CONSOLIDATING THE FLASH LLRF SYSTEM USING DOOCS STANDARD SERVER AND THE FLASH DAQ

O. Hensler, W. Koprek, H. Schlarb, V. Ayvazyan, C. Schmidt, DESY, Hamburg, Germany
Q. Geng, SLAC, Menlo Park, CA, U.S.A.

Abstract

Over the last years the LLRF group developed many different flavors of hardware to control the RF systems at the Free Electron Laser in Hamburg (FLASH). This led to a variety of firmware versions as well as control system programs and display panels.
A joined attempt of the LLRF and the controls group was made over the last year to consolidate hardware, improve the firmware and develop one DOOCS front-end server for all 6 RF stations. Furthermore, DOOCS standard server are used for automation, like simple state machines, and the FLASH DAQ for bunch-to-bunch monitoring tasks, e.g. quench-detection.

An outlook of new developments for the upcoming European XFEL, using xTCA technologies, will be given.

INTRODUCTION

Over the last 15 years FLASH has evolved from a small test facility with a gun and one 8 cavity-accelerator module, running at about 100 MeV, to a photon science user facility. After the last shutdown in 2009/10 FLASH has been upgraded to 7 accelerator modules with eight 1.3 GHz cavities each, plus a 3rd harmonic module with four 3.9 GHz cavities. This set-up allows FLASH to run at a maximum beam energy of about 1.2 GeV. Presently, six RF stations are required to supply the gun, the 3rd harmonic- and the seven 1.3 GHz modules with RF.

Over this long period, the controls for the Low-Level RF (LLRF) evolved alongside the modifications of the accelerator. Many different flavours of LLRF controller hardware, starting from a pure analogue-based system for the first gun, a successfully used DSP[1] system for the modules and different versions of Simcon and SimconDSP[2] systems were developed. All these systems came with dedicated firmware, device server software and operator display panels, leading to a very inhomogeneous, global control system. Such a system was hard to maintain and applying global automation procedures was very difficult, because of the different structure and naming convention of every device server.

The effort to consolidate the LLRF system during the last shutdown will be described.

DOOCS

The Distributed Object Oriented Control System DOOCS[2] is the leading system for the FLASH accelerator. DOOCS is a standard client/server control system and based on an object-oriented approach at the front-end/server and client/display side. It is mainly implemented in C++, but there is now a Java client-side implementation called jDOOCS, on which the new display tool jDDD[3] is based. An interface for MATLAB clients is provided. The communication protocol is based on ONC Remote Procedure Calls (RPC), but a strong effort is on the way to replace them by the TINE[2] protocol.

HARDWARE

In order to achieve a homogeneous LLRF system, it is very important to start at the hardware level already. It was decided to use only two types of SimconDSP VME boards, which are equipped with ten 14 bit ADCs. One type has a Virtex V50 FPGA from Xilinx installed, which is suitable to run all control algorithms needed and is used as master card. If only additional analogue I/O is required, a SimconDSP board, equipped with a Virtex V40 is used as a slave card. The two boards are interconnected via 1 Gb fibre link to exchange the real-time data.[4]

FIRMWARE

After coming up with a common hardware platform, only a few different version of the FPGA firmware are needed, which have many parts in common, like the VME interface structure. The VME part has been optimized to allow the new 10 Hz operation of FLASH. A mapping file is provided for all VME register and tables allowing to change the firmware independent from the device server. The following firmware versions are needed :

- RF gun: This version is special, because the RF gun has no hardware probe signal. This has to be calculated from the forward and reflected power signals. In addition, the gun is a normal conducting cavity, which requires different control algorithms.
- Master board: This version includes all LLRF control and regulation algorithm as well as beam based feedbacks.
- Slave board: A simplified version to readout the ADCs and calculate the partial vector-sum is needed.

