T. Planche, T. Tateyama, J. Lee, P. Jung†
TRIUMF, Vancouver, BC, Canada

Abstract

TRIUMF users and operators use a number of high-level applications (HLAs) written in different languages, with complicated graphical user interfaces, to carry out tasks related to delivering ion beams with required characteristics and to process data from TRIUMF's EPICS-based and legacy cyclotron control systems. Some applications have been developed by the EPICS community, and some at TRIUMF. These applications run on different production computers and are developed on different machines. This model no longer satisfies TRIUMF's needs because of the growing number of applications, the long times required for data processing on current machines, the lack of real-time visualization of beam properties and so on. New infrastructure for HLA development has been implemented to address these issues and is working reliably with room for further expansion.

MOTIVATIONS

TRIUMF doesn't have a dedicated group of software developers tasked with immediate response to issues that arise during day-to-day beam delivery. Such issues are resolved by operators and physicists themselves. Thus, a flexible and simple ("user-friendly") software development environment was the main request when it was decided to set up High-Level Application (HLA) development and production infrastructure. At the same time, developers come with different experience, backgrounds and their own favorite tools for development. Thus, the HLA environment should provide some "default set" of tools which are rather common and compatible with other widely-used tools.

HLA SERVERS SET UP AND THEIR DIFFERENT ROLES

The projects that are under development at TRIUMF usually have three components: the code itself (in a number of programming languages), documentation and information, and input/output data. Thus, the common project can be represented as a point in a "three-dimensional space" where each component defines an axis, as illustrated in Fig. 1.

Some projects are just data projects which process experimental data (from TRIUMF's 520 MeV Cyclotron and Isotope Separator and Accelerator facility, or ISAC) for use with third-party applications. Each component has a corresponding location and directory structure: the code is developed on a development server (hladevel), input/output

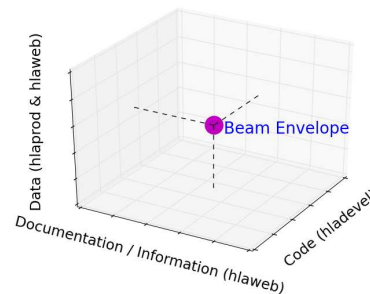


Figure 1: Common HLA project in "3D space".

data are saved on production servers (hlaprod and hlaweb), and documentation and information are maintained on a web server (hlaweb [1]).

To fulfil these tasks rack-mounted Dell servers were chosen [2]. All three servers have identical directories and software. This reduces maintenance time and allows one to easily switch from one server to another in the event of hardware failure. This approach also allows the maintainer to easily expand the infrastructure by cloning the existing servers, and redistributing the load. The differences between the servers are defined only by their different roles. Figure 2 shows these roles and the interactions between servers and user machines and control systems. The HLA development server hladevel and production server hlaprod are accessible only from the TRIUMF network. The web server hlaweb is accessible via the Internet and users can ssh to their accounts. Users have scratch directories (public_html) which may be used e.g. for presenting results at meetings. To reduce security threats it is not possible to connect to any TRIUMF computers from user accounts on hlaweb.

The data and user's scratch directories on the development server, hladevel, are synchronized (using rsync) with hlaweb every 4 hours. Data directories from hlaprod are also synchronized to hladevel. This duplication of data helps to avoid data loss. Nightly backups are done only on hladevel both on local USB drives (for fast restoration) and to a remote location (using TRIUMF's Amanda system [3]).

For version control Git [4] is used. Public and private remote repositories are located on hlaweb and hladevel, respectively. For projects which are expected to involve large number of inter-lab developers Github [5] may also be used.

CentOS 7 was chosen as the operating system for the HLA servers. This is already used at TRIUMF and supported by TRIUMF's Core Networking and Computing Group. To reduce the time spent on system administration the standard tools and applications provided with CentOS 7, at installa-

* Present affiliation: FRIB, MSU, East Lansing, MI, USA

† Present affiliation: Waterloo University, Waterloo, ON, Canada

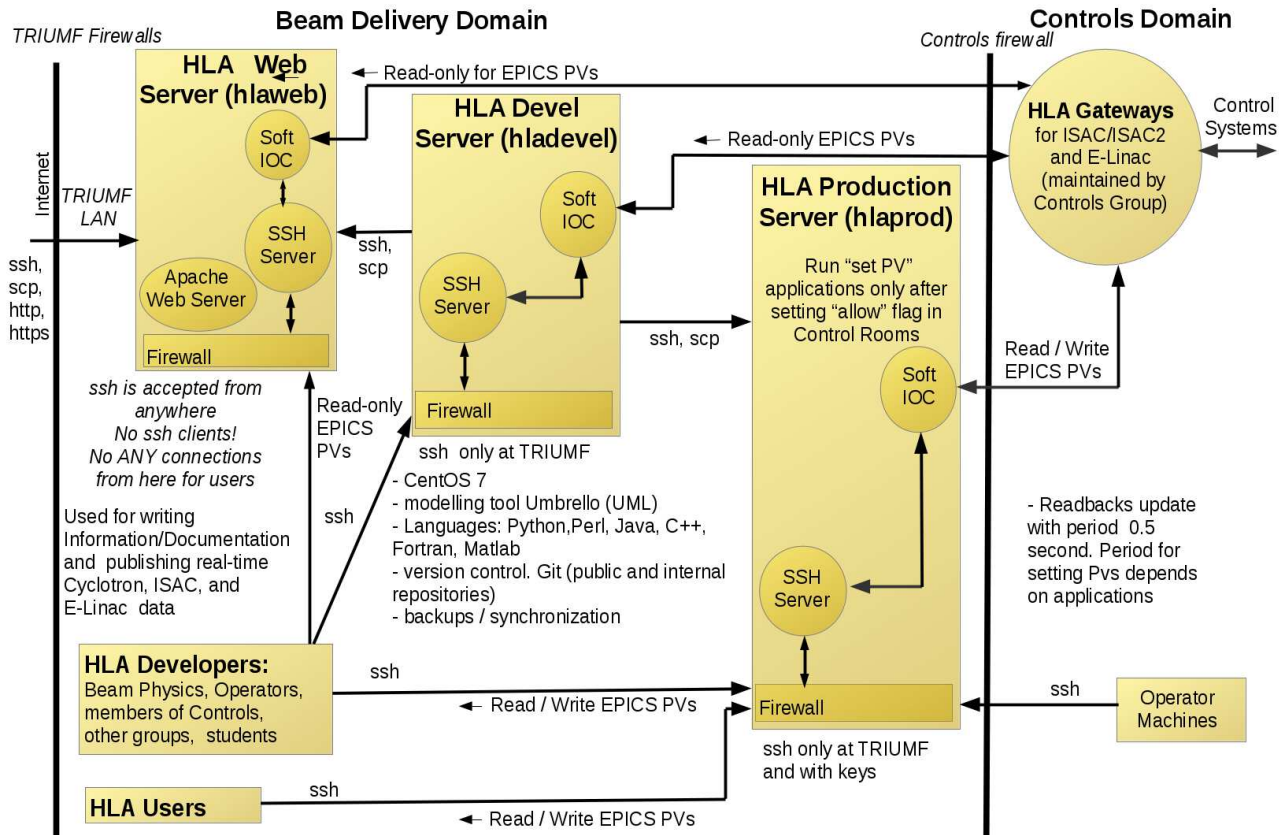


Figure 2: HLA servers and their roles.

tion, are used as much as possible. Users can start remote light-weight Xfce [6] desktops via the remote desktop application X2Go [7]. Graphics-intensive applications (like Matlab [8], Opera [9]) which run very slowly via X-forwarding are started in X2Go session in the "Single Application" mode. Usual tasks like synchronization and backups are done by using short customized scripts.

HLA DEVELOPMENT FLOW

High-level design of any large project starts with creating Unified Modeling Language (UML) diagrams. In our dynamic environment it is very important to create detailed high-level documentation which can be understood by future developers who may not have direct contact with the initial creator of the project. As the default, UML modeller Umbrello [10] is suggested; this comes with standard CentOS installation, and is relatively easy to learn and use. The choice of programming language is based on an evaluation of what is most appropriate for a particular project and on the experience of the current maintainers of similar projects. At present projects are written in Python, Perl, Java, C++, Fortran, and Matlab. For web applications, client side scripting is implemented in JavaScript and JQuery. The purpose of many HLAs is to get and set data from the TRIUMF control systems to collect and process useful analytics and hence improve the performance of the Cyclotron, ISAC, E-Linac,

and, in future, Advanced Rare IsotopE Laboratory (ARIEL) facilities.

Any information and documentation about the project is kept on the HLA web server hlaweb. To write information on line an in-house content management system (CMS) using TinyMCE JavaScript package [11] was developed and set up. The frame-based design of web pages is shown in Fig. 3. All modifications are done by clicking buttons in the Edit Bar, access to which is password-protected. When a new project is started, new directories, a frameset and initial Hypertext Markup Language (HTML) files are created with a single click. Editing Side Bar and Information HTML files is done using either TinyMCE or the simple textarea text editor. All of the necessary links for navigating between HTML files are created automatically in a hierarchical and easily maintainable manner. Navigation to necessary information is very efficient using the Navigation and Side Bars. No relational databases are used. For users who have accounts, it is also possible to create, edit, and link files over a SSH or remote desktop session by manually modifying them directly.

INTERFACE TO TRIUMF EPICS

Control systems at TRIUMF have both new and legacy components, Experimental Physics and Industrial Control Systems (EPICS [12]) for ISAC, E-Linac, ARIEL and the old VMS-based Central Control System for the 520 MeV

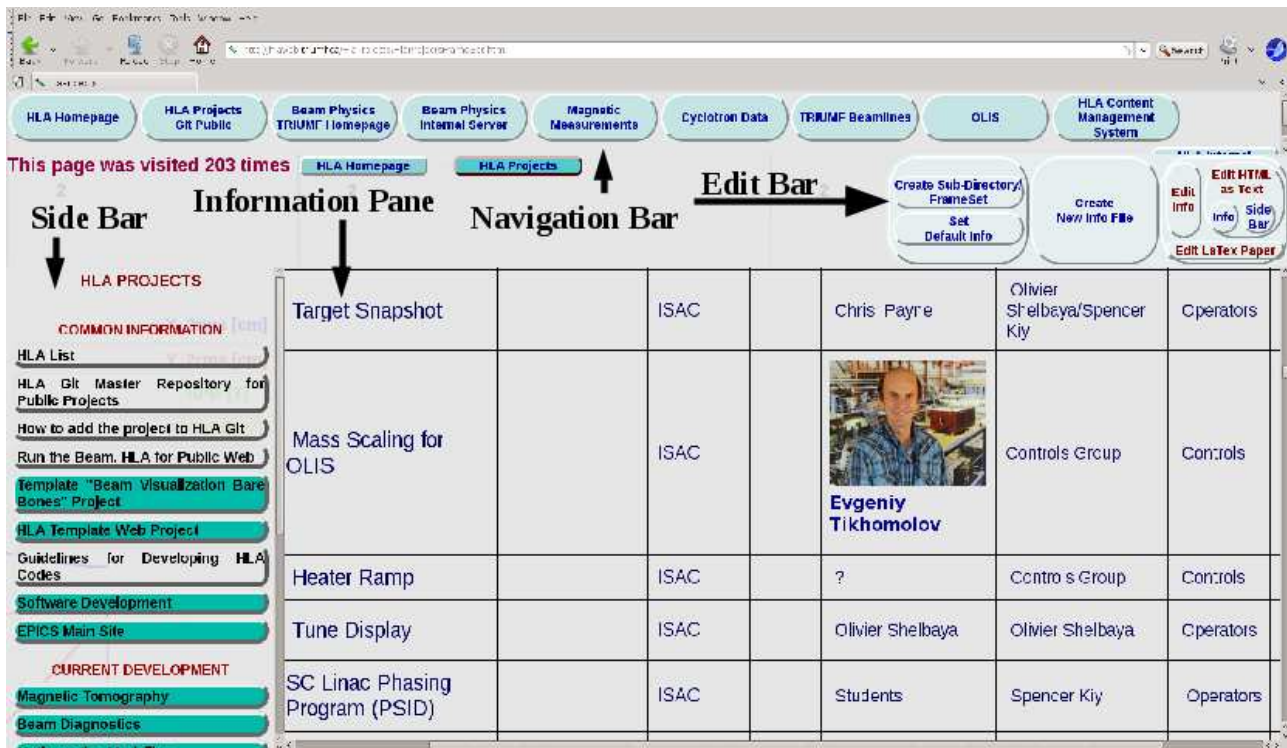


Figure 3: Frame-based design and editing web pages on hlaweb.

main cyclotron. Several years ago an interface between these legacy and new control systems was developed which can be used for data exchange between them. Thus, for HLAs we need only to communicate with EPICS servers. Interface between HLAs and EPICS is done in two steps (Fig. 2): HLA servers run local EPICS Soft IOCs which communicate with EPICS via gateways. Then HLAs get and set data from or in local Soft IOCs. Such a two-level design provides flexibility and complete decoupling from TRIUMF's Controls Group, which maintains only the gateways. Setting values for control system devices is allowed only from the production server hlaprod and only after the operators in Control Rooms set an "allow" flag with a time-out of 1 hour. This way running HLAs are always under the control of operations staff. For other servers, read-only access to EPICS is allowed. At present TRIUMF control systems have typical update times for read-backs of 500 milliseconds. The desire to maintain a similar update rate for processed data from HLA determines the requirement for data processing speed.

CONCLUSION

The created HLA infrastructure allows software developers to develop applications both by using the default set of tools installed on HLA servers and remotely on the computers of individual users. The sequential versions of projects are saved on HLA servers and are easily accessible by developers. Documentation for projects can be written on-line and is maintained in a very simple way using a dedicated HLA CMS. Several methods are used to run HLAs (which work with devices in TRIUMF control systems) in a secure

manner. If the load on the servers will be increased the expansion of existing infrastructures is quite easy and won't create significant additional time for the maintenance.

ACKNOWLEDGMENT

This work is funded by TRIUMF under a contribution from the National Research Council of Canada.

REFERENCES

- [1] TRIUMF HLA WEB Sever, <http://hlaweb.triumf.ca>
- [2] Dell PowerEdge Servers, <http://www.dell.com/ca/business/p/poweredge-rack-servers?~ck=anav>
- [3] Amanda Network Backup, <http://www.amanda.org/>
- [4] Git, distributed version control system, <https://git-scm.com/>
- [5] Github, <https://github.com>
- [6] Xfce Desktop Environment, <https://www.xfce.org/>
- [7] X2Go - everywherehome, <http://wiki.x2go.org/doku.php>
- [8] Matlab. The Language of Technical Computing, <http://www.mathworks.com/products/matlab/>
- [9] Opera Simulation Software, <http://operafea.com/>
- [10] Umbrello, the UML Modeller, <https://umbrello.kde.org/>
- [11] TinyMCE, Web Editing, <https://www.tinymce.com/>
- [12] EPICS, <http://www.aps.anl.gov/epics/>

STATUS OF THE NSLS-II LLRF SYSTEM*

C. Marques[†], J. Rose, N. Towne, B. Holub, F. Gao, G. Wang
NSLS-II, BNL, Upton, NY 11973, U.S.A

Abstract

The NSLS-II RF system uses an in-house FPGA based low level RF (LLRF) solution called the Cavity Field Controller (CFC). The CFC directs the amplitude and phase for the high power RF and directly influences beam acceleration and stability. In this paper we discuss a logically embedded network analyzer in situ with the digital feedback loop controlled via a MATLAB or EPICS interface. The embedded NA was used to evaluate the RF feedback stability and influence of the feedback parameters on the beam. We will also discuss diagnostics tools to investigate longitudinal beam dynamics and other functionality embedded into the FPGA fabric. Future development of the CFC implementation and hardware upgrades will also be discussed.

INTRODUCTION

The Field Programmable Gate Array (FPGA) based CFC was developed as a common hardware platform for all NSLS-II RF systems. The reconfigurable logic on the FPGA allows the CFC to run in multiple modes of operation. For example, open loop feedforward (FF), closed loop feedback (FB) and a combination of the two. In addition to operating modes the CFC is able to run diagnostics, cavity tuner loop adjustments, data buffering, event handling including interlocks and real-time data acquisition simultaneously. The CFC includes eight 500 MHz RF inputs and one 500 MHz RF output along with multiple IO which can be seen in Figure 1. More information about the digital controller can be found elsewhere [1].

CFC FUNCTIONS

One of the primary functions of the CFC is a feedback control loop of the cavity field. A simplified schematic of the feedback loop and other CFC logic can be seen in Figure 2. The feedback loop includes digitized vector signal detection provided by dual 14-Bit ADC LTC2299s at 80 MHz (40 MHz/channel) and can be seen in the schematic by the $I, Q, -I, -Q$ sequence just after the cavity pickup. The $I, Q, -I, -Q$ notation comes from the fact that the 50 MHz IF is sampled at half the clock frequency or 40 MHz. The digitized sequence is then corrected for feedback path latency of approximately 1.1 ms using a phase rotation provided by the $Kp(I), Kp(Q)$ multipliers and where the total Kp magnitude is $|Kp| = \sqrt{Kp(I)^2 + Kp(Q)^2}$. Digital signal processing using a host-front end is used to calculate and properly adjust the loop phase rotation. The summed I and

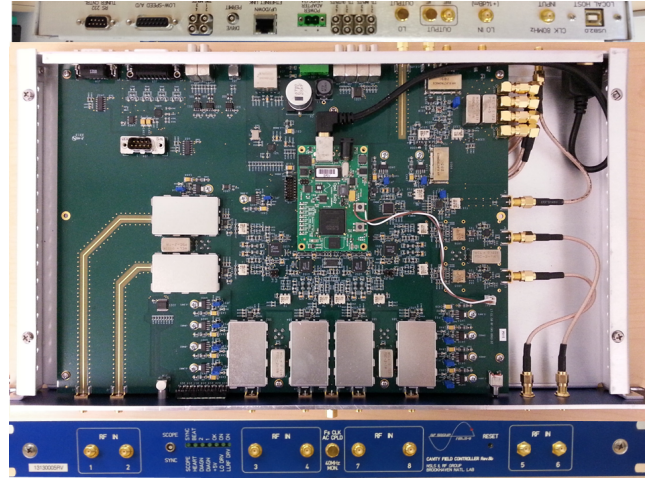


Figure 1: Picture of the CFC. The top and bottom of the picture shows the different IO on the back and front panels respectively.

