Control and Monitoring of the Front-End Electronics in ALICE

Peter Chochula, Lennart Jirdén, André Augustinus

CERN, 1211 Geneva 23, Switzerland
Peter.Chochula@cern.ch

Abstract

This paper describes the configuration and monitoring scheme, which is being developed for the ALICE Front-End Electronics (FEE). The scheme is common to all ALICE sub-detectors although each sub-detector has a different FEE architecture. It is based on a Front-End Device (FED) model, which removes the differences between the various systems and enables a common control approach to be applied. An implementation of the FED for the ALICE Silicon Pixel Detector is described.

I. INTRODUCTION

The ALICE experiment is designed for studies of heavy ion collisions at LHC energies. Requirements on excellent tracking precision, high momentum resolution and particle identification has lead to a relatively complex experiment design. Different detector technologies including silicon tracking detectors, TPC, Cherenkov detectors, transition radiation detectors, etc. are being implemented to satisfy the physics requirements. Sub-detectors have the Front-End Electronics (FEE) architectures optimized to their needs. Differences in their operation modes impose a broad spectrum of demands towards the online systems.

The device control and monitoring strategy in ALICE is focused on standardizing to the maximum possible extent. The choice of power supplies or monitoring systems (PLC, ELMB) is centrally coordinated and common solutions are proposed wherever applicable. The domain of FEE represents a significant complication to this approach.

Most of the ALICE sub-detectors based their FEE architecture on custom chips involving a large amount of parameters which need to be configured, controlled and monitored. Many different techniques are deployed to communicate with these chips (including JTAG, Ethernet, Easynet, GOL, Profibus, CANbus etc.), each requiring a different access strategy from the online systems.

The Front-End Device (FED), described in this paper, provides a solution to this problem by representing the different FEE's as a device which reacts to a set of standard commands and publishes gathered data. The FEE architectures are in this way made transparent to higher software levels.

II. ARCHITECTURE OF ALICE ONLINE SYSTEMS

Four systems are responsible for online operation of the ALICE sub-detectors: the Data Acquisition (DAQ), Trigger (TRG), High Level Trigger (HLT) and Detector Controls (DCS). Each system is partitioned following the ALICE detector architecture allowing for independent operation of individual sub-detectors and associated services.

DCS for each sub-detector is segmented into sub-systems. It treats the high voltage (HV), low voltage (LV), cooling, gas control and FEE individually. DCS coordinates operation of its sub-systems.

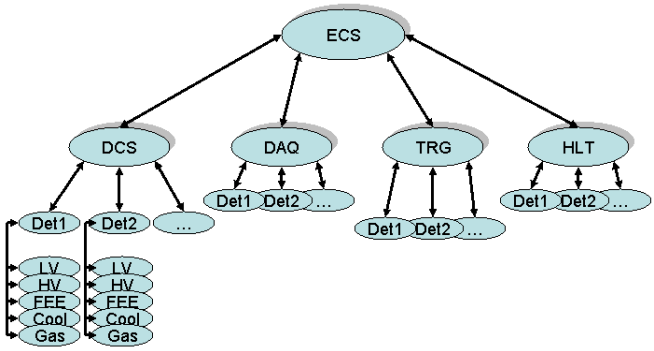


Figure 1: Hierarchical software architecture of ALICE online systems

Interaction between all online systems strictly follows the hierarchical structure. Exchange of commands and information is coordinated by the Experiment Control System (ECS) (Figure 1).

III. ALICE FRONT-END ARCHITECTURES

The first step in establishing a control and monitoring strategy for FEE's is identification of commonalities between the different hardware architectures. Grouping of sub-detectors into categories is based on access path used to download and retrieve data.

The standard link for collecting physics data in ALICE is the Detector Data Link (DDL) [1]. It also provides functionality for downloading configuration information. However, some of the sub-detectors do not use this option.

The FEE architectures are subdivided into the following four control and monitoring strategy classes:

- Class A: DDL is used for FEE control.
- Class B: DDL is used for FEE control and in addition an alternative path connected to DCS, using different technology, is implemented. In some cases both DDL and non-DDL technologies are used simultaneously, sending a part of the data using either path.
- Class C: control of the FEE is performed by the DCS exclusively.
- Class D: the same as class C, however, the control and monitoring share the same path and an access arbitration scheme is needed.

