FESA - THE FRONT-END SOFTWARE ARCHITECTURE AT FAIR

T. Hoffmann, GSI, Darmstadt, Germany

Abstract

The planned Facility for Antiproton and Ion Research (FAIR) at GSI in Darmstadt is a very challenging task due to its dimension and complexity. Several new heavy ion accelerators have to be built and then operated in parallel and multiplexed modes. In order to cope with these unique requirements numerous collaboration partners are involved to add so-called ”in-kind contributions” to the project. Detailed guidelines and interface specifications have to be defined in advance to avoid an indefinite pool of different technologies which have to be handled by the future control system. For that purpose, GSI decided to use the Front-end Software Architecture (FESA) at the lowest level of the control system. FESA was developed by CERN and is already established for usage at LHC and its injectors. It is a framework to integrate any kind of equipment such as beam instrumentation devices, magnet power supplies, vacuum- and cryogenic components into the control system. A framework overview, its advantages, and boundary conditions provided by FESA are described.

FAIR

The Facility for Antiproton and Ion Research FAIR will be built at GSI in Darmstadt. The FAIR project embraces nine new accelerator-installations such as a p-linac, two synchrotrons, four storage rings, the Super-Fragment Separator and FLAIR (see Fig. 1). The injectors for these new accelerators are going to be the existing machines UNILAC and SIS18. A detailed description of FAIR can be found in [1] and [2]. Like in the past for GSI, also the new FAIR facility will be operated in a multiplexed mode, which means that several different experiments will be supplied with several different ion types simultaneously. Due to this requirement, FAIR is going to be one of the most complex accelerator facilities worldwide.

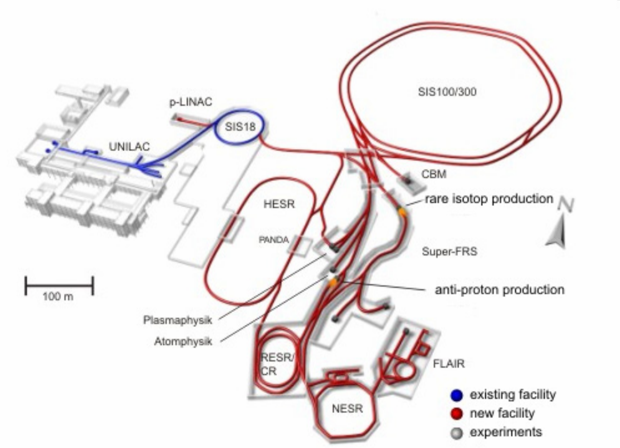


Figure 1: Layout of the FAIR facility including the existing accelerators UNILAC and SIS18.

An important consequence for the accelerator division is the installation of a new control system. The existing control system (CS) at GSI is well adapted for the present needs, but because of technological requirements such as scalability, data acquisition performance and outdated hardware the need of a new control system is obvious. Later on, an upgrade of the existing CS to the new one is envisaged.

DEMANDS ON A NEW CONTROL SYSTEM

Because of the extensive amount of requirements on a new CS such as industrial controls, field bus systems, timing etc., they can not be treated in this contribution in detail. Exemplarily, the implementation of beam instrumentation and diagnostics will be discussed.