Q pair is multiplied by the integral gain factor Ki and subsequently summed. It is worth noting that the integral gain Ki is dependent on the proportional gain Kp and hence so is the corner frequency or bandwidth of the feedback loop. The summed error signal produced in the integral leg of the feedback loop not only provides for error correction but also synthesizes the output drive signal for the CFC.

Network Analyzer

The logically embedded vector network analyzer produces a direct digital synthesized (DDS) stimulus which is summed with the output drive of the controller. The network analyzer module performs a fourier analysis on the CFC readbacks to produce response functions of those readbacks normalized to the DDS output. Much like an S_{21} measurement on a conventional network analyzer except in this case we have several inputs including the feedback loop and longitudinal beam motion from a summed beam position monitor (BPM). Using a narrow bandpass filter of 540 kHz with a center frequency of 499.68 MHz, we eliminate any problematic signals including the charge induced by an empty bunch train from the storage ring fill pattern and we are able to measure an unambiguous beam response. Figure 3 shows the amplitude response function of the cavity field and BPM. The figure shows that for a fixed integral gain Ki and DDS amplitude, the proportional gain Kp influences both the beam and cavity response to the stimulus. For higher Kp gain the effect of the beam from perturbations is decreased and suggests higher stability and lifetime. This effect is also shown by the narrowing of the peak at the synchrotron frequency for increasing Kp values. Figure 3 also shows

* Work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-SC0012704.

[†] cmarques@bnl.gov

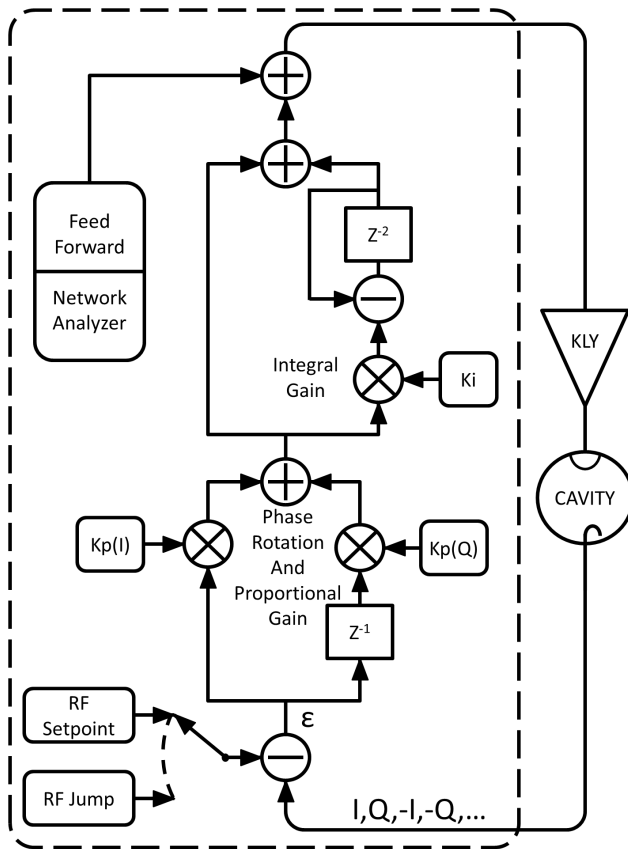


Figure 2: Simplified schematic of the CFC logic and feedback loop.

typical bandwidth extension of the cavity field for increasing Kp at the expense of the amplifier gain.

Circular Buffer

The circular buffer module in the FPGA code continuously writes streaming data from the CFC inputs to the on-board RAM in a cyclic fashion. In the event of a circular buffer trigger the data stream continues to be written into the RAM approximately half of the memory capacity. Since there is an $I, Q, -I, -Q$ data stream at 40 MHz, at least two data points to establish amplitude and phase. Currently we continuously monitor five channels which yields time resolution of 250 ns. With 1703936 samples we can view ~ 213 ms of data before and after the trigger. This data can then be imported into a host computer and processed into a waveform.

Phase Jump

The feedback loop error signal is generated by subtracting the cavity field input from the setpoint data stream. The phase jump module in the FPGA code allows switching of RF setpoints, more precisely the I/Q set points and so can either be an amplitude and/or phase jump. A phase jump was used in beam studies where phase jumps of 100 degrees or more with beam were implemented. The limiting factors for the magnitude of the phase jump are the forward and reverse power limits of the RF system. These are in turn

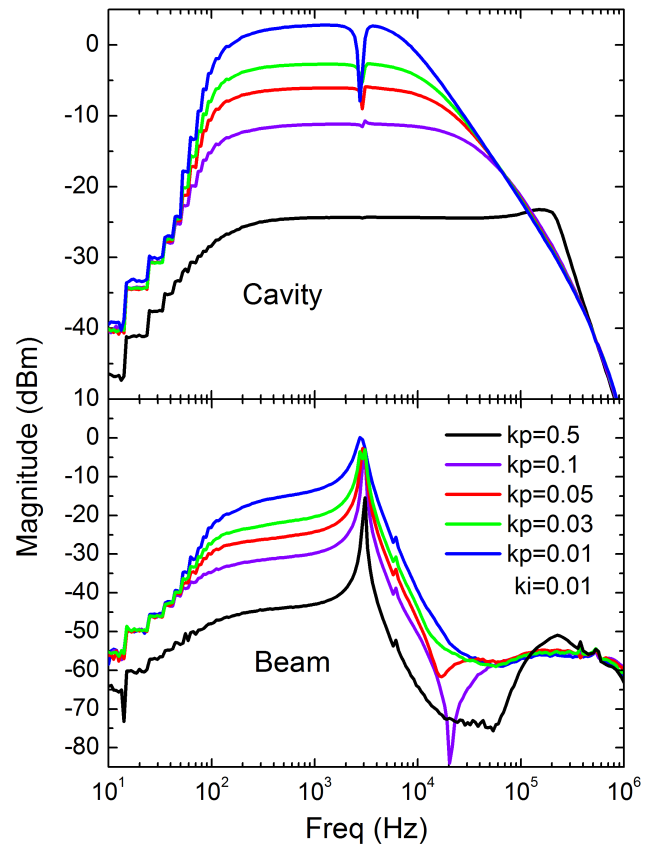


Figure 3: Cavity and beam frequency response. For the above plots there are two cavities 1.2 MV each and 150 mA beam current in the ring. Integral gain is constant at 0.01, proportional gains are 0.01, 0.03, 0.05, 0.1 and 0.5 respectively.

influenced by the size of the phase jump (180 degrees gives maximum power requirement) and the time to complete the jump. This time depends in turn on the feedback gain set points. When the phase jump is triggered the set points immediately change to new values. The cavity field fed into the controller is subtracted from this set point and the error is multiplied by the feedback gains to drive the system to the new values. An example phase jump can be seen in Figure 4 where the cavity field phase was altered by more than 100 degrees causing the beam phase to also readjust by more than 100 degrees. The RF phase jump can induce longitudinal beam oscillations which can be seen in the inset of Figure 4 and serves as a powerful tool for investigating longitudinal beam dynamics.

Using this module and turn-by-turn BPM data several experiments were carried out, i.e. measured the NSLS-II ring momentum aperture with and without damping wigglers, we located the momentum aperture limitation and initiated beam crossing major resonance under the stopband width control [2].

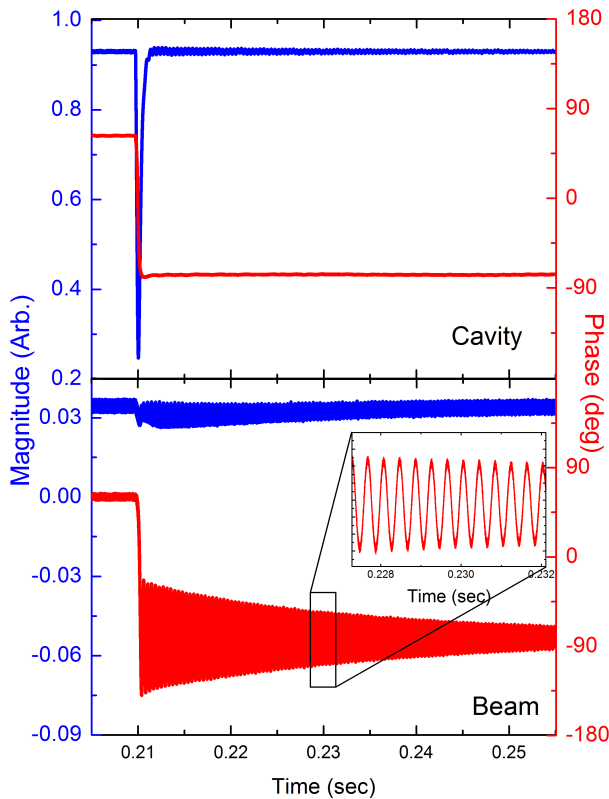


Figure 4: Circular buffer data of a phase jump experiment. The blue and red traces represent amplitude and phase respectively.

Pulsed Gains

The feedback control loop discussed above uses programmable feedback gain settings. An NSLS-II RF system model suggests that system stability might be achieved with gain settings that are a factor 100 higher than are currently in use. The high gains would also make them hard to find if the tolerance about such settings is small, or if there is no stable path between current settings and the higher ones. Such values would be highly advantageous not only because of the quick damping of synchrotron oscillations, but also for greater suppression of noise in the system at other frequencies, such as power-supply noise. The pulsed gains module allows the logic to switch repetitively to alternate gain settings at various timescales while the user watches a scope trace for instabilities. When stable settings are found, those settings are fine tuned and the switched time is progressively increased to verify stability. When running repetitively at a rate of once or twice a second, the gain/phase parameter space should be quickly found, even areas far removed from current settings.

This technique is used in other contexts, such as when pulsing coupled-bunch feedback off to measure growth rates of individual modes, when switching between cavity-phase setpoints as a means to study longitudinal dynamic behavior

described above, and the analogous technique of pulsing feed forward when tuning feedback.

DEVELOPMENT

Although the CFC is highly versatile and expandable there is still room for improvement. One such improvement is converting to a FPGA System on Chip (SoC) device which integrates the programmable software of a processor with the programmable hardware of an FPGA. Software programmability should include supporting an operating system such as Linux or comparable to handle many functions for the FPGA. These functions include communications protocols, filesystems, DSP, peripheral device handling and any host operation described above provided the right drivers and appropriate hardware is installed. The 32 MB system RAM on the CFC is also a bottle neck and must be upgraded to support larger feedforward/feed-back ramp tables and potentially longer circular buffer data. High speed transceiver capabilities are also a possibility for inter CFC communications and system diagnostics which would also free up some of the IO on the CFC. NSLS-II uses event generators (VME-EVRRF-230, PMC-EVR-230, etc.) to create a timing system [3]. Currently the CFC does not support event receivers (EVRs) and so having EVR capabilities would eliminate timing jitter, improve beam studies and overall CFC functionality.

SUMMARY

An FPGA based CFC is being used successfully in the NSLS-II RF systems. The flexibility of the FPGA allows for developing powerful tools embedded in logic. Tools including a vector network analyzer, circular buffer, phase jump and pulsed gains were used to further understand and improve the RF systems. Further development of the CFC is ongoing.

ACKNOWLEDGMENT

The authors wish to acknowledge the help and discussions of Jueri Tagger and Ruslan Kadyrov. In addition the continuous support from the NSLS-II Controls and Utilities groups.

REFERENCES

- [1] B. Holub *et al.*, "NSLS-II Digital RF Controller Logic and Applications", in *Proc. IPAC'15*, paper MOPTY038.
- [2] G. Wang *et al.*, "RF Pinger Commissioning and Beam Dynamics Studies at NSLS-II", in *Proc. IPAC'16*, paper THOBA01.
- [3] M. Davidsaver, "EVR Usage Guide", BNL Controls Internal Document, August, 2015 Rev. 7, <https://wiki.bnl.gov/nsls2controls/index.php/Timingref>

OVERVIEW OF SOME FEEDBACK- & CONTROL SYSTEMS AT SYNCHROTRON SOLEIL

C. Engblom*, F. Alves, F. Blache, D. Corruble, A. Dawiec, M. Diop, N. Hubert,
S. Kubsy, F. Langlois, P. Marchand, N. Jobert, G. Renaud, Y.M. Abiven
Synchrotron SOLEIL, Saint-Aubin, France
T. Stankevic, MAXIV, Lund, Sweden

Abstract

This paper gives an overview of some feedback & control systems at Synchrotron SOLEIL that are in use or in development today. Beam stability is crucial and addressed in all SOLEIL aspects; Fast Orbit Feedback is a multi-input multi-output control system made to stabilize beam position perturbations in the low- & high frequency band. In addition, active RF cavities are used to maintain stable beam energy & spread as well as keeping electron density even throughout the storage ring. Beam stability also comes from feedforward non-linear control in particle trajectory compensation on both sides of electromagnetic undulators. On some beamlines, multi-actuator piezos or pneumatics are used to regulate photon flux to keep within detector operating range; a method to maximize the photon flux while still keeping below detector damage thresholds. Currently in development & at the sample stage level, the Nanoprobe Project collaboration (MAXIV & SOLEIL) focuses on sample stabilization during step- & fly- scans which is realized through multi-axis nano-positioning with high- & low- frequency closed-loop control implementing interferometer feedback &/or compensation tables.

INTRODUCTION

For the past couple of years Synchrotron SOLEIL has, in a collaboration project with MAXIV, been constructing an endstation prototype capable of 2D- and 3D tomography scans on the nanometric scale. Such a project not only relies on passive thermal stability and vibration but also on feedback systems. These systems are not only crucial in the endstation setup but also relies on the stability provided by a series of feedback systems that resides in the storage ring, entry and exits of insertion devices, radiofrequency systems and photon flux regulating devices to provide stable photon beam positioning and flux to beamline endstations. In long beamlines, such as Nanoscopium [1] in Synchrotron SOLEIL or NanoMax [2] in MAXIV, endstations particularly specialize in scanning x-ray microscopy on the nano-scale.

As such, this paper provides an overview of some of the existing feedback systems in Synchrotron SOLEIL with a focus on the progress of the Nanoprobe project.

BEAM ORBIT STABILITY SYSTEM

Stable photon flux to the beamlines is in part provided by ensuring electron beam position and angle stability in the

storage ring. For this reason the Global Orbit Feedback System (GOFB) is necessary against environment perturbations in the long term (hours to a day, ex: thermal effects, sun & moon tides), medium term (seconds to minutes, ex: moving crane, insertion devices), and short term (less than a second, ex: booster cycling operations, ground vibrations) [3]. The frequency spectrum of the noise at SOLEIL Synchrotron has been shown to reside in the range from DC→150 Hz [3].

The GOFB correction algorithm is based on Singular Value Decomposition (SVD) where an inverse response matrix, R^{-1} , together with the orbit error, ΔU_{BPMi} , is used to calculate the correction currents, ΔI_{corrj} [3]. Multiplication is shown in Eq. 1, the matrix dimensions of R^{-1} are here denoted as $M \times N$ and directly corresponds to the number of Beam Position Monitors, N , and number of current actuators, M .

$$\begin{bmatrix} \Delta I_{corr j} \\ \Delta I_{corr j+1} \\ \vdots \\ \Delta I_{corr j+M} \end{bmatrix} = R^{-1} \cdot \begin{bmatrix} \Delta U_{BPM i} \\ \Delta U_{BPM i+1} \\ \vdots \\ \Delta U_{BPM i+N} \end{bmatrix} \quad (1)$$

The GOFB is divided into two systems [3]:

- The Slow Orbit Feedback (SOFB), with correction rate and bandwidth limited to 0.1 Hz
- The Fast Orbit Feedback (FOFB), with correction rate to 10 kHz and efficient up to a few hundred Hz.

The two systems contain their own sets of correctors and can run independently of each other. This can cause interference if their control-frequencies are overlapping which is why there are three approaches that can be used [3]:

1. 'Deadband' method (Fig. 1), a frequency deadband is introduced between the two systems guaranteeing complete independency from each other. The issue with this method is that the deadband needs to be sufficiently large and if there are components creating disturbances inside the deadband spectrum they can't be corrected for.
2. 'FOFB only' method (Fig. 2), run the FOFB system alone for all frequencies down to DC. This approach, due to the weakness of fast correctors, have limits on correction amplitudes and might saturate the correctors.
3. 'FOFB/SOFB interaction' method (Fig. 3), the SOFB corrects low-frequency disturbances while predicting

* christer.engblom@synchrotron-soleil.fr

its next-iteration corrections. It then subtracts the predicted changes from the next-iteration FOFB correction.

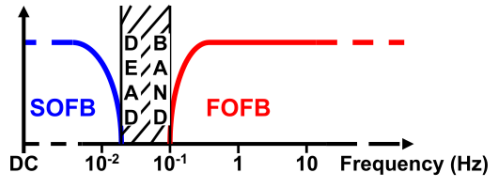


Figure 1: Deadband approach.

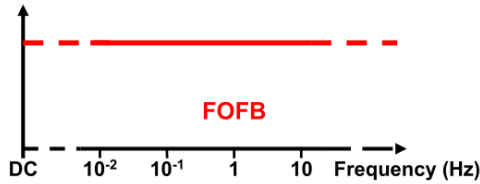


Figure 2: 'FOFB only' approach.

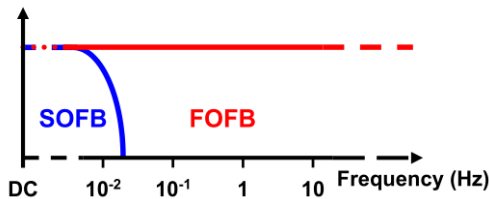


Figure 3: 'FOFB/SOFB interaction' approach.