All four classes use an additional access path for monitoring. For class D this is shared with control tasks. Communication using DDL is treated by the DAQ software package; all remaining technologies are covered by the DCS using the concept of the FED.

IV. The Front-End Device (FED)

The FED (Figure 2) represents a hardware abstraction layer allowing DCS transparent access to the FEE. It responds to standard commands and performs requested tasks such as loading configuration registers, resetting the chips etc. If the FEE provides data which needs to be monitored, this is gathered by the FED and made available to the supervising software.

The FED is built as a bundle of software and hardware with a standardized software interface. The Distributed Information Management System (DIM) [2] has been chosen as communication software allowing for network transparent communication.

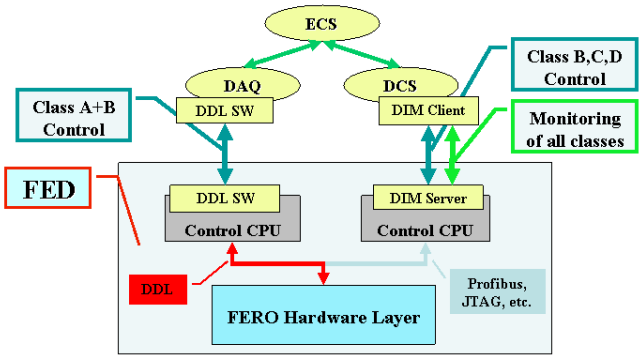


Figure 2: Architecture of the Front-End Device

A client-server communication model has been implemented for the FED. The server communicates with the hardware and publishes data in form of services. A client can subscribe to services and send commands to the server. Several clients can subscribe to the same server in parallel allowing for distributed monitoring.

The heart of the FED software is the FED server which is structured in three main layers. The highest layer is the DIM server, which provides the communication interface to the remote control system. The middle layer (sub-detector custom code) is translating standard commands received from DIM to sub-detector specific actions. Its role is also to process data provided by the FEE and transmit it to the DIM server. Finally, the bottom layer (the hardware access layer) provides access to the hardware itself. It communicates with the hardware controller of the implemented bus and is usually built using commercial components.

Parameters gathered by the FED are published as DIM services. The DCS client subscribes to these services at server start-up time and treats received data according to recipes reflecting the sub-detector requirements.

Using the features of the DIM protocol, DCS is able to evaluate communication status and distinguish between very low activity environment (thus not providing updates) and communication problems. After restarting any of the involved DIM components, the communication is re-established automatically.

Data published by the server can be grouped in two categories: General data provided by every FED (such as errors, warnings, internal server status) and sub-detector specific data (e.g. internal trigger counters, voltages, temperatures etc.). Similar to this we distinguish between General and sub-detector specific commands.

V. Implementation of FED for the ALICE Silicon Pixel Detector (SPD)

The ALICE Silicon Pixel detector (SPD) [3] has been chosen for evaluation of the above described approach. Using this sub-detector as an example we will describe basic architectural and functional features of the FED.

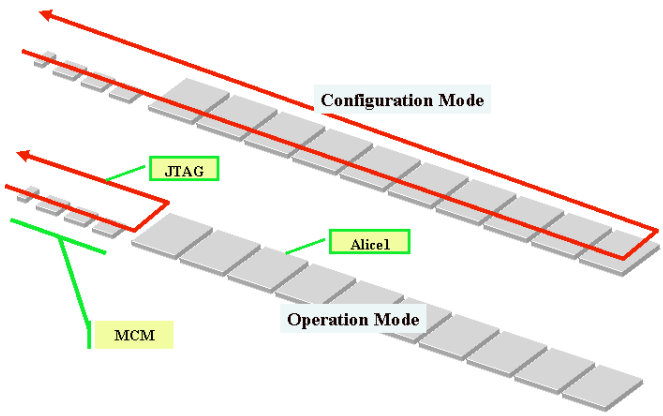


Figure 3: Operation modes of the ALICE Silicon Pixel Detector

The ALICE SPD has a relatively complicated FEE architecture. The basic readout unit, a half-stave, is formed by two pixel sensors each serviced by five bump-bonded ALICE1 readout chips. A carrier bus connects these chips to the multichip module (MCM). All chips are controlled and monitored using the JTAG bus. In order to avoid noise injection during physics run, the JTAG bus can be reconfigured by the MCM in such a way, that it does not enter the readout chips. It still services the MCM and is used for periodical reading of voltages and currents. Only when the ALICE1 chips need to be accessed (at start-up, after recovery from voltage regulator failure or SEU etc.) the JTAG bus is reconfigured and services also this part of a half-stave (Figure 3).