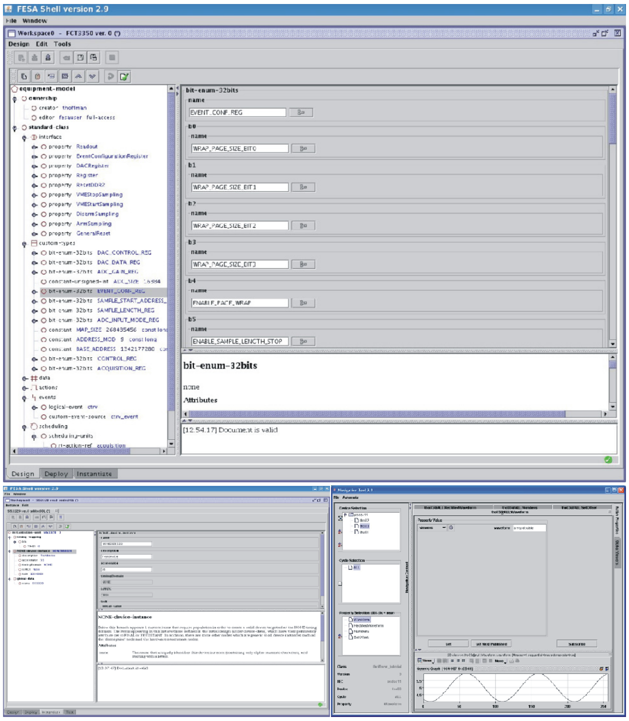
Approximately thousand diagnostic devices such as viewing screens, profile grids, beam loss monitors and many more have to be controlled and read out. Therefore the new CS should be optimized not only for controlling functionality but also for data acquisition performance. At this point, the high data flux and real time acquisition has to be taken into account. In addition, the correlation of locally distributed signals as a function of time must be possible, which requires a sophisticated timing system. To serve many different users with measured data in parallel, a powerful middleware, as a mediator in between, is mandatory. To implement such a large amount of different devices, including also hardware from external partners, which are called "in-kind contributions", a standardized framework is required. It is the aim to avoid frequent costs due to software licence fees, therefore "Open Source" is preferred. The CS should be based on the highly accepted operating system Linux, widespread programming languages such as C/C++, Java and should support most common hardware technologies, such as VME, cPCI/PXI and PCI/PCIe. The most critical and sophisticated demand is the acquisition, control, and handling of data generated by parallel beam operation.

SITUATION

In the course of the strict boundary conditions concerning personnel, time schedule and costs, as given by the large-scale multi-national FAIR project, the strategy to implement new technologies has to be most effective. Besides, the technical groups at GSI have to assure the proper operation of the existing accelerators, thus additionally limiting capacities for R&D work. To fulfil the FAIR demands it is obvious, that external help is required and that expert skills of non-CS staff such as from the beam diagnostics department (BD), have to be integrated to compensate the lack of manpower. Furthermore the start-up of CS-collaboration with a partner institute seems to be a promising approach. A powerful partner is CERN with its newly developed CS for LHC. CERN and its CS is close to the FAIR requirements producing heavy ion beams in a multiplexed operation. CERN had to build a new CS for LHC and already started a software upgrade program for the injectors, such as the linacs and the proton synchrotron PS. This situation is similar as it will be for FAIR. One important part of the CERN CS which has been evaluated for usage at FAIR is the front-end and data acquisition architecture. This part is called *F*ront-*E*nd *S*oftware *A*rchitecture (FESA) and was developed for LHC and its injectors in a partnership between the CERN controls and the beam instrumentation group. The evaluation of FESA by GSI was started in October 2006 and ended in March 2008 leading to the decision to adapt FESA for FAIR.

WHAT IS FESA

The real-time front-end software architecture FESA is a framework used to fully integrate equipment such as power supplies, PLCs or beam diagnostic devices at the front-end level into the CS. The output product of this framework is a so-called equipment class. FESA provides JAVA based graphical user interfaces (GUI) [Fig. 2] to design, deploy, instantiate and test the equipment classes.


FESA Shell version 2.9
File Window
Workspace0 - FCT3350 ver. 0 (*)
Design Edit Tools
equipment-model
ownership
creator thoffman
editor fesauser full-access
standard-class
interface
property Readout
property EventConfigurationRegister
property DACRegister
property Register
property ResetDDR2
property VMEStopSampling
property VMEStartSampling
property DisarmSampling
property ArmSampling
property GeneralReset
custom-types
bit-enum-32bits DAC_CONTROL_REG
bit-enum-32bits DAC_DATA_REG
bit-enum-32bits ADC_GAIN_REG
constant-unsigned-int ADC_SIZE 16384
bit-enum-32bits EVENT_CONF_REG
bit-enum-32bits SAMPLE_START_ADDRESS_
bit-enum-32bits SAMPLE_LENGTH_REG
bit-enum-32bits ADC_INPUT_MODE_REG
constant MAP_SIZE 268435456 const long
constant ADDRESS_MOD 9 const long
constant BASE_ADDRESS 1342177280
bit-enum-32bits CONTROL_REG
bit-enum-32bits ACQUISITION_REG
data
actions
events
logical-event ctrv
custom-event-source ctrv_event
scheduling
scheduling-units
rt-action-ref acquisition
bit-enum-32bits
name
EVENT_CONF_REG
b0
name
WRAP_PAGE_SIZE_BIT0
b1
name
WRAP_PAGE_SIZE_BIT1
b2
name
WRAP_PAGE_SIZE_BIT2
b3
name
WRAP_PAGE_SIZE_BIT3
b4
name
ENABLE_PAGE_WRAP
b5
name
ENABLE_SAMPLE_LENGTH_STOP
bit-enum-32bits
none
Attributes
Design
Deploy
Instantiate