Status of the GOFB at Synchrotron SOLEIL

SOLEIL is currently using the 'FOFB/SOFB interaction' approach with a DC-download algorithm which allows it to do corrections in the frequency domain from DC \rightarrow 250 Hz [3]. Two different sets of correctors are being used: strong iron-core correctors for SOFB (bandwidth limitations), and low-inductance correctors for FOFB (high bandwidth, but with lower degree of corrections) [3]. From these, SOLEIL can benefit from having the fast correction without saturation.

The GOFB has been using in total 122 e-BPM as sensors and has been doing so since 2008 [4]. Since 2013 however, XBPMs have been included as additional sensors in the correction algorithm; since the XBPMs are situated in the front-end, they provide a better position angular measurement of the beam [4]. They are implemented into the SVD SOFB system and have been able to improve peak-to-peak position stability of the photon beam by a factor of 1.3 – 3 [4].

STORAGE RING RF SYSTEMS

In the SOLEIL storage ring the RF system provides the power of 600 kW and the RF voltage of 3-4 MV at 352 MHz, which are required to store 500 mA at the nominal energy

of 2.75 GeV. This is achieved using 2 cryomodules, each containing a pair of superconducting cavities. Each of the four cavities is powered by a 180 kW solid state amplifier and the two cryomodules are supplied with liquid helium at 4.5 K from a single cryogenic station [5].

Figure 4 shows a block diagram of one of the four low level RF (LLRF) systems which allows to control the cavity accelerating voltage with the required level of stability in terms of frequency, amplitude and phase. The amplitude loop (Fig. 4, orange) regulates the amplitude of the cavity accelerating voltage by comparing the signal from a cavity pickup with a reference voltage; the error signal acts on a variable attenuator through a PID. The phase loop (Fig. 4, green) maintains constant the phase between the RF generator and the cavity voltage by driving a phase shifter with the error signal from a phase comparator through a PID. Each cavity has its own frequency tuner, a mechanism driven by a linear stepper motor, which changes the cavity length and therefore its resonant frequency. The motor is controlled by the error signal from the tuning loop (Fig. 4, in grey) and a SOLEIL standard ControlBox unit (Fig. 5). The fast direct RF feedback (Fig. 4, in red), which reinjects a sample of the cavity RF voltage at the input of the amplification chain, is aimed at coping with the Robinson instability at high beam loading. The achieved stability is 0.1% in amplitude and 0.1° in phase.

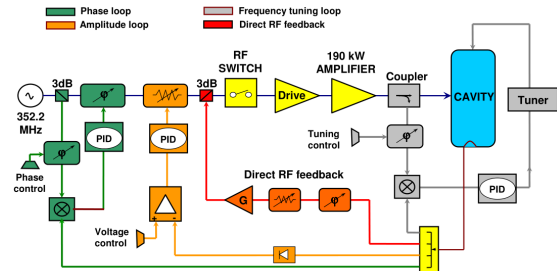


Figure 4: Low-Level Control scheme of an RF cavity.

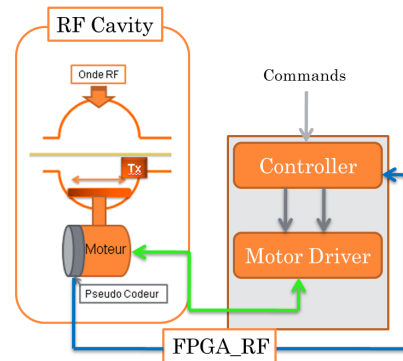


Figure 5: RF cavity motor control scheme. The stepper motor is controlled by a SOLEIL ControlBox using pseudo encoder feedback (in the form of an analog signal) from an FPGA system.

In the Booster, a 5-cell normal conducting copper cavity, powered with a 35 kW solid state amplifier, provides an RF

voltage of 1 MV at 352 MHz. The LLRF system is similar to that of the storage ring, but there is no need for direct RF feedback as the beam current is much lower. Although we have developed a prototype of digital FPGA-based LLRF system, which could achieve similar performance, we are still using our original fully analog LLRF system [6].

TRAJECTORY COMPENSATION IN ELECTROMAGNETIC INSERTION DEVICES

In order to correct for close orbit distortions (COD) in electromagnetic insertion devices (ID), a fast electromagnetic field variation in the correctors (at entry/exit of the ID) and in the ID is needed [7]. Previous control systems used high-level software for control via Profibus which had limitations on COD synchronization and electromagnetic switching rates [7]. In order to improve this performance, a new control system based on a set of boards (“SPI BOARD PACKAGE”, developed at SOLEIL) was implemented [7]. Figure 6 shows the control architecture using the SPICONTROLLER and SPIDAC configuration, a low-level micro-controller which in this case manages the feedforward compensation in the form of look-up tables. These tables, derived from position measurements from the ID, generate corrections analog signals (for the correctors and main ID coil) in function of main coil current [7]. A TANGO high-level software is used to configure the SPICONTROLLER [7].

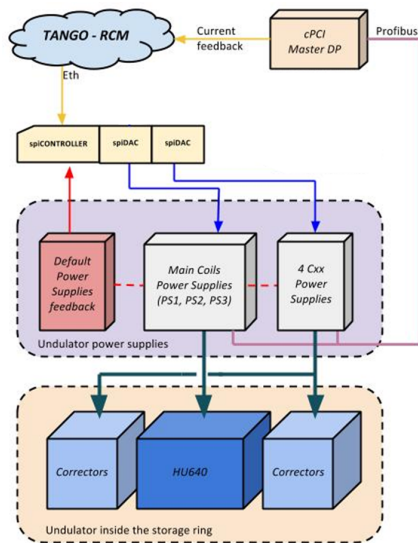


Figure 6: Control architecture of HU640 insertion device control with SPIBOARDS package.

Figure 7 shows the results from using the SPICONTROLLER setup on a HU256 undulator: the COD were reduced by as much as a factor of 3 down to $8 \mu\text{m}$ [7].

FAST ATTENUATION DEVICE

The Fast Attenuation (FastAtt) device was originally implemented for the SIXS beamline in 2012 in order to regulate

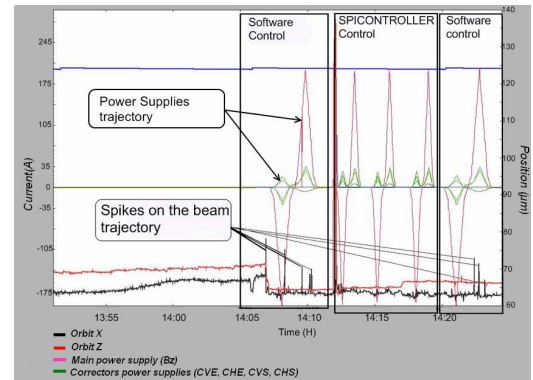


Figure 7: The difference between old and new control system of HU256 undulators. Beam trajectory spikes are reduced from about $25 \mu\text{m}$ down to $8 \mu\text{m}$ using the new control system. The center of the figure shows the power supply trajectory and in the bottom is the machine beam orbit. In the graph, the old control system is identified by software control and the new one by SPICONTROLLER control.

the photon flux to keep below detector damage thresholds and keep within their optimal operating range [8]. The system contains a control unit, here based on a CompactPCI board, and attenuator actuators (See Fig. 8) [8]. The control board measures the mean photon flux from a detector and compares the value with pre-programmed thresholds and makes the necessary actuators adjustments [8]. Depending on the photon detector, two different architectures are possible: a point detector (since 2012) or an XPAD 2D-sensor [9] (since 2015). Both systems are in use depending on the application. In the case of the XPAD- setup (see Fig. 8), an intensity analyser board performs image processing.

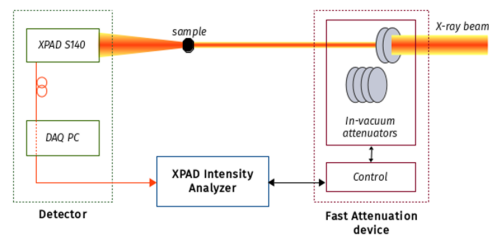


Figure 8: Global architecture of the controlled 2D XPAD attenuation system.

The FastAtt actuators each hold photon filters which are controlled with binary logic that moves each filter independently in and out of the photon beam pathway. Combined control of all actuators then provides for a very high range of beam attenuation. This is particularly useful when performing scans over high photon energy ranges that would normally saturate detectors at certain points, the FastAtt system would simply apply the necessary filtering and avoid detector saturation.

The attenuation reaction time is an important factor as it determines how useful the system would be for continuous scans such as Flyscans [10] the pneumatic actuators have shown to have a 150 ms reaction times, and 10ms for piezo

actuators [8]. Today, the SIXS beamline uses two FastAtt setups: one with pneumatic actuators and another with piezo driven ones.

NANOPROBE PROJECT

The Nanoprobe Project was initiated to deliver a scanning hard X-ray Fresnel Zone Plate (FZP)- based microscope with a scanning sample stage for long beamline endstations in Nanoscopium in Synchrotron SOLEIL and NanoMAX in MAXIV. Some of the challenging aspects were to produce nanometric precision coupled with millimeter range and 360° sample movement and rotation, while also providing Flyscans [10] and long-term stability. Figure 9 shows a schematic of the endstation setup with beam focusing stages (Fresnel Zone Plates, Central Stop, Order Sorting Aperture) and Sample Stage.

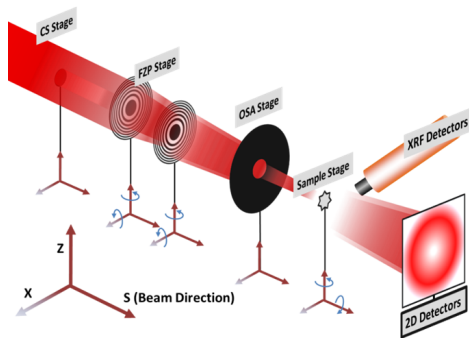


Figure 9: End-station scheme of stages and detectors and their orientation in respect to the beam.

The approach was to, in addition of providing a stable environment in terms of vibration and temperature, to construct a modular and stacked design with an interferometric feedback system and the possibility of using position compensation (in feedforward control) to diminish repeatable errors of the linear and rotation stages.

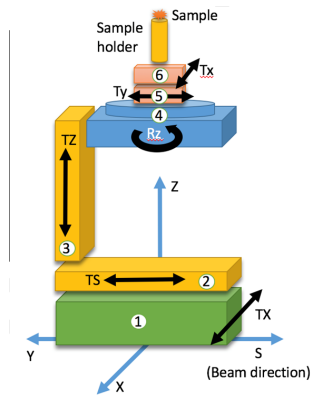


Figure 10: Sample stage schematic setup.

Figure 10 gives a schematic example of how the sample stage was built; a stacked design with (X, S, Z, Rz) from numbers 1-4 and eccentricity- correction stages numbered on

5-6. The sample holder is a cylindrical reflector, providing feedback for interferometer readings.

Control Schemes

Control for the different stages was achieved by utilizing two main schemes; one that is used for fast and dynamic control (see Fig. 11) while the other one provides for slow point-to-point static positioning (see Fig. 12). These control schemes are cascaded with an inner loop (usually higher frequency) residing in the actuator driver, and an outer loop containing either a Delta Tau or a Galil as superior controllers. Due to the stacked and modular design, the same control scheme as depicted in Fig. 11 can be used for simultaneous motion of all axes within the sample stage without the need of resorting to kinematic conversions. The scheme depicted in Fig. 12 was used in the beam focusing stages such as the FZP stage. Positioning errors in the form of parasitic movements, thermal drifts, and vibrations were still evident at the nanometric level which were diminished by two main approaches:

1. *Position Compensation*: repeatable errors were measured and compensated for in a feedforward manner. This approach needs interferometry for building the compensation table. Position compensation can be linked between axes, effectively creating compensation in several degrees of freedom.
2. *Interferometry Closed-Loop*: closed-loop control using interferometry at the Delta Tau/Galil controller level: interferometry feedback on the sample stage is done very close to the sample (a few mm) and it is possible to correct for errors at the same frequency in which they are perceived. Assuming a perfect reflector, any perceived error is the actual position error. If the reflector is not a perfect cylinder, its shape needs to be determined in an independent measurement. The shape is then used for mapping on which position compensation on the interferometry signal can be done.

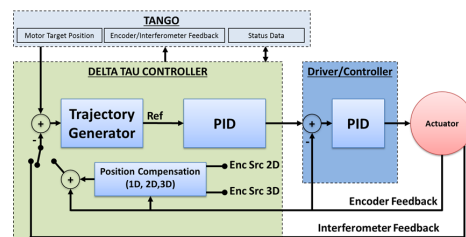


Figure 11: Sample stage control scheme with a Delta Tau Controller.

Sample Stage Scan Results

2D- scanning performance with continuous and stepping motion along the XZ-axes were tested using the different control schemes to diminish positioning errors. In Fig. 13 the motion errors are shown during $1 \times 1 \mu\text{m}$ scans with continuous Flyscan [10] acquisition in the X-axis while performing

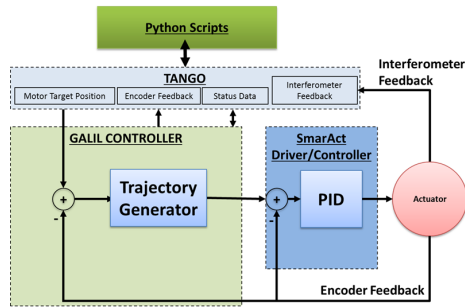


Figure 12: FZP stage control scheme with a Galil controller.

step-scanning on the Z-axis; here closed-loop with direct feedback from the interferometers was done yielding X-error of 8.2 nm FWHM and 2.8 nm FWHM on the Z-error. The errors were calculated by subtracting the reference trajectory from the displacements measured using interferometry.

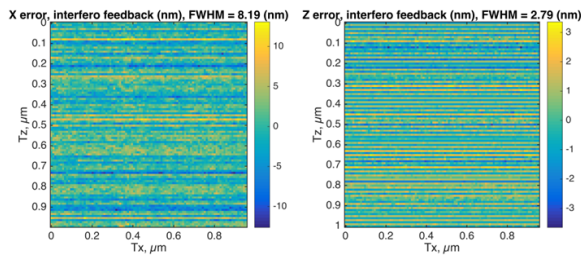


Figure 13: Positioning errors during $1 \times 1 \mu\text{m}$ 2D continuous scans. Continuous scans were done along the X-axis and Z-Axis performed steps in 10 nm increments. Correction of X and Z errors by using closed-loop interferometry feedback. Color bar is in nanometers.

FZP Active Stabilization Results

An active closed-loop stabilization with interferometers (as depicted in Fig. 12) was implemented that calculated the drift of the FZP holder from the interferometry measurements and kept it within a set threshold by commanding the corresponding positioner movements. Multi-axes control with kinematic equations was used and programmed in a python script that acted through a TANGO device. The control frequency was about 1 Hz. As seen in Fig. 14, the FZP position kept within approximately 5 nm FWHM.

REFERENCES

- [1] A. Somogyi *et al.*, "Status of the Nanoscopium scanning hard x-ray nanoprobe beamline of synchrotron SOLEIL", in *11th Int. Conf. on X-ray Microscopy (XRM2012)*, Shanghai, China, Aug. 2012.
- [2] U. Johansson *et al.*, "NanoMAX: a hard x-ray nanoprobe beamline at MAX IV", in *Proc. of SPIE Vol. 8851 88510L-9*, 2013.
- [3] N. Hubert, L. Cassinari, J-C. Denard, A. Nadj, L. Nadolski, "Global orbit feedback systems down to DC using fast and slow correctors", in *Proc. 9th European Workshop on Beam Diagnostics and Instrumentation for Particle Accelerators (DIPAC'09)*, Basel, Switzerland, May 2009, paper MOOC01, pp.27-31.
- [4] N. Hubert *et al.*, "SOLEIL beam stability status", in *Proc. 4th Int. Particle Accelerator Conf. (IPAC'13)*, Shanghai, China, May 2013, paper WEPME001.
- [5] P. Marchand *et al.*, "Commissioning of the SOLEIL RF systems", in *Proc. 10th European Particle Accelerator Conf. (EPAC'06)*, Edinburgh, Scotland, June 2006, paper MOPCH142, pp.384-386.
- [6] M. Diop *et al.*, "Low level RF system development for SOLEIL" in *Proc. 10th European Particle Accelerator Conf. (EPAC'06)*, Edinburgh, Scotland, June 2006, paper TUPCH186, pp.1447-1449.
- [7] Y.M. Abiven *et al.*, "Analogue feedforward for SOLEIL electromagnetic insertion devices", in *Proc. 11th Int. Conf. on Synchrotron Radiation Instrumentation (SRI2012)*, Lyons, France, July 2012, pp.201-204.
- [8] G. Renaud *et al.*, "Beamline fast and automatic attenuation system for x-ray detectors at Synchrotron SOLEIL", in *Proc. 11th Int. Conf. on Synchrotron Radiation Instrumentation (SRI2012)*, Lyon, France, July 2012.
- [9] S. Basolo *et al.*, "XPAD: pixel detector for material sciences", in *IEEE Transactions on Nuclear Science, Institute of Electrical and Electronics Engineers*, 2005, 52, pp.1994-1998.
- [10] N. Leclercq *et al.*, "Flyscan: A fast and multi-technique data acquisition platform for the SOLEIL beamlines", in *Proc. 15th of Int. Conf. on Accelerator & Large Experimental Physics Control Systems (ICALPCS'15)*, Melbourne, Australia, Oct. 2015, paper WEPGF056, pp.826-829.

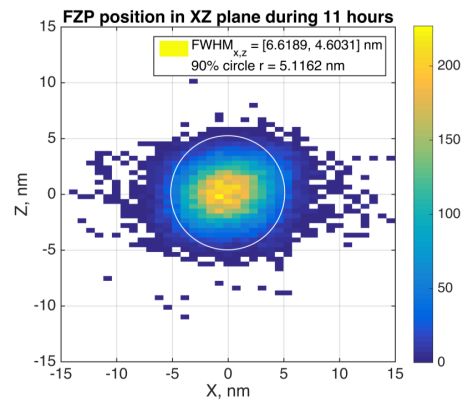


Figure 14: 2D histogram of the FZP XZ-position during 11 hours of active stabilization. The FWHM is 6.6 nm in X and 4.6 nm in Z. 90% of the time the FZP position is within $r = 5.1$ nm circle.