A group of six half-staves is connected via optical links (running over 100 metres) to a VME Router board which communicates with DAQ, DCS and Trigger systems. There are in total twenty routers servicing the whole SPD (Figure 4).

The VME bus of the router modules is allocated to the DCS. Physical connection between the control computer and VME crate is based on PCI-VME bridge (the National Instruments MXI-2 technology) using NI-VISA and NI-VXI as a software driver [4, 5]. The NI-VISA also forms the lowest layer of SPD FED server.

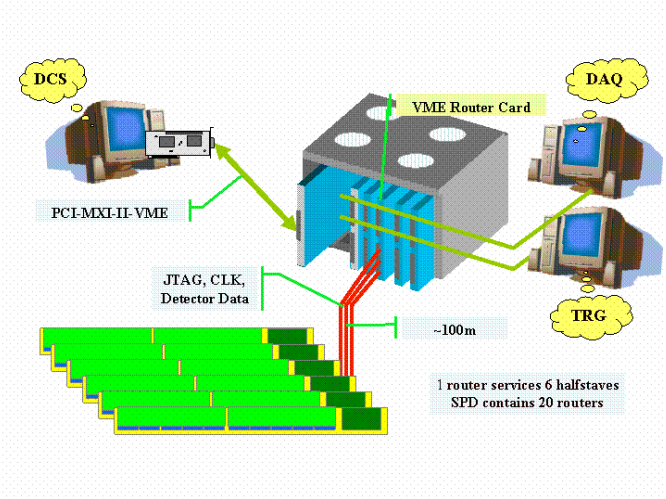
The SPD FED server is implemented on Windows operating system. The code is written in C++ using the Visual Studio.Net as development environment.

The SPD detector specific code is based on agents, which are implemented as separate program threads. Monitoring Agents (MA) perform periodical reading of desired parameters (MCM voltages and currents, SPD temperatures, status flags of voltage regulators etc.). Gathered values are published via DIM and sent to all subscribed clients. In order to minimize network traffic, only values exceeding predefined limits are published. The limits are set by the DCS client at start-up and can be modified during running in case that it is required by SPD conditions.

Control Agents (CA) are interpreting commands received from DIM. Recognized general commands include setting of monitoring frequencies and limits, FEE initialization and request for publishing server operation parameters.

On request a dedicated CA can initialize the SPD FEE. This is a general command recognized by each sub-detector; however, its implementation is dependent on the hardware architecture. All required data is loaded from the configuration database. In the case of SPD this data includes settings for 42 DACs implemented on each of the 1200 ALICE1 chips and local settings (threshold adjustment, mask and test register) for each of the 10 million pixels. The configuration database holds the settings for each chip, however, in order to limit the amount of data to manipulate, only the differences from an ideal reference chip are registered. In this case the required data is compressed by a factor of 40.

Figure 4: Hardware components of the SPD readout and control system



Connection to the database is implemented using the ADO technology [6], which makes the FED server independent from the type of underlying database used. Successful tests were performed using MySQL, SQL Server, MS Access and Oracle databases without having to change the FED server code.

SPD specific commands enable control of agent execution. For safety reasons, all monitoring agents are suspended at FED server start-up. A dedicated agent first verifies the status of the bus and communicates this to DCS, which can decide to start monitoring by issuing a corresponding command. Suspending of the MA's is typically required to avoid collisions when accessing the ALICE1 chips. This is managed by the FED server, which also restarts the agents after completion of a command execution. At start-up the FED server locks VISA resources and effectively disables hardware access from any other software. This protects the sub-detector against incorrect use of software.

The internal status of agents is indicated to DCS by dedicated DIM service. If an agent remains in suspended state for a long time the DCS can take corrective actions.