[12.54.17] Document is valid

Figure 2: Example screenshots of some FESA GUIs made in Java. Top: Design-Tool. Bottom-Left: Instantiation-Tool. Bottom-Right: Navigator-Tool.

The XML based output of the design phase is used to generate C++ source code via XSLT processing. This code has to be adapted to the device-specific readout and control methods based on non-generic drivers and libraries. The following compilation leads to a platform dependent executable file. This has to be associated with the corresponding front-end CPU (VME, PXI, PCI) by using the Deployment-Tool. Then, this executable file can either be started on reboot of the front-end system or manually. It handles the whole real-time data acquisition and communication with the middleware. The proper functioning of the DAQ can be tested with the integrated Navigator-Tool. A basic approach to FESA, especially from the user's point of view, is described in [3].

ADVANTAGES OF FESA FOR FAIR

Basically, there is no must for the usage of FESA at FAIR from a technical point of view. A home-made or other different control- and data acquisition system could perform similarly. The main reason to choose FESA is the high level of maturity and availability right now to set the boundary conditions and to start education and usage with the existing staff and the in-kind contributors.

FESA is not only a control but also data acquisition system and therefore allows the integration of demanding applications such as video imaging or high resolution bunch to bunch beam position monitoring.

The framework addresses also non-CS-department device experts such as BD members to integrate complex devices into the CS. So, partially, the lack of manpower can be compensated.

The FESA-user does not need any specific knowledge about sophisticated programming on interfaces, timing, middleware access, visualisation etc. as this source code is generated automatically. This approach of re-usage of debugged and tested code saves plenty of developing time and prevents errors and inconsistencies.

The FESA community with about 70 active developers is growing. That leads to a possible exchange of experience, developments, tips and tricks. Large synergy effects are expected here.

The decision for FESA leads to a high standardization of the supported environment. It is intended to prevent a confusing pool of used hard- and software, in order to maximize the maintainability of the DAQ.

Changes to the design or code by the user, by others, and also ten years later, are much easier thanks to the uniform source code structure.

FESA includes some safety features such as role based design and test access, instantaneous syntax checks and CVS storage.

Since the FESA project was started approximately five years ago, the framework has passed several releases for debugging and enhancements. This leads to a stable and tested and almost complete framework. The actual FESA version is 2.10 and includes beneath many other features timing-simulation, RT priority settings, run-time diagnostics and monitoring tools [4].

CONSEQUENCES

It is obvious, that a complete FESA environment installation at GSI is a challenging task. There are fundamental dependencies between FESA and the CERN environment, which have to be resolved for GSI. An example is the CERN timing system. Within FESA the scheduling of all actions is designed by using the CERN timing events and domains. The existing GSI timing is very different to the CERN timing and can not be used directly with FESA. It is planned to develop an FPGA based converter as a first workaround. Finally, a new timing system for FAIR is foreseen, which is subject of ongoing studies of the CS department at GSI.

To establish FESA and to create a mandatory FESA competence centre at GSI, a FESA expert core team has to be set up. This requires full understanding, training and adaption of FESA to the GSI accelerators. Collaboration between the CERN and GSI CS groups to transfer the expertise is already established. Critical technical issues such as the operation of the 50 Hz universal linear accelerator UNILAC have to be handled in the future.

TEST INSTALLATIONS

Preliminary preparations and tests such as the real-time behaviour were carried out to investigate the principle and technical usability of FESA at GSI. Hence the latency time of a FESA real-time action (RT) was investigated by taking the time between an external RT-trigger and the change of an LED state on a VME module, after the RT action has written data to a memory register.