A FAST, CUSTOM FPGA-BASED SIGNAL PROCESSOR AND ITS APPLICATIONS TO INTRA-TRAIN BEAM STABILISATION

G. B. Christian*, N. Blaskovic Kraljevic, R. Bodenstein, T. Bromwich,
P. N. Burrows, C. Perry, R. Ramjaiwan, J. Roberts¹

John Adams Institute for Accelerator Science at the University of Oxford, Oxford, UK

¹also at CERN, Geneva, Switzerland

Abstract

A custom 9-channel feedback controller has been developed for low-latency applications in beam-based stabilisation. Fast 14-bit ADCs and DACs are used for high-resolution signal conversion and a Xilinx Virtex-5 FPGA is used for core high-bandwidth digital computation. The sampling, and fast digital logic, can be clocked in the range 200 to 400 MHz, derived from an external or internal source. A custom data acquisition system, based around LabVIEW, has been developed for real-time control and monitoring at up to 460 kbps transfer rates, and is capable of writing and reading from EPICS data records. Details of the hardware, signal processing, and data acquisition will be presented. Two examples of applications will also be presented: a position and angle bunch-by-bunch feedback system using strip-line beam position monitors to stabilise intra-train positional jitter to below the micron level with a latency less than 154 ns; and a phase feedforward system using RF cavity-based phase monitors to stabilise the downstream rms phase jitter to below 50 fs with a total latency less than the 380 ns beam time-of-flight.

INTRODUCTION

Many modern particle accelerators and future colliders require the generation and preservation of low emittance beams with a high degree of stability. Future electron-positron collider designs, such as the International Linear Collider (ILC) [1] and the Compact Linear Collider (CLIC) [2], call for beam spot sizes of 5 nm and below at the interaction point (IP), in order to maximise the luminosity. In order to achieve the design luminosity, $2 \times 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$ in the case of ILC, in the presence of ground motion and facilities noise, a fast feedback is envisaged, operating at the interaction point, to correct the incoming position error of one beam with respect to the other, within the duration of the bunch train. Prototypes of such systems have been developed by the Feedback On Nanosecond Timescales (FONT) group. Initially, a purely analogue system was developed for room temperature RF cavity based linear collider designs, such as the NLC, and achieved a system latency of 23 ns, on a 56 ns duration bunch train at the KEK Accelerator Test Facility (ATF) in 2005 [3]. Following the choice of superconducting RF for the ILC, with ~ 1000 bunches separated by ~ 500 ns, a digital IP feedback system prototype was developed, using a custom FPGA-based digital feedback controller. This has

been tested and used extensively at ATF, and has been employed in the beam stabilisation efforts at the ungraded ATF extraction line, ATF2. The 'FONT5' feedback controller has also found applications in other beam stabilisation systems requiring low latency, for example, the CLIC drive beam phase feedforward demonstration at the CLIC Test Facility (CTF3). Details of the feedback controller, including data acquisition, will be presented as well an overview and key results from the applications above.

FONT5 FEEDBACK CONTROLLER

Figure 1 shows the PCB and front panel of the FONT5 digital feedback controller. The board is based around a Xilinx Virtex-5 FPGA (XC5VLX50T) [4], with a maximum speed of 550 MHz and 2160 Mb integrated block memory. The board has nine analogue input channels using 14-bit ADCs [5], capable of digitising up to 400 MSPS. Only the most significant 13-bits are connected to the FPGA however, in order to reduce routing congestion, and hence ease timing closure in the FPGA fabric. The ADCs have a low-latency (3.5 clock cycles) making them very suitable for fast feedback applications. The nine channels are arranged as three banks of three, with each bank sharing a common ADC clock. Offset DACs are provided to trim the ADC pedestals. The board also features four 14-bit DACs [6], with a maximum conversion speed of 210 MHz and 0.5 clock cycle latency. As for the ADCs, only the upper 13-bits are connected to the FPGA.

An on-board 40 MHz oscillator is provided for clocking slow logic and ancillary functions, as well as a fast comparator for an external system clock, usually in the range 200 to 400 MHz. A fast system clock can either be sourced externally, with an optional PLL-based jitter filter, or synthesised internally using a digital clock manager. Two programmable-level digital inputs are provided, which are normally used for trigger inputs, as well as several buffered and non-buffered I/Os. Communication to the FPGA is made via an RS-232 connection, running at up to 460.8 kbps. 128 7-bit control registers are used to communicate commands and variables to the FPGA, and up to 1024 samples per channel can be stored in Block RAM and transmitted via a UART, alongside read-backs of the control registers and status bytes. ADC data is displayed and saved to file using custom DAQ software written in LabVIEW. This software can also set control registers, and load RAM tables on the FPGA. Data and settings can be published as EPICS process variables, and infor-

* glenn.christian@physics.ox.ac.uk

mation from other systems can be incorporated via EPICS channel access.

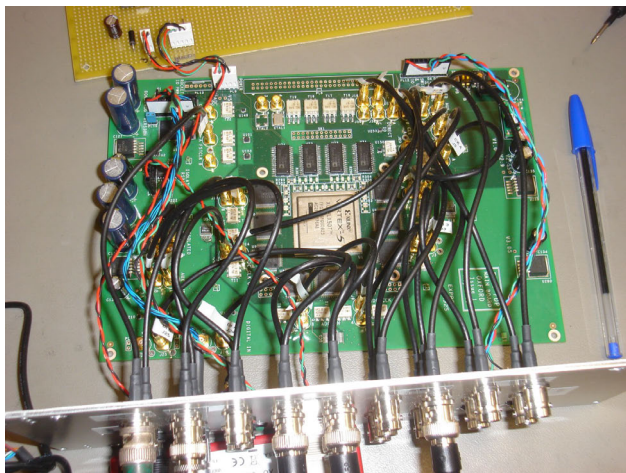


Figure 1: FONT5 digital feedback controller.

POSITION AND ANGLE STABILISATION AT ATF2

The ATF2 project at KEK, in Japan, provides an energy-scaled mock-up of the compact final-focus system for ILC, in the extraction line of the ATF. It aims to, firstly, demonstrate production of a 37 nm vertical spot size at the focus, and, secondly, demonstrate stabilisation of the beam spot at the nanometre scale. The FONT5 system is a two-phase vertical position correction system upstream in the extraction line, Fig. 2, designed to stabilise the beam to the micron level at the entrance to the final-focus. The system consists of three stripline beam position monitors (BPMs) on movers, P1, P2 and P3, and two stripline kickers, K1 and K2. The BPMs, P2 and P3, and kickers, K1 and K2, are approximately orthogonal in betatron phase, and each kicker is driven by a linear combination of both BPMs, such that the beam jitter is corrected at both phases. The other BPM, P1, is used as a diagnostic for the incoming beam jitter, and for measuring the resolution. High-sensitivity analogue front-end signal processors down-mix the high frequency BPM signals to below 100 MHz for digitisation. A measurement resolution of ~ 300 nm has been demonstrated, at a bunch charge of approximately 1 nC, for a linear range of $\pm 500 \mu\text{m}$ [7]. The use of BPM movers, with a range of ± 1.5 mm, ensures that the BPM measurement is not limited by changes in beam orbit in the extraction line.

The BPM processors produce three output signals; two quadrature phase sum signals, and one difference signal. The difference between opposing strips in the BPM being proportional to beam intensity and position offset, and the sum being proportional to intensity alone; a difference-over-sum algorithm is employed to remove the dependence on intensity. The difference output is guaranteed to be matched in time with the in-phase sum signal, and so a quadrature phase difference signal is not necessary. The quadrature

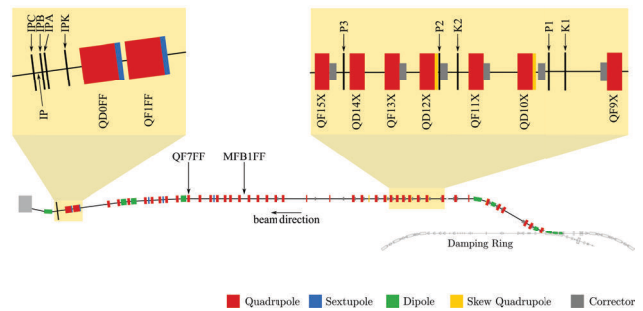


Figure 2: Layout [8] of the ATF2 extraction and final focus beamline with the FONT regions shown in detail.

phase sum signal is highly sensitive to timing jitter of the beam signal with respect to the phase of the local oscillator, which itself is sourced from the machine RF. Therefore, by characterising the degree of sensitivity for each BPM to the variation in phase, it is possible to remove offline any contribution to the apparent position jitter caused by phase jitter; for example, that originating from the synchrotron oscillation of the bunches. In practice, for use in the real-time feedback system, phase shifters placed between the BPM and the processor are used to remove any residual path difference between the processor inputs, and hence minimise the sensitivity to phase jitter. The drive signal is applied to the kickers using custom kicker drive amplifiers [9].

For each BPM, the three signals from the BPM processor are digitised by the ADCs on the FONT5 board, with a common ADC clock used for all signals from each BPM, as the signals from different BPMs will arrive asynchronously at the FONT5 board. The signals from the BPM processor have a width of around 5 ns, and so to reliably capture these, a fast ADC clock is used, derived from the machine RF, such that the sampling remains phase-locked to the beam. In practice, a frequency of 357 MHz is used, which corresponds to the sub-harmonic bunching frequency. The Virtex-5 IODELAY [10] elements are used to ensure that the signal waveforms are sampled at the peak of the in-phase sum signal. These provide a 64-tap delay line, with a tap resolution of 75 ps. The ADC clocks are provided via the FPGA, and a 'data ready' signal from the ADCs is used in a feedback to adjust the centre of the data-eye to the capture edge of the system clock. The output delays on the ADC clocks are then scanned to find the peak of the signal, with the data and 'data ready' input delays being modified in concert.

A timing signal from the extraction kicker is used as a trigger, around 1 ms before beam is extracted from the damping ring. As this signal is not guaranteed to be stable with respect to the bunching frequency, a secondary counter is used, the period of which corresponds to the revolution period of the damping ring. The primary counter is therefore used to select the correct cycle of the secondary counter, in which the bunches will occur; and the timing of the bunches is hence locked to the secondary counter. Data is acquired, and the feedback system is active, during this cycle of the secondary counter, which corresponds to 462 ns. The ATF can

extract up to three bunches with an ILC-like bunch spacing, with a ~ 310 ns duration kicker pulse.

The firmware for the feedback application consists primarily of charge normalisation, gain application, a ‘delay loop’ to provide memory of the correction signal for subsequent bunches, and FIR filtering, to account for droop in the output stages and amplifier. Gain and charge normalisation are applied via look-up tables implemented in Block RAM, which can be loaded in real-time from the LabVIEW DAQ software. All nine channels of ADC data, as well the kicker drive signals applied to the DACs, and the values of the control registers and status bytes, are read-out to the DAQ at 460.8 kbps, every 3.12 Hz. This data then is published as EPICS process variables via the National Instruments EPICS I/O Server [11].

Figure 3 shows the result of the feedback operation as measured at P2, P3 and at MFB1FF, a location with high vertical beta-function approximately 30 m downstream used to witness the correction. A minimum latency of approximately 140 ns has been demonstrated previously [12], and for these tests a beam consisting of two bunches separated by 182 ns was used. The feedback tests therefore involve measuring the vertical position of bunch one and correcting the vertical position of bunch two. The system was typically operated in an ‘interleaved’ mode, whereby the feedback correction was toggled on and off on alternate machine pulses; the feedback ‘off’ pulses thereby provide a continual ‘pedestal’ measure of the uncorrected beam position. The position jitter is reduced from $1.6 \mu\text{m}$ to 610 nm at P2, and from $1.8 \mu\text{m}$ to 520 nm at P3. This factor of ~ 3 reduction in jitter is successfully preserved out to MFB1FF, with the beam jitter being stabilised from $30 \mu\text{m}$ to below $10 \mu\text{m}$.

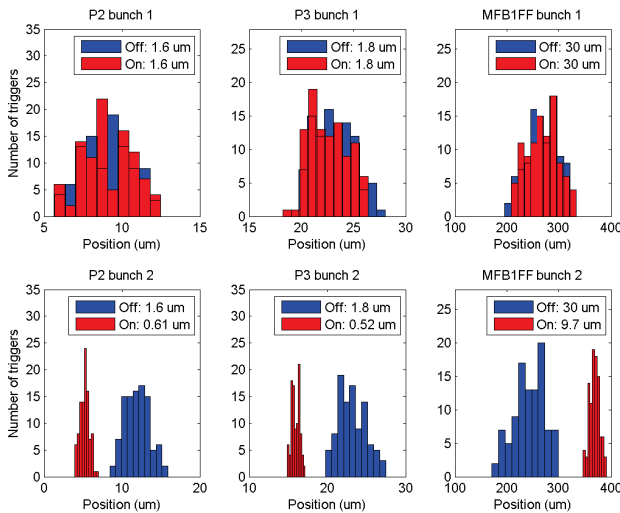


Figure 3: Distribution of the vertical position of bunches 1 and 2 in P2, P3 and MFB1FF with (red) and without (blue) application of the feedback correction. Values of the position jitter are quoted for each BPM.

DRIVE BEAM PHASE STABILISATION AT CTF3

The two-beam acceleration concept for CLIC [2], places strict requirements on the phase stability of the drive beam. To limit luminosity loss from emittance growth due to energy jitter to less than 1%, the phase of the drive beam needs to be stable to within 0.2 degrees (at 12 GHz), or ~ 50 fs, with respect to the main beam. To this end, it is envisaged to have a phase feedforward (PFF) system, at each of the drive beam decelerator sections along the CLIC linacs. For each system, the path length through a four-bend magnetic chicane is varied using fast electromagnetic kickers situated around the bending magnets. The phase offset is measured at the entrance to a turn-around loop, with the correction chicane at the exit of the loop. The system latency is therefore designed to be less than time it takes the beam to traverse the loop.

A prototype of such a system has been tested at CTF3, at CERN. This system uses three high precision 12 GHz RF cavity-based phase monitors and two stripline kickers [13,14], a high-power, high bandwidth amplifier system, and the FONT5 digital feedback controller. The system layout is shown in Fig. 4, where only part of the CTF3 facility is shown for clarity. Two phase monitors are located upstream of the TL1 transfer line into the CTF3 combiner ring; one of these is used as the phase input to the PFF system, the other to cross-check the monitor performance. The kickers are located downstream, prior to the first and last dipole in a four-bend dog-leg chicane in the TL2 transfer line between the combiner ring and the CLIC Experimental Area (CLEX). By varying the voltages applied to the two kickers the beam can be deflected onto longer or shorter paths through the chicane, hence correcting the phase. The third phase monitor is located downstream of the correction chicane, in CLEX, to witness the phase correction. For this demonstration, uncombined beam, bunched at 3 GHz, has been used, and the time-of-flight of the beam from the upstream monitor to the correction chicane, including half a turn of the combiner ring, is ~ 380 ns; thereby defining the latency constraint for the correction system. The demonstration aims to achieve 0.2 degrees phase stability, at a bandwidth above 30 MHz.

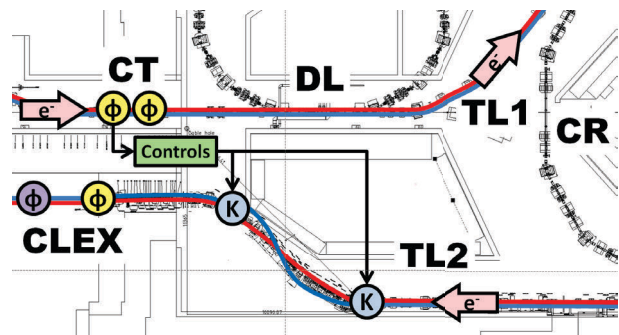


Figure 4: Simplified schematic of the PFF system. Red and blue lines depict orbits for bunches arriving late and early at the first phase monitor, ϕ , respectively. The trajectory through the TL2 chicane is changed using two kickers, K.

As well as minor modifications to the lattice to accommodate the correction kickers [15], new optics were required for TL2 to maximise the change in phase as function of applied kick (R52), whilst minimising the effect of energy jitter on phase (R56), and ensuring orbit closure after the chicane [16]. The phase monitor noise was measured by comparing the residuals between the two upstream monitors, which yielded a resolution at best of 0.14 degrees (at 12 GHz), with a typical performance around 0.2 degrees. The modular amplifier design, utilising SiC FETs, provides up to 20 kW output power, with 50 MHz bandwidth. This provides ± 700 V of drive to the kickers, which coupled with the optics design, gives ± 5.5 degrees of phase change in the chicane.

Custom application firmware was also written for the FONT5 boards to process the down-mixed phase monitor signals, as well as apply gain and offset control, in order to provide the correct magnitude of drive, and to centre the limited correction range with respect to the measured upstream phase. Programmable delay, using 32-tap shift registers, was also included to accurately match the timing of the correction signal to the arrival of the beam in TL2. The overall system latency is dominated by delay in the cables between the kicker amplifiers and kickers, which is constrained by the routing of available cable trays at around 175 ns. The FONT5 takes a minimum of 20 clock cycles (at 357 MHz) for the signal processing, with seven clock cycles of timing slack, taken up by the digital delays.

An example, illustrating the operation of the phase feed-forward system, is given in Fig. 5. The mean upstream and downstream phase measurements over 75 machine pulses are compared, both with and without operation of the PFF system. The section of the pulse which is correctable within the limits of the amplifier is shown by the black vertical lines; outside of these limits the amplifier is in saturation and produces a roughly constant phase offset with respect to the uncorrected phase. Within the time region marked by the black lines, the PFF system reduces the RMS downstream phase from 1.68 ± 0.02 degrees to 0.26 ± 0.01 degrees of 12 GHz.