LLRF CONTROLLER SERVER

The LLRF controller server is the interface between the SimconDSP board and the control system and programmed using the DOOCS tool kit. The server runs

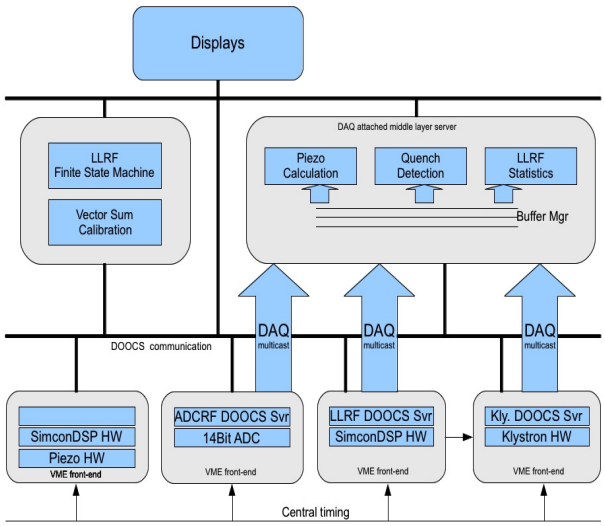

Figure 1: This picture shows the overall concept for one RF station with all required front-end computer and middle-layer server using DAQ and standard communication

on a local SPARC CPU inside a VME crate. As operating system Sun Solaris is being used. The CPU board has a hard-disk attached to store the server binary, shared libraries, configuration files, archiving and the FPGA firmware, providing complete network independent operation.

The main design goal is to have just one device server binary, which should be configurable to the needs of the individual RF stations. This was achieved by programming individual classes e.g. for cavity read-out, vector-sum, main control or board set-up. These classes are activated during start-up of the DOOCS server, while reading the server configuration file, the locations are created with its individual properties. Complex control algorithms are separated in a C library, which allows to use these algorithms in several projects and is provided by the LLRF team. All classes recover its values into the firmware after a power-up or firmware reload.

The LLRF controller server uses the FLASH nomenclature now, e.g. one location name per cavity. This eases the correlation by date with other server information and simplifies the design of operator panels. The following classes are implemented :

Board class

This class loads the FPGA firmware, in case the system is powered up or a new firmware version should be loaded. By monitoring a firmware counter, the overall operation of the firmware is monitored. In case of a failure, the RF is switched off via the FLASH timing system.

Main Class

This class is the central part of the LLRF controller server. Most of the controls, like amplitude or phase set-point, is done here. All control tables for the firmware are generated in this class and downloaded to hardware.

Due to thermal effects during startup of the accelerator, it is required to change the output rotation matrix in feed-forward mode in order to speed up. During feedback operation, these values are drifting back, though they need to be adjusted slowly. Beam based feedbacks are closely connected with the LLRF regulation and handled in this class as well.

A similar version of this class is used for the RF gun.

Cavity Class

This class reads the I and Q values of one cavity probe for one macro bunch and calculates amplitude and phase from it. A calibration parameter for each cavity is stored in this location.

Vector-Sum Class

The vector-sum class is similar to the cavity class, but is reading the partial or total vector sum of the system. The total output rotation matrix is calibrated here.

Learning Feed-Forward Class

This class monitors the error signal of the LLRF system, which is the difference between set-point and the driving output. In case this error signal gets too big, the learning feed-forward (LFF) algorithm tries to compensate by calculating new feed-forward correction tables.

Toroid Class

This class is monitoring the attached toroid signal. This channel is needed for beam-loading compensation (BLC).

Pyro Class

The Pyro signal, which allows to measure the compression in the bunch compressors, is connected to one of the ADCs. This class monitors this signal and sets a parameter needed for the pyro feedback into the firmware.

ACC1-ACC39 Class

The purpose of the 3rd harmonic module ACC39 is to linearise the 1.3 GHz RF signal. The two RF stations for ACC1 and ACC39 have to be operated in parallel. This class takes care, that amplitude or phase is set simultaneously to both stations between macro bunches.