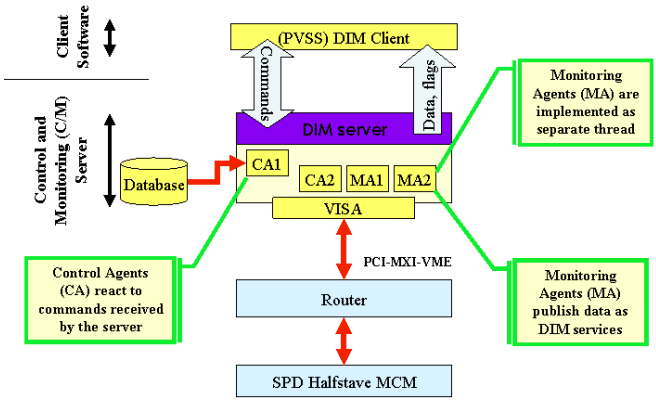


Figure 5: Structure of the FED server implemented for ALICE SPD

All server actions result in messages indicating success, failure, errors, warnings and debugging information. All messages are published as DIM services. In order to protect the DCS client from message flooding, the information complexity can be tuned. In normal operation only errors, failures and warnings are published. Optionally, the messages can be logged into a local file or attached to the Windows system event logger.

As the FED server remains the one and only access point to the FEE, it must provide any additional functionality required for sub-detector operation or tuning. These functions are implemented in form of sub-detector specific commands and executed as CA. SPD implements additional commands for checking the integrity of its JTAG bus, for SEU monitoring and recovery, for detector calibration and for local scans.

VI. Integration of FED into the DCS System

The main DCS operational tool is the SCADA system PVSS II complemented by a framework of tools [7] providing the necessary DIM functionalities. DCS is the main subscriber of FED data and is the only online system sending commands to the FEE via the FED. As some information published by the FED might be relevant to other online systems (e.g. internal trigger counters or readout status), clients outside DCS are allowed to subscribe to its services. Custom clients are typically implemented in C++, C or Java. Universal clients such as DID or DimTree (included in DIM distribution) are very useful for remote monitoring without the need of installing PVSS II system or writing custom software.

DCS related information provided by FED's (e.g. voltages or temperatures) is treated in the same way as information originating in any other device. If acquired values exceed predefined limits, DCS will pass this information to the operator and execute pre-programmed actions. Overheating of a module will for example result in switching off the corresponding LV channels, keeping the rest of the sub-detector operational. This approach is much more efficient for the experiment than interlocking the whole affected sub-detector.

The DCS is describing operation of its sub-systems and devices in terms of finite state machines, using the SMI++ framework [8]. This approach is also used for the FED allowing easy integration into DCS.

The SMI++ interface simplifies the task of synchronization between different sub-systems. For example, the FED moves into a state allowing for receiving of commands only after the corresponding low voltage becomes ready. Similar to this if the LV system indicates an error (e.g. voltage regulator over current condition), the FED moves to a Not-Ready state and waits for a command to reconfigure affected chips.

The state machine approach is further extended up to the ECS which allows for communication between online systems and maintains integrity of the whole experiment.

VII. Conclusions

A strategy for control and monitoring of different FEE architectures has been defined and successfully prototyped. The concept of the Front-End Device (FED) provides abstraction from the underlying hardware and allows for transparent access to sub-detector FEE.

Prototype implementation of the FED for the ALICE SPD shows encouraging results. Using a sub-detector having one of the most complex FEE architectures in terms of control and monitoring the concept enabling transparent access to detector hardware has been validated.

The described approach can be further extended to any devices which need to be remotely controlled or monitored. In ALICE it will be implemented for example for non-standard power supplies, or stepper motor control of calibration systems.

VIII. References

[1] ALICE-INT-1998-21 v.1.3, ALICE DDL - Hardware Guide for the front-end Designers

[2] C.Gaspar et al. "DIM, a Portable, Light weight Package for Information Publishing, Data Transfer and Inter-process

Communication", International Conference on Computing in High Energy and Nuclear Physics (Padova, Italy, 1-11 February 2000)

[3] P.Chochula et al. "The ALICE Silicon Pixel detector", NIM A715 (2003),849c

[4] http://www.ni.com/pdf/products/us/0vxv136a.pdf

[5] http://www.ni.com/pdf/manuals/321228a.pdf

[6] Data Access Technologies, http://msdn.microsoft.com

[7] http://itco.web.cern.ch/itco/ProjectsServices/JCOP

[8] B.Franek and C.Gaspar : "SMI++ Object Oriented framework for designing Distributed Control Systems" Presented at: Xth IEEE Real Time Conference 97 (Beaune, France, Sep 22-26 1997)