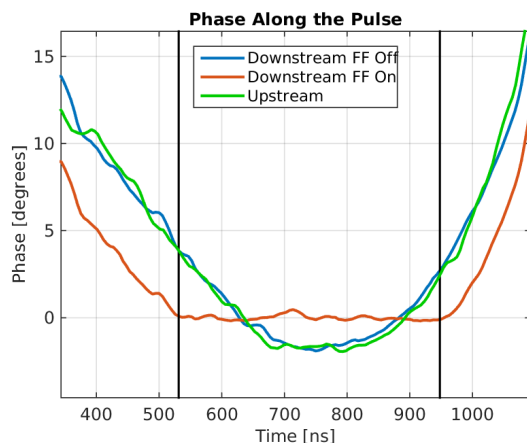


Figure 5: Phase variations along the pulse with the PFF system on and off.

ACKNOWLEDGMENTS

We are grateful for the help and support of our colleagues and collaborators at KEK-ATF, CERN-CTF3, INFN Frascati, IFIC Valencia, and the ATF2 Collaboration. Work supported by the European Commission under the FP7 Research Infrastructures project Eu-CARD, grant agreement no. 227579.

REFERENCES

- [1] C. Adolphsen *et al.*, *The ILC technical design report*, volume 3: Accelerator, JAI-2013-001, 2013.
- [2] CLIC Collaboration, “CLIC Conceptual Design Report”, CERN-2012-007.
- [3] P. N. Burrows *et al.*, “Performance of the FONT3 Fast Analogue Intra-train Beam-based Feedback System at ATF”, in *Proc. EPAC’06*, paper MOPLS123.
- [4] Xilinx Inc., Virtex-5 Family Overview (DS100), http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf
- [5] Texas Instruments Inc., ADS5474, <http://www.ti.com/product/ADS5474>
- [6] Analog Devices Inc., AD9744, <http://www.analog.com/media/en/technical-documentation/data-sheets/AD9744.pdf>
- [7] R. J. Apsimon *et al.*, *Phys. Rev. ST Accel. Beams*, vol. 18, p. 032803, 2015.
- [8] G. R. White *et al.*, *Phys. Rev. Lett.*, vol. 112, p. 034802, 2014.
- [9] TMD Technologies Ltd., www.tmd.co.uk
- [10] Xilinx Inc., *Virtex-5 FPGA User Guide (UG190)*, http://www.xilinx.com/support/documentation/user_guides/ug190.pdf
- [11] National Instruments, “Introduction to EPICS”, <http://www.ni.com/white-paper/14144/en/>
- [12] G. B. Christian *et al.*, “Latest Performance Results from the FONT5 Intra-train Position and Angle Feedback System at ATF2”, in *Proc. IPAC’11*, paper MOPO017.
- [13] F. Marcellini *et al.*, “The CLIC Drive Beam Phase Monitor”, in *Proc. IPAC’10*, paper WEPEB035.
- [14] A. Ghigo *et al.*, “Kicker and Monitor for CTF3 Phase Feed Forward”, in *Proc. IPAC’11*, paper TUPC007.
- [15] P. Skowronski *et al.*, “Design of Phase Feedforward System in CTF3 and Performance of Fast Beam Phase Monitors”, in *Proc. IPAC’13*, paper WEOBB203.
- [16] J. Roberts *et al.*, “Design, Hardware Tests and First Results From the CLIC Drive Beam Phase Feedforward Prototype at CTF3”, in *Proc. LINAC’14*, paper MOPP033.

HARMONY: A GENERIC FPGA BASED SOLUTION FOR FLEXIBLE FEEDBACK SYSTEMS

X. Serra-Gallifa[†], J. Avila-Abellan, M. Broseta, G. Cuni, D. Fernandez-Carreiras, O. Matilla, A. Ruz, ALBA-CELLS Synchrotron, Barcelona, Spain

Abstract

Feedback and complex acquisition systems usually need real-time interaction among instruments with micro-second's time response. These implementations are hard to achieve with processors but feasible using FPGAs. There are some cases, such as synchrotron beamlines, where high flexibility and continuous tuning are also required, but the implementation of multiple full-custom FPGA designs are extremely time-consuming. Harmony is a solution based in FPGA that offers, via high level programming, a unique framework with common time base, data acquisition, storage, real-time processing, data sharing and diagnostic services designed to implement flexible feedback systems. It is based in two interconnected buses: Self-Describing Bus, developed at CERN/GSI under OHWR license, that communicates with Control System; and Harmony Bus which creates a bus framework where different modules can share timestamped data capable of pre-programed events generation. The first version of Harmony is already successfully being used in Em# project which objective is the development of a performant four-channel electrometer.

ELECTROMETER'S DEVELOPMENTS AT ALBA

During the phase design of the beamlines of ALBA the large number of four channel electrometer needs lead Computing Division to the development of a medium performance electrometer oriented to diagnostic applications [1]. The simplicity of integration in the beamline Control System was one of its main goals. The project was a success with more than 40 units installed and a very good performance of the current amplifier. Inherent analog capabilities extended the use of the electrometer to more complex requirements. Soon it was agreed that a new development project to take advantage of the current amplifier performance was needed to improve different aspects [2, 3]. The main points focused on:

- Improving the signal to noise rate increasing the digital resolution up to 18 bits and the sampling rate.
- The possibility to perform current measurements under voltage biased conditions up to 1 kV and also to avoid ground loops.
- Capability of implementing real-time feedback systems.
- Minimizing the obsolescence problems by adopting a flexible and modular architecture based in standards.

HARMONY: A FLEXIBLE FEEDBACK SOLUTION

From the beginning of the project, it was clearly seen that the new electrometer (Em#) should be easily adaptable to future needs of beamlines experimental stations. However, there is always a trade-off between performance and flexibility. Since the Em# is mainly designed to fulfil the requirements of the Beamlines in a Synchrotron (although is applicable in many other environments) a detailed analysis of the latencies needed by the different "players" was carried out. It states that:

- Low current measurements (bellow μA) have inherently due to its high gain stages and parasitic capacitors maximum bandwidths of few kHz.
- Encoder or trigger readouts can reach frequencies of few MHz.
- Actuators and motor reaction times are higher than hundreds of microseconds.
- Data acquisition timestamped in the microsecond range enables the correlation of different instruments' data during the analysis.

Therefore the solution targeted feed-back systems at frequencies up to 10 kHz, and consequently the FPGA technology was chosen among different options including several microprocessors.

Actually, the performance of modern FPGAs is largely sufficient for the required feedback speeds, with clock frequencies allowing hundreds of clock cycles to process inputs and produce outputs. This enabled a FPGA architecture with independent blocks sharing data. It was also considered that the use of a bus for data sharing design would optimize the desired flexibility for multiple applications.

Development in Workgroup

In previous developments at ALBA, the electronics engineers developed the hardware, gateware (FPGA software) and firmware; once everything was tested it was delivered to software engineers who integrate the instrument in the control and data acquisition system and developed the user interface. In order to overcome this issue, this new project incorporated the contributions from the software engineers earlier in the schedule, reducing the number of late requirements to the hardware and working in parallel shortening the whole duration of the project.

This approach involved a clear definition of the project and its interfaces from beginning. A crucial help to

[†] xserra@cells.es

achieve this objective was the use of Self-Describing Bus (SDB) [4] available in Open Hardware (OHWR). SDB defines a protocol of communication that enumerates all the cores of the hardware in a tree-like structure, similar to a filesystem with folders (see Figure 1). The OHWR community has made available a number of tools of great value to this project such as wgen2 [5], which generates a memory map of the whole set of registers in C and VHDL from a text configuration file specification of the core. This tool defined a “de facto” interface for hardware and software and consequently fostered the development in parallel. The modifications appeared during the development of the project were orchestrated as formal reviewed improvements of this interface. Also the granularity of the project in well-defined cores accessible from high level software allows a segmentation of the functionality of the device and makes simpler the development and tests of the software.

Origin of Harmony

The flexibility given by the SDB had some disadvantages: the communication between the cores was not flexible and fast enough for the needs of the project. The intercommunication between cores using the SDB showed latencies way above the requirements and was not deterministic.

The Harmony Bus (HB) was the proposed solution for a fast interconnection between cores overcoming the aforementioned limitations. The HB will act as a second optional bus for fast data sharing between cores, following the structure of SDB.

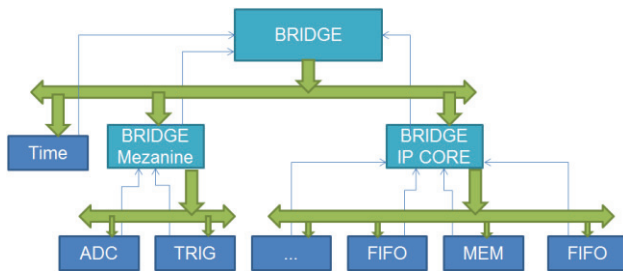


Figure 1: Data flow of the Harmony Bus physically implemented as a tree structure. The thin arrows represent the upstream bus and the green thick arrows represent the broadcast bus.

Harmony in Detail

One of the main components of HB is the set of Harmony Bridges, following the nomenclature of SDB. The cores are connected to bridges by HB, which act as bus arbiters. The cores send their data to a Harmony bridge, which gateway the frames to the top layer all the way up to the top Bridge (see Figure 1). Finally the top bridge broadcasts the data back to all components through the set of bridges, which register the broadcasted data to fit easily the timing constraints. The bridges add latency and jitter to the messages, but still the bus is fast enough (125 MHz) so these delays became negligible and the system is per-

ceived as an instantaneous pool where all components share their data among them (see Figure 2).

The HB is composed of 2 buses one for sending messages and a second where bridges broadcast all messages received. Both are 64-bit buses composed by 32 bits for data, 24 bits for timestamp in nanoseconds (TS) and 8 bits dedicated to data source identification (ID). There are also 3 lines for bus handshaking, one line indicating valid data in broadcast bus and 2 lines (data request and reading) to manage the upstream bus (see Figure 4).

The data transmitted by HB have an ID of 8 bits which allows up to 256 different sources. The IDs are assigned to each component by the control software via the SDB. There are two reserved message IDs: ID 0 is used to reset TS every 16 ms acting as a periodic time reference, and ID 255 is reserved for error messages.

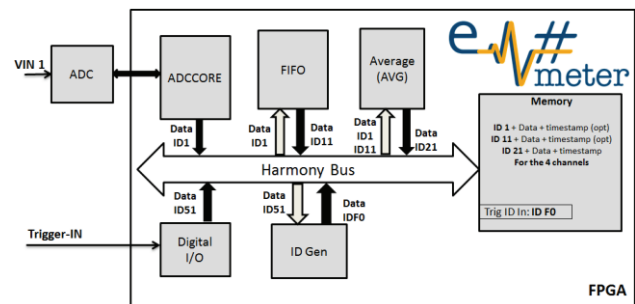


Figure 2: Block diagram of Em# Harmony implementation.

The Harmony bridges do a continuous diagnosis of the HB, measuring parameters like data sent or the maximum number of occupied slots. These parameters are accessible via SDB and allow the high level software, for example the control system to check the HB occupancy. Also each core has status flags used to ensure a correct communication process.

EM#. NEW HARDWARE MORE THAN AN ELECTROMETER

In addition to the SDB bus, other major decision was was the FPGA board (SPEC) developed at CERN in OHWR. The use of that FPGA board speeded up the development, as some critical issues were already tested for the OHWR community.

Em# included a Single board computer (SBC), running a Linux based operating system where the software for managing the configuration, data transfer and communication with the outside world is executed [6]. The SBC performs communications with the outside world via a 1 Gb Ethernet link, and also manages different diagnostics such as monitoring the temperatures and the consumption of different parts and communications using 1xPCIe port to FPGA.

The Em# also features custom electronics designed at ALBA such as a ADC FMC board, a Front Panel Board, a Current Amplifier Carrier Board, 4 Current Amplifiers Boards and a Power Supply Board. That hardware includes an 18-bit ADC isolated up to 1200 V, that reads

4 current amplifiers with 7 ranges from 1 mA to 100 pA, also isolated. Furthermore, the front panel has 4 analogic outputs, 4 digital In/Out Coaxial ports that support up to 100 MHz, and 9 In/Out differential ports (RS422) that can be used for synchronization or feedback implementations (Figure 3).

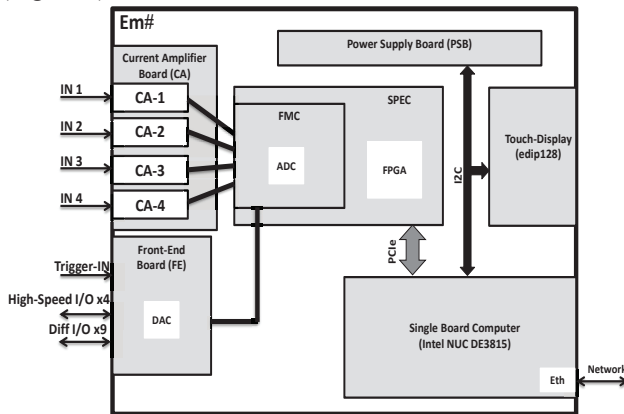


Figure 3: Em# hardware block diagram.

Harmony in Em#

The HDL cores already implemented in the FPGA are:

- ADCCORE: Component that controls the ADC device using high-speed serial lines.
- FIFO: used for storage or delays any data from a pre-specified ID. There is one module per channel.
- AVERAGE: used to calculate a dynamic average of data in 32-bit 2-Complement format. There is also one module per channel.
- ID GEN: generates data with and ID. It can be used to setup memories or for diagnostics
- Memory: Ram memory used to store data from fast BUS. It implements a circular buffer that can be started/stopped from fast or slow bus
- Digital I/O: Controls the digital input output ports. It can be used to implement a counter or an external trigger.

A typical example of an acquisition using the Harmony Bus is shown in Figure 2: the data acquired by the ADC in channel 1, is read by the ADCCORE block and sent to the Harmony bus with the identifier ID1, to be processed by the other modules, if necessary. FIFO uses the data with ID1 and generates a frame sent to the Harmony Bus with ID11 (containing data from the ADC delayed some messages). Average block sums data with ID1 and subtracts data ID11 from an accumulator register and generates data ID21. All the data sent to the Harmony Bus is available to be stored in memory block. Before starting the acquisition, at configuration time, the main software in the SBC setups how every component should behave in the HB. These communications between cores can be seen in the screenshot of Figure 4.

CURRENT STATE

After the first production tests, the first production of the Em# has already been launched. The project has enlarged the scope since one year ago when MaxIV and ALBA decided to create a international collaboration to further develop Em# architecture. A functional prototype is already working and 12 units are being produced by ALBA and 50 units by MAXIV. The Em# will be in both facilities the standard electrometer to be used for new installations. The first units are expected in mid-November.

From the technical point of view the future developments will be focus in the implementation of the management of the SPECS DDR3 memory and the DMA access from SBC to FPGA.

Moreover one the future challenges already considered in Em# architecture is to share the deterministic network provided by the Harmony among different Em# units to extend for example the feedback capabilities. One of the upcoming hardware developments for the platform is a multiple Input/Output module. Most of the components for such a module are already in place.

Finally a math processor for real-time calculus in the FPGA is also foreseen. This will increase the feedback capabilities of the Harmony devices.

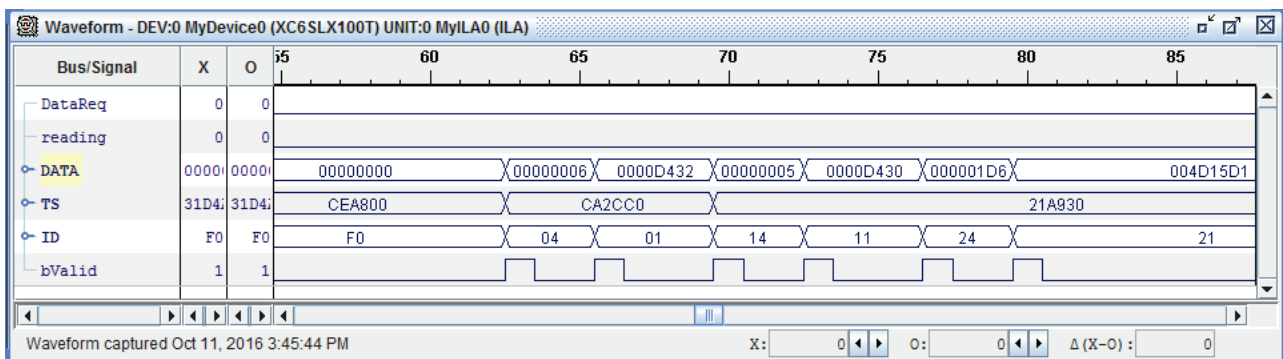


Figure 4: Data trace from ChipScope application, where it is shown the 3 control lines and the broadcast bus in a real Em# acquisition. ID0F is the trigger from the Digital IO block, ID04 and ID01 are the outputs of channels 1 and 4 of the ADC, ID14 and ID11 are the FIFO messages and ID21 and ID24 are the outputs of AVG Blocks.

CONCLUSION

The development of Harmony is finished and the first units are being manufactured. The design overcomes the problems identified in the first version and fulfils the requested project specifications being flexible, scalable and easy-to-upgrade. The modularity of the project naturally fosters several developers working on different parts of the project simultaneously. Few months ago, MAXIV and ALBA signed a collaboration agreement, also open to other institutes to join, to further develop the system in a continual improvement lifecycle.

Harmony will be the development framework for new instrumentation designed at ALBA, profiting of the flexibility of the platform, maximizing the possibility of interconnection of devices and sharing a common timebase for precise timestamps of data.