DAQ ATTACHED SERVER

The FLASH DAQ[5] system pushes so called spectrum data (2K float array) from many front-end computer to a central shared memory with the 10 Hz repetition rate of the accelerator. This shared memory synchronizes this data on a macro-pulse basis. This allows to correlate data

from the whole machine easily. A second advantage of this scheme is, that this huge amount of data is transported only once over the network, but may be used by several DAQ attached server. A library called DOOCSddaq is provided to read spectrum or float data from the Buffer Manager directly. A trigger to the server is issued, after the data buffer are filled. This concept is used by the following LLRF server :

- Quench-detection: calculates from the I and Q values the Q loaded and detuning of each cavity. With these values a quench event can be derived. The server generates in case of a quench a flag per klystron section. This flag is used by the finite state machine to switch off the RF.
- LLRF diagnostic: this server calculates values like flattop mean, RMS, flattop slope, bunch to bunch stability and others for every cavity to generate performance statistics.
- PIEZO calculation: this server calculates the Lorenz force detuning of the individual cavities and drives the piezo front-end server accordingly.

AUTOMATION

The concept to automate the RF is based on a simple finite state machine (FSM) approach. The main purpose is to simplify the on/off-switching procedure and faster recovery from trips.

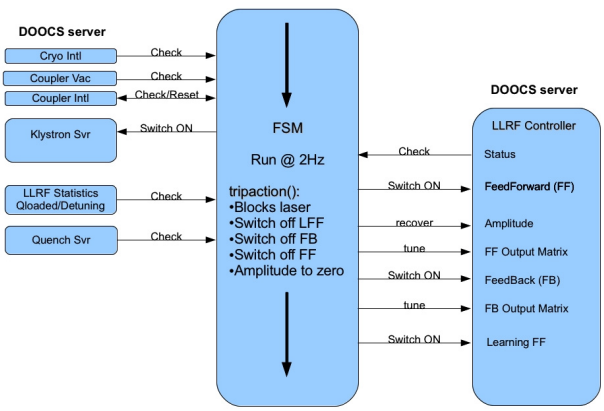

Figure 2: This picture shows the FSM concept for one RF station

This FSM is realized in the standard DOOCS server framework with the addition of the DOOCSdfsm library, which provides simple classes for monitoring float values, recover set-values or resetting interlocks. The FSM is the central server for the automation, it starts up or switches off the whole RF. All actions in other server are triggered by the FSM, giving the operator one central location to look for the status or problems of the RF system; no other software should switch the RF.

The FSM runs with a repetition rate of 2 Hz, checking several things, like interlocks, coupler vacuum, klystron status or quenches. In case of a problem in one state, the so called tripaction() function is triggered to bring the system to a save condition, then the FSM tries to recover the RF system. The same states are checked, when starting-up or recovering from trips.

OUTLOOK

For the upcoming European XFEL project, it is planned to use xTCA as hardware platform, because of the modern PCIx communication and the standardized remote monitoring capabilities. The required down-converter and fast ADCs µTCA cards are already available, the LLRF controller board is in the design phase. Porting of the LLRF controller server code from the old SPARC VME CPU to a INTEL x86 CPU is in progress. The goal is to have one source code base only by exchanging the hardware interface through compile flags. Due to the much better performance of the INTEL CPUs, it will be possible to run most of the middle layer server, like quench detection locally.

SUMMARY

The LLRF system at FLASH has been consolidated to one unified set-up for all RF stations in terms of hardware, firmware, software and naming conventions. Operator panels have been simplified and better expert panels have been designed.

The concept of a simple FSM is in standard operation, but some improvements have to be implemented to sort out conflicts between operator intervention and FSM recovery action.

The learning feed-forward algorithm has been ported from a MATLAB tool to the LLRF controller server and is in standard operation as well. Applications like vector-sum calibration or quench detection are implemented as DOOCS server already, but need more commissioning and tighter integration into the FSM framework. Further work is needed to improve the reproducibility of the RF system behaviour.

It is reasonable to say, that the first user run showed already improved performance of the LLRF controls and the new structure will be well-suited to be the base for the European XFEL.

REFERENCES

[1] DSP-Based Low LEevel RF Control as an integrated part of DOOCS Control System, V.Ayvazyan EPAC2006
[2] http://doocs.desy.de
[3] JDDD": A Java DOOCS Data Display for the XFEL E.Sombrowski, K.Rehlich, ICAEPCS 2007
[4] LLRF Control System Upgrade at FLASH, V.Ayvazyan PCaPAC2010
[5] Buffer Manager Implementation for the FLASH Data Acquisition System, V.Rybnikow, PCaPAC2008