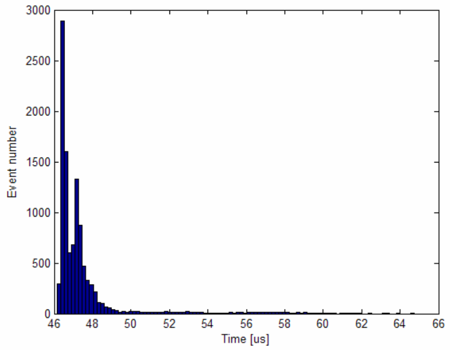


Figure 3: The latency time of a FESA RT action is well distributed below 50µs on a VME RIO3 800 MHz controller.

The measurement was performed on a VME RIO3 800 MHz front-end CPU. The binary file was started with a priority setting of 50 in LynxOS. The result of approx. 50µs latency as shown in Fig. 3 is sufficient for all momentarily known purposes except the real-time handling of the UNILAC. This special task will be handled by FPGA-based CPUs supervised by overlaid FESA controllers.

Furthermore the complete vertical hierarchy from the sensor and front-end system up to the GUI in the main control room has to be tested to learn and demonstrate the possible controls architecture for FAIR. Therefore, collaboration between BD and the CS was set up to install a complete beam position monitoring (BPM) system for the SIS 18. Due to personal and time critical issues the commercial control system company Cosylab [5] was additionally integrated into this task. The concept of this DAQ is shown in Fig. 4. As the analogue input 12 Libera Hadron digitizers [6] with 4x125MSamples/s are used.

Aim is to develop adequate FESA classes for all parts of the system such as the data acquisition, programmable timing, amplification gain settings etc. plus connection to the middleware (CMW) via remote device access (RDA) and to Java based GUIs for control and orbit visualisation. The challenging part is the handling of the high data rate of approx. 10 GBit/s from all stations. First results of this FESA test project are expected for Spring 2009.

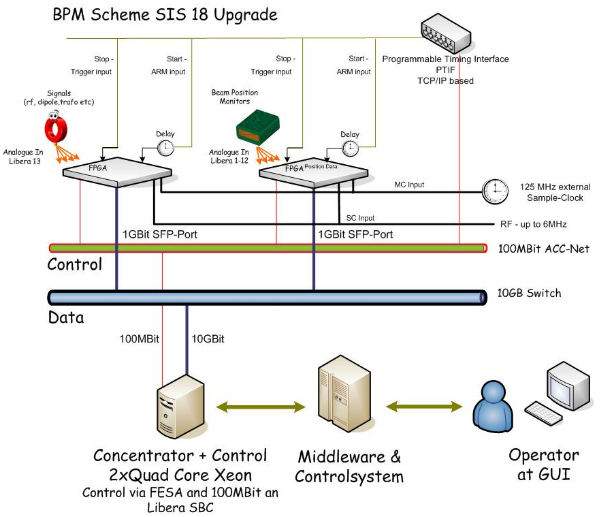


Figure 4: Scheme of the planned SIS18 BPM system to test and demonstrate the usability of FESA.

OUTLOOK

The decision for using FESA as the front-end system for FAIR is a promising way to go. It adds the data acquisition part to the control system and increases cooperation between beam diagnostic and control system departments. To be successful with this installation the assistance of CERN is essential. The author would like to thank all members of the CERN control system group for all the work done so far within the frame of this collaboration.

REFERENCES

[1] P. Spiller, "FAIR at GSI", HB2006, Tsukuba, Japan, MOBP02, p. 24-28.
[2] R. Baer et al., "The new FAIR accelerator complex at GSI: project, controls challenges and first steps", ICALEPCS 2007, Knoxville, Tennessee, USA, TOAB01, p. 59 – 61.
[3] T. Hoffmann, "FESA Primer – A simple and fast approach from a user's point of view", 3/2007 Fesa Version 2.9, t.hoffmann@gsi.de
[4] M. Arruat et al., "Front-End Software Architecture", ICALEPCS 2007, Knoxville, Tennessee, USA, WOPA04, p. 310-312.
[5] www.cosylab.si
[6] www.i-tech.si