REFERENCES

- [1] J. Lidon-Simon *et al.*, “Low Current Measurements at ALBA”, ALBA-CELLS Synchrotron, in *Proc. ICALEPCS’11*, paper WEPMS025.
- [2] X. Serra-Gallifa *et al.*, “Em# Project. Improvement of Low Current Measurements at ALBA”, ALBA-CELLS Synchrotron, in *Proc. ICALEPCS’13*, paper TUPPC094.
- [3] O. Matilla *et al.*, “Em# Platform: Towards a Hardware Interface Standardization Scheme”, ALBA-CELLS Synchrotron, in *Proc. ICALEPCS’15*, paper WEPGF081.
- [4] A. Rubini, W. Terpstra, M. Vange, “Self-Describing Bus (SDB) – Specification for Logic cores – Version 1.1”, April 2013.
- [5] Open Hardware Repository, Wishbone Slave Generator, <http://www.ohwr.org/projects/wishbone-gen>
- [6] M. Broseta *et al.*, “Embedded Control System for Programmable Multi-Purpose Instruments”, ALBA-CELLS Synchrotron, presented at PCaPAC’16, Campinas, Brazil, Oct. 2016, paper THDAPLCO01, this conference.

CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY AT FRIB*

M. Konrad[†], D. Maxwell, G. Shen

Facility for Rare Isotope Beams, Michigan State University, East Lansing, MI 48824, USA

Abstract

Development of many software projects at the Facility for Rare Isotope Beams (FRIB) follows an agile development approach. An important part of this practice is to make new software versions available to users frequently to get feedback in a timely manner. Unfortunately building, testing, packaging, and deploying software can be a time consuming and error prone process. We will present the processes and tools we use at FRIB to standardize and automate this process. This includes use of a central code repository, a continuous integration server performing automatic builds and running automatic test, as well as automated software packaging. For each revision of the software in the code repository the continuous delivery pipeline automatically provides a software package that is ready to be released. The decision to deploy this new version of the software into our production environment is the only manual step remaining. The high degree of reproducibility as well as extensive automated tests allow us to release more frequently without jeopardizing the quality of our production systems.

INTRODUCTION

FRIB [1] is a project under cooperative agreement between US Department of Energy and Michigan State University (MSU). It is under construction on the campus of MSU and will be a new national user facility for nuclear physics. Its driver accelerator is designed to accelerate all stable ions to energies >200 MeV/u with beam power on the target up to 400 kW [2]. Commissioning of the front-end is currently underway and the remaining parts of the accelerator are planned to be commissioned over the next two years.

FRIB's controls group strives to support commissioning and operation by rolling out bug fixes and new features as fast as possible. To make this happen we are following principles of agile software development which include iterative, incremental and evolutionary development and a short feedback and adaption cycle. Unfortunately this approach can be slowed down considerably by the fact that building and deploying control-system software can be a complex and error prone process that often requires considerable manual work by experts. In the following we will describe how we speed up the build and deployment process for FRIB's controls software by following continuous integration (CI) and continuous delivery (CD) principles.

* Work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661

[†] konrad@frib.msu.edu

CONTINUOUS DELIVERY

The process of building and deploying software generally consists of a series of tasks that can be thought of as a pipeline. Figure 1 shows the steps of a typical software build and deployment process as it has been implemented at FRIB. Each task in this pipeline is carried out after the preceding step has been completed successfully. In the following we will describe each of these steps in detail.

Revision Control

All source code required to build software for FRIB's accelerator control system is stored on a central Git [3] repository server. The repository server is running Atlassian Bitbucket Server [4] which, in addition to basic Git server functionality, provides a web interface, pull requests and branch permissions.

Our Git work flow largely follows the Gitflow [5] approach which requires developers to implement new features or bug fixes on feature branches allowing them to work on their feature without the risk of breaking other developer's build. If however the number of feature branches becomes too high and feature branches live for too long merge conflicts are becoming more likely. To reduce time-consuming conflict resolution we are following the practice of CI which requires feature branches to be merged into a shared mainline frequently. CI principles generally recommend branches to be merged at least once a day. In our experience many controls projects have a rather low rate of change or a very low number of developers making a life time of a few days feasible with an acceptable risk of running into merge conflicts. For critical code we use pull requests as a tool to facilitate code reviews. The goal still remains the same: Code reviews and possibly required rework should be performed timely so that feature branches can be merged into mainline as fast as possible.

A significant amount of FRIB's control system software is developed in collaboration with other laboratories with its source code being tracked in an upstream repository on the Internet. In this case our CI server mirrors the mainline branch of the upstream repository into a branch in our local repository on a regular basis. Following CI principles, we are merging upstream changes into our own mainline branch as soon as possible to keep merge conflicts with our feature branches to a minimum. In general we are aiming to keep the number of FRIB-specific modifications to a minimum. Instead we prefer to contribute our improvements back to the upstream project. This reduces the risk of merge conflict in our repository and thus reduces the maintenance effort in the long run. At the same time this approach allows us to fix critical bugs in our local repository until they are fixed upstream.

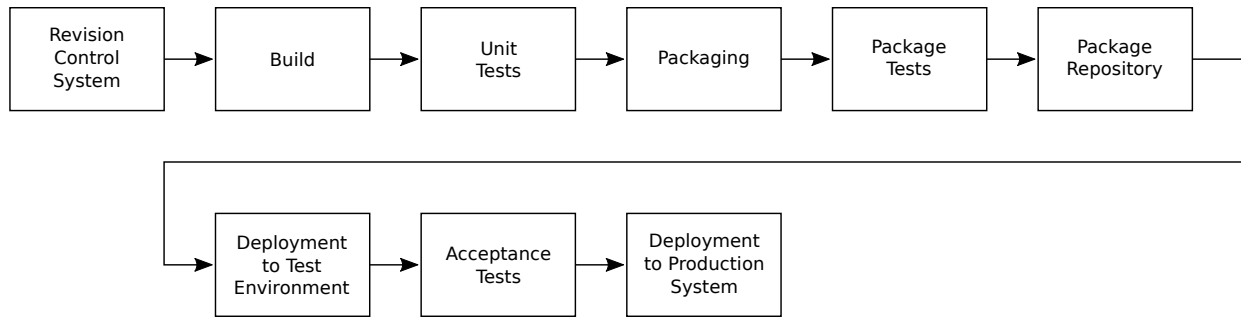


Figure 1: Overview of the FRIB continuous delivery pipeline.

Build

Building after each commit/merge is a best practice that ensures build issues are caught as early as possible. Frequent integration of feature branches into the mainline branch means the mainline code needs to be built frequently. At FRIB software is built automatically by a CI cluster running Jenkins [6]. This cluster consists of two Linux build nodes, a Windows build node, and a build node for FPGA projects. The two Linux build nodes can each run four jobs in parallel and are thus capable of completing builds within an acceptable time even at times of high load. A separate Jenkins master machine orchestrates the software builds by distributing them to the appropriate build nodes based on build requirements and load. All machines are virtual machines which facilitates adding resources or more build nodes.

Build logs are being archived for each build along with the artifacts that resulted from a successful build (e. g. binary files/package files). This data is available to developers via a web interface. For each successful build the resulting artifacts are passed on to the next step in the pipeline.

Unit Tests

Developers need to ensure their unit tests can be executed in a standard way (e. g. by running `make test`). Tests that have been set up in this way are automatically executed after the software has been built. Test results printed to standard output are archived by Jenkins as part of the build log. Test logs that are compliant to well known standards can also be visualized by Jenkins. For packages that do not come with unit tests this step is skipped automatically.

Packaging

FRIB is using Debian GNU/Linux as the standard operating system for control-system computers. Software is packaged into Debian packages to facilitate installation, upgrade and complete removal. Packages also allow developers to define dependencies between software components. Possible incompatibilities with specific versions of these dependencies can also be reflected in these dependencies.

Debian packages are built using Jenkins Debian Glue [7] which adds Debian packaging capabilities to Jenkins. Packages are generated in a two-step process. First a Debian source package is generated from the source code stored in the Git repository. In a second step this source package is built in a “clean-room” environment consisting of a chroot sandbox. In addition to a Debian base system only packages that are defined in the list of build dependencies for the package are installed in the sandbox guaranteeing a reproducible environment. Each build starts with a fresh sandbox. By using copy-on-write techniques Jenkins Debian Glue is able to speed up the process of generating build environments significantly.

Binary packages are built for each entry in a predefined build matrix. This matrix can be configured for each project individually and can contain multiple target architectures as well as multiple Debian operating system versions. Being able to build packages for multiple operating system versions is important during a phase of operating system upgrades where some machines might already run the new version while others are still using an older version.

Package Tests

A set of automated tests are run on each Debian package to catch common packaging issues. These tests ensure the package can be installed, upgraded and removed cleanly.

Package Repository

Once a Debian package has passed all tests it is pushed into a Debian package repository. This package repository is being managed by aptly [8]. It lives on the Jenkins master machine and makes packages available for download via HTTP. The package repository uses signatures based on state-of-the-art cryptography to make it more difficult for attackers to inject malicious packages into the controls network. Packages from this repository can also be installed by Jenkins to satisfy build dependencies of other packages.

Management of Jenkins Jobs

For most job types the build pipeline is broken into multiple jobs on the CI server to make it more obvious to devel-

opers in which step of the pipeline an error occurred. For Debian packages the pipeline consists of the following jobs:

1. Build source package
2. Build binary package(s)
3. Run package tests
4. Push package to the package repository

Note that the source package is only built once whereas the remaining steps are being performed for all entries of the build matrix.

So far more than 700 jobs are defined on the FRIB Jenkins server; the majority are related to building Debian packages. Instead of setting them up manually they are managed using Jenkins Job Builder [9]. This tool configures jobs based on a set of templates and a short description of each project. This ensures all jobs of a certain type are configured the same way.

In addition to this basic job management we automatically generate triggers between pipelines belonging to different projects based on the dependencies defined in the Debian packages. This ensures that after a library has been built all projects that use this library are being rebuilt automatically as well. This approach helps to catch issues as early as possible. The dependencies are visualized by Jenkins in form of a dependency graph.

AUTOMATIC DEPLOYMENT

FRIB uses Puppet [10] for IT configuration management. This tool takes a description of the configuration of the target computers as an input and ensures these machines are configured accordingly. For most machines deployment is completely automated allowing to re-install machines from scratch within a few minutes.

Deployment to Test Environment

At FRIB controls applications are tested in a development/test environment before they are deployed to the production network. This test environment mimics the FRIB production network as close as feasible (similar network layout, running the same services etc.). Configuration of machines on the test network is also managed by Puppet. Deployment to the test environment follows the principles of *continuous deployment* which means that the latest version of each application is deployed to this network automatically.

Acceptance Test

For most applications a manual acceptance test is performed in the test environment. If this tests passes the corresponding packages can be deployed to the production environment.

Deployment to Production Environment

Deployment to the production network follows the principles of *continuous delivery*. Each code version is being

built into a package which potentially can be released to the production environment. However, the release process itself requires a manual decision. This decision is based on the results of the acceptance test as well as on the operational needs of the accelerator. For example deployment can be postponed until the next maintenance shutdown.

SUMMARY

FRIB's build and deployment process has been automated successfully. This allows us to support commissioning and operation of FRIB by applying an agile development work flow with short adaption cycles. The automated processes have improved reproducibility significantly. Full traceability makes it easy to find out which version of the source code has been used to build a certain package. It also allows developers to inspect the corresponding build logs and intermediary build results making troubleshooting much easier. CD makes it easier for developers to contribute to many different projects since no expert knowledge is required to build and deploy the software. This facilitates team work.

The biggest challenge we encountered while introducing CD is a lack of test automation which can lead to a lack of developer confidence. In a similar way users can feel uncomfortable with software being upgraded frequently, especially during critical parts of accelerator operation. Until test coverage has been improved this can be overcome by careful scheduling of upgrades.

REFERENCES

- [1] FRIB, <http://www.frib.msu.edu>
- [2] J. Wei *et al.*, "FRIB Accelerator: Design and Construction Status," in *Proc. 13th Int. Conf. on Heavy Ion Accelerator Technology. (HIAT'15)*, Yokohama, Japan, September 2015, paper MOM1102, pp. 6–10.
- [3] Git Distributed Version Control System, <https://git-scm.com/>
- [4] Atlassian Bitbucket Server, <https://bitbucket.org/product/server>
- [5] V. Driessen: A successful Git branching model, <http://nvie.com/posts/a-successful-git-branching-model/>
- [6] Jenkins Automation Server, <https://jenkins.io/>
- [7] Jenkins Debian Glue, <http://jenkins-debian-glue.org/>
- [8] aptly Debian Repository Management Tool, <https://www.aptly.info>
- [9] Jenkins Job Builder, <http://docs.openstack.org/infra/jenkins-job-builder/>
- [10] Puppet Configuration Management Tool, <https://puppet.com/>

EXPERIENCE GAINED DURING THE COMMISSIONING OF THE UNDULATOR CONTROL SYSTEM AT THE EUROPEAN XFEL

S. Karabekyan[†], S. Abeghyan, J. Pflueger, M. Yakopov, European XFEL, Hamburg, Germany

Abstract

The European XFEL is a fourth-generation light source, which will start the operation in spring 2017. Three undulator systems - SASE 1, SASE 2 and SASE 3 - will be used to produce photon beams. For operation of all undulator systems, in total 91 undulators have been produced and commissioned. SASE 1 and SASE 3 undulator systems, consisting of total 56 undulator cells, have been installed and prepared for the operation in the tunnel in spring and summer 2016. SASE 2 will be installed by the end of 2016. This paper describes the commissioning process of the whole undulator control system and reports about the experience gained over the entire duration of undulator control system commissioning.

INTRODUCTION

Commissioning of the undulator control system was a multilayer task and was carried out in several steps. The strategy of commissioning was to test all hardware components at least once before those components would be installed in the tunnel. The same is true for the software development. Each release of software has been tested first using a simulation software and then on the undulator system test setup, before using it for the real undulator system.

COMMISSIONING OF THE HARDWARE

Undulator System Test Setup

An undulator system consists of an array of up to 35 undulator cells installed in a row in the tunnel along the electron beam. It consists of up to 35 undulator segments and intersections. The system is controlled by a central control node (CCN). It is installed in the control room, which is located about 1 km away from the undulator system. CCN communicates with the undulator cells over optical fibers, and the communication between individual cells is implemented using copper Ethernet and EtherCAT cables. Two media converter racks (MCR) installed from both sides of the system are used to convert signals from copper carriers to optical fiber carriers and vice versa. These two MCR racks are necessary for implementation of the redundant ring topology used for control of the undulator system.

It was obvious that all the envisaged components were to be tested before installation in the tunnel. For this purpose, an undulator system test setup was built in the undulator hall (see Fig. 1). The only difference between the real undulator system and test setup was the amount of undulator cells, which was reduced to four in the test setup. This test setup allowed to test the complete hard-

ware components before installation in a tunnel. It also allowed developing the global control system software three years ahead of the installation of the system in the tunnel.

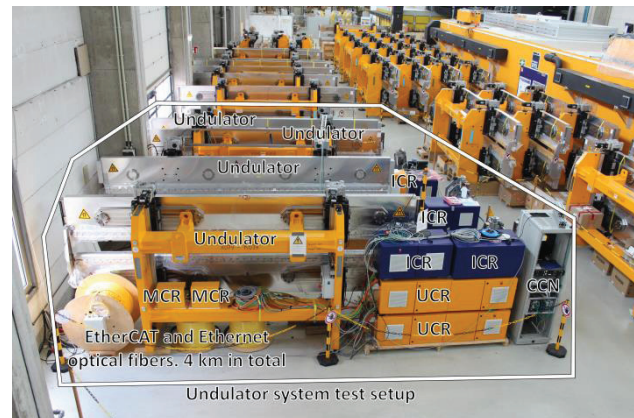


Figure 1: Undulator system test setup in the hall.

Control Components For Undulator Segments

The undulator control system needed to be commissioned before the magnetic commissioning of the undulator. This process is described in details in reference [1]. The undulator control hardware contains components installed on the undulator frame, as well as in the undulator control rack (UCR). The rack is connected to the undulator using a cable bundle. The first operation of the undulators and UCRs took place at the companies producing the undulator frame. It included only the operation of the motors, encoders and limit switches installed on the frame.

For the magnetic commissioning of the undulator it was necessary to bring it into the magnetic measurement hutch. After placing it in the hutch, the undulator had to be connected to the control rack installed on the rooftop of the hutch. Control of all undulators introduced to the hutch can be carried out using the same rack. Nevertheless, it was decided to commission the undulator with the assigned control rack. During this commissioning, a complete set of tests was carried out. This strategy allowed to check the whole undulator control system before installation in the tunnel and helped to discover of hardware-related problems in more relaxed situation on about 5% of the system.

Intersection Control Components

The intersection control components consist of the hardware installed on the Quadrupole Mover (QM) and Phase Shifter (PS) as well as Intersection Control Rack (ICR). The phase shifters have been produced by three different companies. For production and adjustment pur-

[†] suren.karabekyan@xfel.eu

poses, as well as for the factory acceptance tests (FAT), dedicated PS control cabinets were produced and delivered to these companies. In a similar way, companies producing the QMs were assisted. The ICRs were produced by one company and also needed to be tested before the delivery to European XFEL. Therefore, one PS and one QM were provided to the company for the FAT.

After delivery of all components to the European XFEL, site acceptance tests (SATs) were arranged. A separate, air conditioned hutch was prepared for the SATs. The hutch was equipped with a granite stone to fix the QM and PS. The tests with QMs and PSs were carried out in full compliance with technical specifications [2, 3]. If any of the components didn't match the specifications, this component was sent back to the manufacturer. Our experience has demonstrated that after the SAT only 1% of the delivered QMs and PSs needed to be returned. Approximately the same failure rate has been observed for ICRs.

SOFTWARE FOR COMMISSIONING OF AN UNDULATOR SYSTEM

A specific feature of undulator systems for free-electron lasers is the large number of recurring elements. Each undulator cell in the system at European XFEL is controlled by a local control node (LCN). The LCN is an industrial PC produced by Beckhoff Automation GmbH running a Programmable Logic Controller (PLC) implemented in the TwinCAT system. The TwinCAT runs under the Windows operating system. The software running on each LCN must be identical, although each cell component has its individual settings. The other aspect which should be taken into account is the need to have possibilities to update the version of the TwinCAT software as well as specific firmware. These arguments lead to the decision to develop software which will automate this process.

Setup, Configuration and Maintenance of the Undulator Control System

The Image Deployment Automation (IDA) software was developed in preparation of making the system operational [4]. The main objectives of IDA are following:

- Minimize the time used to set up, configure and maintain undulator systems;
- Interlink the configuration to the Undulator Systems Database (USD);
- Automate as many operations as possible;
- Provide possibility to create a master image and follow the version numbering of the images;
- Distribute the master image to the LCNs;
- Provide capability to quickly update Beckhoff TwinCAT, whenever an update is released
- Minimize presence in the tunnel;
- Eliminate errors arising from manual work.

IDA components are distributed between CCN, LCN and database (see Fig. 2). The DESY network and the

private LAN of the undulator system are separated from each other, but interlinked by the CCN. DB Proxy is establishing communication between the Undulator Systems Database and the Client, which is residing on the LCN. The Image Manager is issuing different commands to the Client. Preboot eXecution Environment (PXE) technology, implemented in the BIOS of the LCN, made it possible to develop IDA. A tiny Linux Distro, which is loaded on LCN through PXE, is intended to write or clone images. At the same time the CCN is serving as storage for LCN images. The Client application is performing the actual work of configuration.

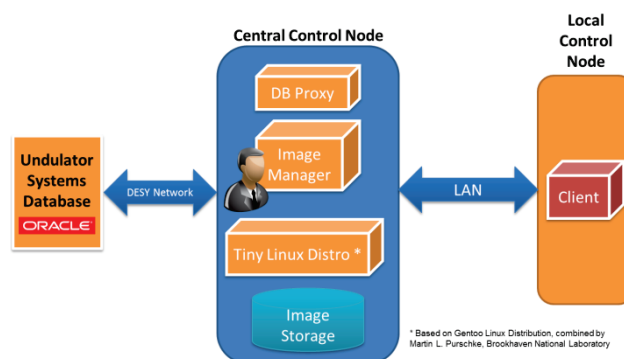


Figure 2: Schematic layout of the components of IDA.

One of the main functionalities of IDA is updating the images on the LCNs. The actual command is issued from the Image Manager application. The Image Manager is triggering an update command to LCN and rebooting it. As a result of the master boot record (MBR) handling, the LCN is not booting from compact flash, but starts to boot the operating system from the network using PXE. Afterwards, Tiny Linux distribution is booting on the LCN and running a dedicated script. The script, in this particular case, is reading an image from CCN Image Storage and writing it onto the LCN disk. Then LCN reboots into normal Windows. Lastly, the Client application starts running on the LCN, connecting to DB through the DB Proxy and configuring LCN according to its location. The main user interface after accomplishing the update of LCNs is illustrated in Fig. 3.

Cell	IP Address	Serial	MAC Address	Version	Status	Ping	Operation	Last Status
1	10.10.1.1	N/A	0001011A29D0	1.0	✓	1.19c7	Update	10/10/2016 9:25:52 AM
2	10.10.1.2	N/A	0001011A29D0	1.0	✓	1.19c7	Update	9/14/2016 12:50:23 PM
3	10.10.1.3	X083.K029	00010511EE3E	1.1.9c7	✓	1.19c7	Update	10/12/2016 3:08:23 PM
4	10.10.1.4	X082.K022	00010511EEA4	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:41:59 PM
5	10.10.1.5	X088.K006	000105130984	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:02 PM
6	10.10.1.6	X044.K004	0001050FE4B2	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:05 PM
7	10.10.1.7	X002.K002	0001060DEC9C	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:05 PM
8	10.10.1.8	X084.K024	00010511B344	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:08 PM
9	10.10.1.9	X088.K018	0001050DEC82	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:08 PM
10	10.10.1.10	X058.K010	0001050FE4A4	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:08 PM
11	10.10.1.11	X045.K005	0001050FE3B2	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:10 PM
12	10.10.1.12	X010.N008	00010511EE02	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:11 PM
13	10.10.1.13	X052.K012	0001050FE4A2	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:14 PM
14	10.10.1.14	X052.K001	0001050FE4AC	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:17 PM
15	10.10.1.15	X059.K019	00010511B44E	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:16 PM
16	10.10.1.16	X051.K011	0001050FE3B8	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:17 PM
17	10.10.1.17	X001.K001	0001050FE3B0	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:18 PM
18	10.10.1.18	X085.K025	00010511B45C	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:19 PM
19	10.10.1.19	X086.K026	00010511B46E	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:20 PM
20	10.10.1.20	X087.K027	0001050FE4B4	1.1.9c7	✓	1.19c7	Update	10/12/2016 3:36:24 PM
21	10.10.1.21	X060.K020	00010511EF28	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:24 PM
22	10.10.1.22	X061.K021	00010511EF36	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:24 PM
23	10.10.1.23	X045.K006	0001050FE49C	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:41 PM
24	10.10.1.24	X044.K009	00010511EE3E	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:49 PM
25	10.10.1.25	X071.K031	00010511EF30	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:55 PM
26	10.10.1.26	X053.K013	00010511EF2A	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:55 PM
27	10.10.1.27	X073.K033	0001050FE44E	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:42:59 PM
28	10.10.1.28	X063.K023	00010511EF18	1.1.9c7	✓	1.19c7	Update	10/6/2016 1:43:00 PM

Figure 3: Main user interface of the IDA.

After the assembly of the undulator system in the tunnel the procedure of putting the system into operation took two working days. The main problems encountered were related to the quality of the Ethernet connections between the LCNs. The performance of IDA for distribution of an image to LCNs, is strongly dependent on the network bandwidth. For an undulator system with 35 cells, it takes approximately 3 hours.

Undulator System Tester (UST)

The next step after putting the undulator system into operation is the commissioning of all control components that belong to it.

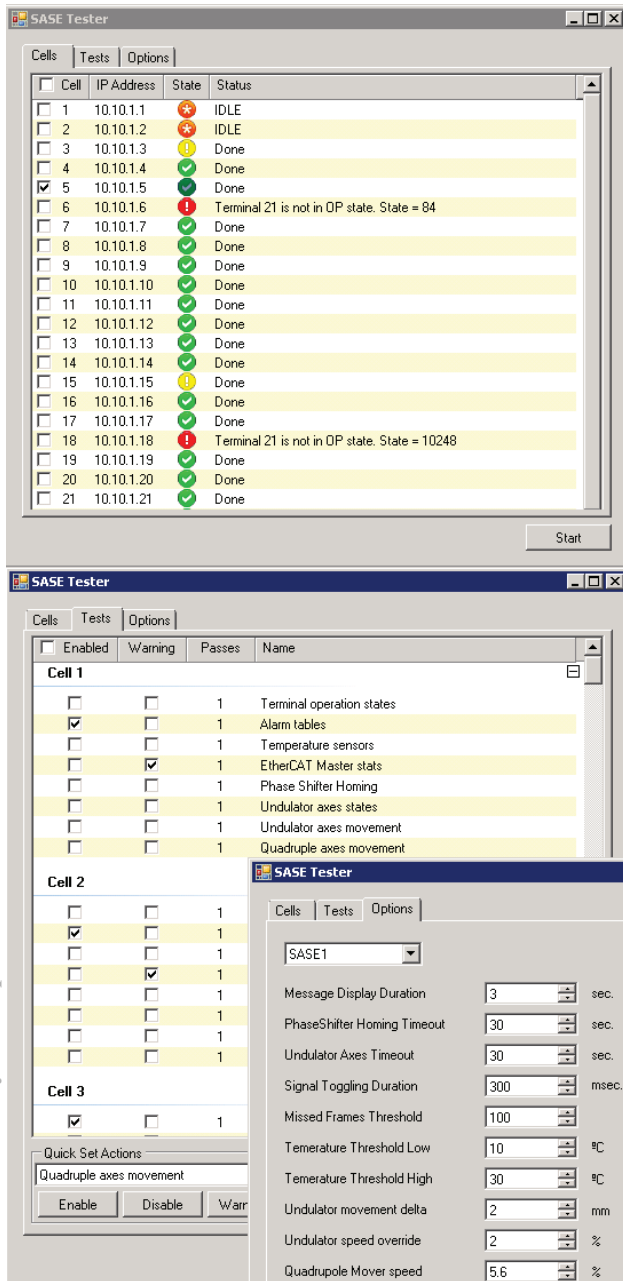


Figure 4: Main user interface of the UST as well as interfaces for configuration of the test scenarios and the setting values of the tests.

This task could be time consuming, taking into account, that several hundred values per system must be examined. This was a motivation to create a supervisory control and data acquisition (SCADA) program. This program is running on the CCN and is sending commands to the LCNs. After the execution of the commands, the program receives a feedback value. If the value is inside of an expected range, then the test is accepted; otherwise the test is marked as not successful. A double-click on the individual cell brings a popup log window. The error log provides precise information about the failure. The program provides a possibility to configure the test scenario for each individual cell. One can select what kind of tests must be carried out on which cell, how the feedback results must be interpreted, or in which case the test must be stopped. These are the settings of parameters, parameter ranges and timeouts for the test (see Fig. 4).

The executions of these tests are taking place simultaneously on each undulator cell. Depending on the test scenario, the complete test of the undulator system may take, only 10 to 30 minutes.

CONCLUSIONS

Our experience shows that after the assembly of the two undulator systems - SASE 1 and SASE 3 - in the tunnel, the full process of commissioning of the undulator control system takes approximately one week per system. This time was spent for fixing the problems with hardware used for the first time, in particular improving the quality of Ethernet and EtherCAT copper cables or fixing problems with wrongly swapped optical fibers.

The selected strategy - to test each piece of hardware before installation in the tunnel - fully justified itself, since the problems were solved during several years of hardware production and commissioning, in a more relaxed situation.

The software developed for setting up, configuration, commissioning and maintenance of the undulator control system allowed to reduce the final work in the tunnel to a couple of weeks instead of several months.

REFERENCES

- [1] S. Karabekyan *et al.*, "Use of Automation in Commissioning Process of the Undulators of the European X-Ray Free Electron Laser", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, p 64.
- [2] J. Munilla *et al.*, "Experience On Serial Production of the Quadrupole Movers with Submicrometric Repeatability for The European XFEL", in *Proc IPAC'15*, Richmond, VA, USA, May 2015, paper TUPWI015, p 2271.
- [3] P. Moreno-Torres *et al.*, "Design Process and Series Production of the Intersection Control Rack for the European XFEL Linear Accelerator", Power Electronics and Applications (EPE'15 ECCE-Europe), 2015 17th European Conference on, 8-10 Sept. 2015
- [4] S. Abeghyan *et al.*, "Image Deployment Automation", European XFEL, Schenefeld, Germany, Rep. WP71/2016/01, Jan. 2016.

List of Authors

Bold papercodes indicate primary authors

— A —	
Abbadì, A.A.	WEPOPRP010
Abbott, M.G.	THHWPLI001
Abeghyan, S.	THPOPRP014 , FRITPLC004
Abiven, Y.-M.	THHWPLI001 , FRFMPLI003
Allaria, E.	THPOPRP013
Althaus, A.	THPOPRP023
Aoyama, T.	THPOPRP009
Araujo, D.H.C.	WEPOPRP024
Araujo, W.R.	THPOPRP005
Avila-Abellan, J.A.	THDAPLC001 , FRFMPLC006

— B —	
Bacher, R.	WEUIPLI001
Balalykin, N.	THPOPRP016
Behrens, C.	THPOPRP013
Beniz, D.B.	WEPOPRP025 , THPOPRP011
Beutner, B.	THPOPRP013
Bickerton, R.C.	WEPOPRP026
Bisegni, C.	WECSPLC005
Bisou, J.	THHWPLI001
Blache, F.	FRFMPLI003
Blanch-Torné, S.	THDAPLC001
Blaskovic Kraljevic, N.	FRFMPLC005
Bobnar, J.	WECSPLC002 , WEPOPRP018
Bodenstein, R.M.	FRFMPLC005
Brinker, F.	THPOPRP022
Brito Neto, J.L.	THHWPLC004
Bromwich, T.	FRFMPLC005
Broseta, M.	THDAPLC001 , FRFMPLC006
Bruno, G.B.M.	THHWPLC004 , THPOPRP004
Burrows, P.	FRFMPLC005
Bylinskii, I.V.	THPOPRP027

— C —	
Caliari, R.M.	WEPOPRP020
Canova, H.F.	WEPOPRP023 , WEPOPRP024
Cardoso, F.H.	WEPOPRP023
Cascadan, A.M.	THDAPLC006
Caschera, S.	WECSPLC005
Castro Carballo, M.E.	THPOPRP013
Catani, L.	WECSPLC005
Chrin, J.T.M.	WEUIPLC004
Christian, G.B.	FRFMPLC005
Ciuffetti, P.	WECSPLC005
Claus, R.	THHWPLC002
Cobb, T.M.	THHWPLI001
Coelho, E.P.	THPOPRP003
Corruble, D.C.	FRFMPLI003
Cousins, A.M.	THHWPLI001
Cuní, G.	THDAPLC001 , FRFMPLC006

— D —	
D'Ewart, J.M.	THHWPLC002
D'Uffizi, A.	WECSPLC005
Dawiec, A.	FRFMPLI003
de Almeida, H.D.	WECSPLI001
de Oliveira, H.G.P.	WEPOPRP020
Decking, W.	THPOPRP013
Di Giulio, D.G.C.	WECSPLC005
Di Pirro, G.	WECSPLC005
Diop, M.	FRFMPLI003
Donadio, M.P.	TUTT3T02 , WECSPLI001 , WEPOPRP024
Duval, P.	WECSPLC002 , WEPOPRP018

— E —	
Ehrlichmann, H.	WEPOPRP018
Engblom, C.	FRFMPLI003
Errada, M.B.	THPOPRP005
Espindola, A.M.	WEPOPRP025

— F —	
Fatnani, P.	FRFMPLC004
Fernández-Carreiras, D.	THDAPLC001 , FRFMPLC006
Ferreira Filho, J.V.	TUTT3T01 , TUTT3T03 , TUTT3T04 , THDAPLC003
Ferreira, V.F.	THDAPLC006
Figueiredo, F.P.	WECSPLI001
Foggetta, L.G.	WECSPLC005
Franco, J.G.R.S.	THPOPRP003
Frisch, J.C.	THHWPLC002 , FRFMPLC002
Fröhlich, L.	THPOPRP013
Fujimaki, M.	WEPOPRP012
Fukunishi, N.	WEPOPRP011 , WEPOPRP012

— G —	
Galante, D.	THPOPRP005
Galletti, F.	WECSPLC005
Gao, F.	FRFMPLC001
Gargana, R.	WECSPLC005
Geraldes, R.R.	WEPOPRP024
Gioscio, E.	WECSPLC005

— H —	
Haller, G.	THHWPLC002
Hamad, A.	WEPOPRP010
Hamanaka, M.	WEPOPRP011
Hartmann, P.	THPOPRP023
Hensler, O.	THPOPRP013
Herbst, R.T.	THHWPLC002
Heron, M.T.	FRKTPLK01
Hierholzer, M.	WECSPLC003
Holub, B.	FRFMPLC001

Hong, B. THHWPLC002
Hubert, N. FRFMPLI003

— I —

Iitsuka, T. THPOPRP009
Ishii, H. WEPOPRP003

— J —

Jobert, N. FRFMPLI003
Jung, P. THPOPRP027

— K —

Kamikubota, N. THPOPRP009
Kammering, R. THPOPRP013
Karabekyan, S. THPOPRP014, FRITPLC004
Khan, S. THPOPRP023
Killenberg, M. WECSPLC003
Kimura, T. WEPOPRP004
Kobayashi, Y. WECSPLC004
Komiyama, M. WEPOPRP011, WEPOPRP012
Konrad, M.G. THPOPRP010, FRITPLC001
Kosuge, T. WEPOPRP003
Kozak, T. WECSPLC003
Kubsky, S.K. FRFMPLI003

— L —

Langlois, F. FRFMPLI003
Lee, J. THPOPRP027
Legat, U. THHWPLC002
Lescano, S. THPOPRP003
Limberg, T. THPOPRP013
Lomperski, M. WECSPLC002, WEPOPRP018

— M —

Ma, L. THHWPLC002
Marchand, P. FRFMPLI003
Marques, C. FRFMPLC001
Marques, S.R. THHWPLC004
Martins, L.A. THHWPLC004
Matilla, O. THDAPLC001, FRFMPLC006
Maxwell, D.G. FRITPLC001
Meykopff, S.M. THPOPRP013
Michelotti, A. WECSPLC005
Milas, N. WEPOPRP022
Minashkin, V.F. THPOPRP016
Modic, R. FRITPLC002
Moldes, J. THDAPLC001
Morton, A.C. THPOPRP027
Mukai, A.H.C. WEPOPRP021

— N —

N., N. THDAPLC005
Nagatani, Y. WEPOPRP003
Nakamura, T. WEPOPRP011

Nallin, P.H. THPOPRP003
Nemoto, H. THPOPRP009
Nitani, H. WEPOPRP003
Norum, W.E. THHWPLI003
Nozdrin, M.A. THPOPRP016

— O —

Okumura, R. WECSPLC004
Olsen, J.J. THHWPLC002

— P —

Paiva, T. THDAPLC006
Patel, H.P. THDAPLC005
Patel, J. THDAPLC005
Patel, P.J. THDAPLC005
Paulino, G. THPOPRP005
Penco, G. THPOPRP013
Pepler, D.A. WEPOPRP026
Perry, C. FRFMPLC005
Pflüger, J. THPOPRP014, FRITPLC004
Pinton, G.C. THPOPRP003
Piton, J.R. WECSPLI001
Planche, T. THPOPRP027
Purohit, D. THDAPLC005

— R —

Ramalho, L. TUTT2T01, TUTT2T02, TUTT2T03, TUTT2T04
Ramalho, L.A. THDAPLC006
Ramjiawan, R.L. FRFMPLC005
Rana, R. FRFMPLC004
Reese, B.A. THHWPLC002
Renaud, G. THHWPLI001, FRFMPLI003
Resende, X.R. WEPOPRP021, WEPOPRP022
Rivetta, C.H. FRFMPLC002
Roberts, J. FRFMPLC005
Rodrigues, A.R.D. THPOPRP003
Rodrigues, G.L.M.P. WEPOPRP020
Rodriguez, M. THDAPLC001
Rohde, D. THPOPRP023
Rollins, J.G. WEKTPLK01
Rose, J. FRFMPLC001
Rubio-Manrique, S. THDAPLC001
Ruckman, R. THHWPLC002
Russo, L.M. THHWPLC004, THDAPLC003
Ruz, A. FRFMPLC006

— S —

Sahoo, G.K. THPOPRP022
Saito, Y. WEPOPRP003
Salabert, J. THDAPLC001
Saleh, I. WEPOPRP010
Sant'Anna, F.C. THHWPLC004
Sapozhnikov, L. THHWPLC002
Sato, K.C. THPOPRP009

Sato, N.	WECSPLC004	— U —	
Schirmer, D.	THPOPRP023	Uchiyama, A.	WEPOPRP011, WEPOPRP012
Scholz, M.	THPOPRP013	Uzun, I.S.	THHWPLI001
Schünemann, G.	THPOPRP023		
Semissatto, G.T.	WEPOPRP023	— V —	
Serra-Gallifa, X.	THDAPLC001, FRFMPLC006	Varghese, G.	WECSPLC003
Serrano, J.	THKTPLK01	Vásquez, J.A.	THHWPLC002
Shehzad, N.	WECSPLC003	Vaz, M.	THDAPLC006
Shen, G.	FRITPLC001	Vilela, L.N.P.	WEPOPRP021, WEPOPRP022
Shinoda, A.A.	THDAPLC006	Viti, M.	WECSPLC003
Shirkov, G.	THPOPRP016		
Shukla, B.K.	THDAPLC005	— W —	
Silva, H.A.	THHWPLC004, THPOPRP004	Wang, G.M.	FRFMPLC001
Sivolella, A.	TUTT2T01, TUTT2T02, TUTT2T03 , TUTT2T04	Weaver, M.	THHWPLC002
Smith, S.R.	THHWPLC002, FRFMPLC002	Wedtstein, F.	THPOPRP022
Souza, J.	WEPOPRP023	Weis, T.	THPOPRP023
Stankevic, T.	FRFMPLI003	Weisse, S.	THPOPRP016
Stecchi, A.	WECSPLC005	Wilgen, J.	THPOPRP013
Steinhagen, R.J.	TUTT1T01, TUTT1T02, TUTT1T03 , TUTT1T04	Williams, E.	THHWPLC002
Stevani, I.	WEPOPRP021, WEPOPRP022	— X —	
Straumann, T.	THHWPLC002	Xu, C.	THHWPLC002
Szczesny, J.	THPOPRP013		
		— Y —	
— T —		Yadav, R.P.	FRFMPLC004
Ta, F.	THHWPLI001	Yakopov, M.	THPOPRP014 , FRITPLC004
Takamiya, K.	WECSPLC004	Yamada, S.	THPOPRP009
Tanigaki, M.	WECSPLC004	Yamamoto, N.	THPOPRP009
Tateyama, T.M.	THPOPRP027	Yoshida, S.Y.	THPOPRP009
Tavares, D.O.	THHWPLC004	Yoshinaga, H.	WECSPLC004
Tho, C.H.	WEPOPRP020	Yoshino, H.	WECSPLC004
Tikhomolov, E.	THPOPRP027	Young, A.	THHWPLC002
Towalski, P.	THPOPRP023		
Towne, N.A.	FRFMPLC001	— Z —	
Tylee, A.J.	WEPOPRP026	Zhao, Z.	THPOPRP014

Institutes List

ACMOS INC.

Tokai-mura, Ibaraki, Japan

- Nemoto, H.

ALBA-CELLS Synchrotron

Cerdanyola del Vallès, Spain

- Avila-Abellan, J.A.
- Blanch-Torné, S.
- Broseta, M.
- Cuní, G.
- Fernández-Carreiras, D.
- Matilla, O.
- Moldes, J.
- Rodríguez, M.
- Rubio-Manrique, S.
- Ruz, A.
- Salabert, J.
- Serra-Gallifa, X.

BNL

Upton, Long Island, New York, USA

- Gao, F.
- Holub, B.
- Marques, C.
- Rose, J.
- Towne, N.A.
- Wang, G.M.

CALTECH

Pasadena, California, USA

- Rollins, J.G.

CERN

Geneva, Switzerland

- Roberts, J.
- Serrano, J.

Cosylab

Ljubljana, Slovenia

- Bobnar, J.
- Modic, R.

DELTA

Dortmund, Germany

- Althaus, A.
- Hartmann, P.
- Khan, S.
- Rohde, D.
- Schirmer, D.
- Schünemann, G.
- Towalski, P.
- Weis, T.

DESY Zeuthen

Zeuthen, Germany

- Weisse, S.

DESY

Hamburg, Germany

- Bacher, R.
- Behrens, C.
- Beutner, B.
- Brinker, F.
- Castro Carballo, M.E.
- Decking, W.
- Duval, P.
- Ehrlichmann, H.
- Fröhlich, L.
- Hensler, O.
- Hierholzer, M.
- Kammering, R.
- Killenberg, M.
- Kozak, T.
- Limberg, T.
- Lomperski, M.
- Meykopff, S.M.
- Sahoo, G.K.
- Scholz, M.
- Shehzad, N.
- Szczesny, J.
- Varghese, G.
- Viti, M.
- Wedtstein, F.
- Wilgen, J.

DLS

Oxfordshire, United Kingdom

- Abbott, M.G.
- Cobb, T.M.
- Cousins, A.M.
- Heron, M.T.
- Uzun, I.S.

Elettra-Sincrotrone Trieste S.C.p.A.

Basovizza, Italy

- Allaria, E.
- Penco, G.

FRIB

East Lansing, USA

- Konrad, M.G.
- Maxwell, D.G.
- Morton, A.C.
- Shen, G.

GSI

Darmstadt, Germany

- Steinhagen, R.J.

INFN - Roma Tor Vergata

Roma, Italy

- Di Giulio, D.G.C.

INFN-Roma II

Roma, Italy

- Catani, L.

INFN/LNF

Frascati (Roma), Italy

- Bisegni, C.
- Caschera, S.
- Ciuffetti, P.
- D'Uffizi, A.
- Di Pirro, G.
- Foggetta, L.G.
- Galletti, F.
- Gargana, R.
- Gioscio, E.
- Michelotti, A.
- Stecchi, A.

INSA Lyon

Villeurbanne, France

- Lescano, S.

Institute for Plasma Research

Bhat, Gandhinagar, India

- N., N.
- Patel, H.P.
- Patel, J.
- Patel, P.J.
- Purohit, D.
- Shukla, B.K.

J-PARC, KEK & JAEA

Ibaraki-ken, Japan

- Sato, K.C.
- Yamada, S.
- Yamamoto, N.

JAI

Oxford, United Kingdom

- Blaskovic Kraljevic, N.
- Bodenstein, R.M.
- Bromwich, T.
- Burrows, P.
- Christian, G.B.
- Perry, C.
- Ramjiawan, R.L.
- Roberts, J.

JINR

Dubna, Moscow Region, Russia

- Balalykin, N.
- Minashkin, V.F.

- Nozdrin, M.A.

- Shirkov, G.

Kanto Information Service (KIS), Accelerator Group

Ibaraki, Japan

- Aoyama, T.
- Iitsuka, T.
- Yoshida, S.Y.

KEK

Ibaraki, Japan

- Ishii, H.
- Kamikubota, N.
- Kimura, T.
- Kosuge, T.
- Nagatani, Y.
- Nitani, H.
- Saito, Y.

Kyoto University, Research Reactor Institute

Osaka, Japan

- Kobayashi, Y.
- Okumura, R.
- Sato, N.
- Takamiya, K.
- Tanigaki, M.
- Yoshinaga, H.
- Yoshino, H.

LBNL

Berkeley, California, USA

- Norum, W.E.

LNLS

Campinas, Brazil

- Araujo, D.H.C.
- Araujo, W.R.
- Beniz, D.B.
- Brito Neto, J.L.
- Bruno, G.B.M.
- Caliari, R.M.
- Canova, H.F.
- Cardoso, F.H.
- Coelho, E.P.
- de Almeida, H.D.
- de Oliveira, H.G.P.
- Donadio, M.P.
- Errada, M.B.
- Espindola, A.M.
- Ferreira Filho, J.V.
- Figueiredo, F.P.
- Franco, J.G.R.S.
- Galante, D.
- Geraldies, R.R.
- Marques, S.R.
- Martins, L.A.
- Milas, N.
- Mukai, A.H.C.
- Nallin, P.H.

- Pinton, G.C.
- Piton, J.R.
- Resende, X.R.
- Rodrigues, A.R.D.
- Rodrigues, G.L.M.P.
- Russo, L.M.
- Sant'Anna, F.C.
- Semissatto, G.T.
- Silva, H.A.
- Souza, J.
- Stevani, I.
- Tavares, D.O.
- Tho, C.H.
- Vilela, L.N.P.

MAX IV Laboratory, Lund University

Lund, Sweden

- Stankevic, T.

NCC UNESP

São Paulo, Brazil

- Cascadan, A.M.
- Ferreira, V.F.
- Paiva, T.
- Ramalho, L.A.
- Shinoda, A.A.
- Vaz, M.

Oxford University, Physics Department

Oxford, Oxon, United Kingdom

- Burrows, P.
- Perry, C.

PSI

Villigen PSI, Switzerland

- Chrin, J.T.M.

RIKEN Nishina Center

Wako, Japan

- Fujimaki, M.
- Fukunishi, N.
- Komiyama, M.
- Uchiyama, A.

RRCAT

Indore (M.P.), India

- Fatnani, P.
- Rana, R.
- Yadav, R.P.

SESAME

Allan, Jordan

- Abbadi, A.A.
- Hamad, A.
- Saleh, I.

SHI Accelerator Service Ltd.

Tokyo, Japan

- Hamanaka, M.
- Nakamura, T.

SLAC

Menlo Park, California, USA

- Claus, R.
- D'Ewart, J.M.
- Frisch, J.C.
- Haller, G.
- Herbst, R.T.
- Hong, B.
- Legat, U.
- Ma, L.
- Olsen, J.J.
- Reese, B.A.
- Rivetta, C.H.
- Ruckman, R.
- Sapozhnikov, L.
- Smith, S.R.
- Straumann, T.
- Vásquez, J.A.
- Weaver, M.
- Williams, E.
- Xu, C.
- Young, A.

SOLEIL

Gif-sur-Yvette, France

- Abiven, Y.-M.
- Bisou, J.
- Blache, F.
- Corruble, D.C.
- Dawiec, A.
- Diop, M.
- Engblom, C.
- Hubert, N.
- Jobert, N.
- Kubsky, S.K.
- Langlois, F.
- Marchand, P.
- Renaud, G.
- Ta, F.

STFC/RAL

Chilton, Didcot, Oxon, United Kingdom

- Bickerton, R.C.
- Pepler, D.A.
- Tylee, A.J.

ThoughtWorks Brasil

Porto Alegre, RS, Brazil

- Ramalho, L.
- Sivoilella, A.

TRIUMF

Vancouver, Canada

- Bylinskii, I.V.
- Lee, J.
- Planche, T.
- Tateyama, T.M.
- Tikhomolov, E.

UNICAMP

Campinas, São Paulo, Brazil

- Paulino, G.

USTC/NSRL

Hefei, Anhui, People's Republic of China

- Zhao, Z.

UW/Physics

Waterloo, Ontario, Canada

- Jung, P.

XFEL. EU

Hamburg, Germany

- Abeghyan, S.
- Karabekyan, S.
- Pflüger, J.
- Yakopov, M.

Participants List

Anas Ahmad Abbadi
SESAME
Jordan

Débora de Paiva Magalhães
LNLS/CNPEM
Brazil

Gajendra Kumar Sahoo
DESY
Germany

Akito Uchiyama
RIKEN
Japan

Douglas Bezerra Beniz
LNLS/CNPEM
Brazil

Glenn Brian Christian
University of Oxford
United Kingdom

Andraz Pozar
Australian Synchrotron
Australia

Eduardo Pereira Coelho
LNLS/CNPEM
Brazil

Guifre Cuni Recepción
ALBA
Spain

André Muller Cascadan
São Paulo State University-UNESP
Brazil

Evgeniy Tikhomolov
TRIUMF
Canada

Guilherme Paulino
LNLS/CNPEM
Brazil

André Pontes Barbosa Lima
LNLS/CNPEM
Brazil

Felipe Koji Godinho Hoshino
LNLS/CNPEM
Brazil

Guilherme Teixeira Semissatto
LNLS/CNPEM
Brazil

Andrei Guinancio de Carvalho Pereira
LNLS/CNPEM
Brazil

Fernando Cambauva Sant Anna
LNLS/CNPEM
Brazil

Gustavo Barbosa Monteiro Bruno
LNLS/CNPEM
Brazil

Armand Hattar
Axon' Cable
Brazil

Fernando H. Cardoso
LNLS/CNPEM
Brazil

Gustavo Ciotto Pinton
LNLS/CNPEM
Brazil

Carlos Marques
BNL
USA

Franco Carbonera
Axon' Cable
Brazil

Gustavo Henrique Ornaghi Aranha
LNLS/CNPEM
Brazil

Christer Engblom
Soleil Synchrotron
France

Frank O'Dowd
LNLS/CNPEM
Brazil

Gustavo Lorencini Martins Pereira Rodrigues
LNLS/CNPEM
Brazil

Daniel de Oliveira Tavares
LNLS/CNPEM
Brazil

Gabriel Barros Zanoni Lopes Moreno
LNLS/CNPEM
Brazil

Heitor dos Santos Sousa
LNLS/CNPEM
Brazil

Dave Pepler
STFC
United Kingdom

Gabriel de Souza Fedel
LNLS/CNPEM
Brazil

Henrique Aires Silva
LNLS/CNPEM
Brazil

David Fernández
ALBA
Spain

Gabriel Oehlmeyer Brunheira
LNLS/CNPEM
Brazil

Henrique Alves Filho
LNLS/CNPEM
Brazil

Henrique de Oliveira Caiafa Duarte
LNLS/CNPEM
Brazil

Henrique Ferreira Canova
LNLS/CNPEM
Brazil

Joseph Carl Finlay
Canadian Light Source Inc
Canada

Marcelo Alexandre Leite de Moraes
LNLS/CNPEM
Brazil

Henrique Geraissate Paranhos de Oliveira
LNLS/CNPEM
Brazil

Júnior Cintra Maurício
LNLS/CNPEM
Brazil

Márcio Paduan Donadio
LNLS/CNPEM
Brazil

Isa Uzun
Diamond Light Source
United Kingdom

Laís Oliveira de Souza
LNLS/CNPEM
Brazil

Marcos Roberto Bissiano Errada
LNLS/CNPEM
Brazil

Isabella Stevani
LNLS/CNPEM
Brazil

Laís Pessine do Carmo
LNLS/CNPEM
Brazil

Mário Vaz da Silva Filho
UNESP
Brazil

James Rezende Piton
LNLS/CNPEM
Brazil

Lars Fröhlich
DESY
Germany

Mark Heron
Diamond Light Source
UK

Jameson Graef Rollins
LIGO
USA

Luana Nayara Pires Vilela
CNPEM/LNLS
Brazil

Martin Hierholzer
DESY
Germany

Jan Chrin
Paul Scherrer Institute
Switzerland

Lucas Alves Martins
LNLS/CNPEM
Brazil

Martin Konrad
Facility for Rare Isotope Beams
USA

Jan Marjanovic
CAEN ELS
France

Lucas Maziero Russo
LNLS/CNPEM
Brazil

Matthias Clausen
DESY
Germany

Janito Vaqueiro Ferreira Filho
LNLS/CNPEM
Brazil

Lucas Sanfelici
LNLS/CNPEM
Brazil

Mikhail Nozdrin
JINR
Russian Federation

Javier Serrano
CERN
Switzerland

Lucas Volpe
LNLS/CNPEM
Brazil

Mikhail Yakopov
European XFEL GmbH
Germany

Jeferson de Souza
LNLS/CNPEM
Brazil

Luciano Catani
Istituto Nazionale di Fisica Nucleare
Italy

Minoru Tanigaki
Kyoto University
Japan

João Leandro de Brito Neto
LNLS/CNPEM
Brazil

Luciene Moreira Marceul
Axon' Cable
Brazil

Mirko Milas
LNLS/CNPEM
Brazil

João Nilton Henrique da Rosa
LNLS/CNPEM
Brazil

Manuel Broseta Sebastia
ALBA
Spain

Misaki Komiyama
RIKEN Nishina Center
Japan

Norihiko Kamikubota
J-PARC/KEK
Japan

Ricieri Akihito Pessinatti Ohashi
LNLS/CNPEM
Brazil

Takuro Kimura
KEK
Japan

Paloma Cristina Jackson
LNLS/CNPEM
Brazil

Robert Modic
Cosylab
Slovenia

Thomas Walter
DESY
Germany

Patrícia Henriques Nallin
LNLS/CNPEM
Brazil

Robson Barbieri
Siemens
Brasil

Till Straumann
SLAC
USA

Paul Louis Hernandez
Axon' Cable
Brazil

Rodolfo Picoreti
UFES
Brazil

Torben Worm
Aarhus Universitet-ISA
Denmark

Peter Hartmann
TU Dortmund University
Germany

Ross Hogan
Australian Synchrotron
Australia

Volker RW Schaa
GSI
Germany

Philip Duval
DESY
Germany

Sergio Rodrigo Marques
LNLS/CNPEM
Brazil

William Roberto de Araujo
LNLS/CNPEM
Brazil

Ralph J. Steinhagen
GSI
Germany

Simin Chen
Australian Synchrotron
Australia

Xavier Serra-Gallifa
ALBA
Spain

Reinhard Bacher
DESY
Germany

Suren Karabekyan
European XFEL GmbH
Germany

Yves-Marie Abiven
Synchrotron Soleil
France

Ricardo Caliri
LNLS/CNPEM
Brazil

Takashi Kosuge
KEK
Japan

