DCS Communication Software for the ALICE TPC Front-End Electronics

M. Richter, J. Alme, S. Bablok, D. Larsen, D. Röhrich, K. Ullaland
Department of Physics and Technology, University of Bergen, Norway
Matthias.Richter@ift.uib.no

K. Røed
Faculty of Engineering, Bergen University College, Norway

R. Keidel, Ch. Kofler
Center for Technology Transfer and Telecommunications, University of Applied Science Worms, Germany

T. Alt, D. Gottschalk, V. Lindenstruth, H. Tilsner
Kirchhoff Institute of Physics, University of Heidelberg, Germany

U. Frankenfeld
GSI, Gesellschaft für Schwerionenforschung, Darmstadt, Germany

Abstract

The ALICE Time Projection Chamber (TPC) is read out by 4356 Front-End Cards serving roughly 560000 channels. Each channel has to be configured and monitored individually. As one part of the overall controlling of the detector this task is covered by the Detector Control System (DCS).

Since fault tolerance, error correction and system stability in general are major concerns, a system consisting of independently running layers has been designed. The functionality layers are running on a large number of nodes and sub-nodes.

The low-level node controlling the Front-End Electronics is an embedded computer system, the DCS board, which provides the opportunity to run a light-weight Linux system on the card. The board interfaces to the front-end electronics via a dedicated hardware interface and connects to the higher DCS-layers via the DIM communication framework over Ethernet.

This article presents the structure of the communication software and the application of the DCS board.

I. Introduction

The ALICE experiment described in [1] will investigate Pb-Pb collisions at a center of mass energy of about 5.5 TeV per nucleon pair and p-p collisions at 14 TeV. The detectors are optimized for charged particle multiplicities of up to $dN_{ch}/d\eta$ of 8000 in the central rapidity region.

In general, the Detector Control System (DCS) covers the tasks of controlling the cooling system, the ventilation system, the magnetic fields and other supports as well as the configuration and monitoring of the Front-end electronics. Detailed information on the components and architecture can be found in [2]. The various components not concerning the TPC Front-end electronics will not be investigated further.

It is important to notice that the control system is detached from the data-flow. The data is transported from the Front-end electronics to Data Acquisition (DAQ) through an optical link. The main task of the control system is to avoid occurring system errors interrupting the data-flow.

Sharing of devices between different sub-systems is avoided whenever it's possible, so that independent operation is ensured. This is also an important issue in development and commissioning of the system, so that each sub-system can be debugged and tested separately from other parts of the system. This technique is called *partitioning* and is a widely used feature in the design of ALICE.

The Time Projection Chamber (TPC) is one of the main tracking detectors of the ALICE experiment. Charged particles ionize the gas volume on their way through the detector, the produced electrons drift in an electromagnetic field towards the end-caps where the charge is amplified and collected by a 2-dimensional readout system. Together with the drift time this provides a 3-dimensional resolution. The TPC consists of 36 sectors which are read out by 4356 Front-End Cards (FEC) serving roughly 560000 channels. All FECs have to be configured and monitored. Furthermore Programmable Logic Devices (PLDs) are widely used in all hardware devices of the TPC Front-end electronics to keep the system open and flexible. The configuration thus must include the upload of firmware to the FPGAs.

II. Hardware Architecture

Each of the 36 sectors of the TPC is read out by 6 identical subsystems. The TPC detector uses a specific hardware device, the Readout Control Unit (RCU), to control a set of Front-End Cards (FECs). An RCU contains the RCU motherboard, from now on referred to as RCU board, which hosts two additional interface boards customized for the ALICE experiment. The

Detector Data Link Source Interface Unit (DDL SIU) is the ALICE standard interface to the DAQ. The second card, the DCS board, is an embedded computer which implements the DCS/Trigger interface.

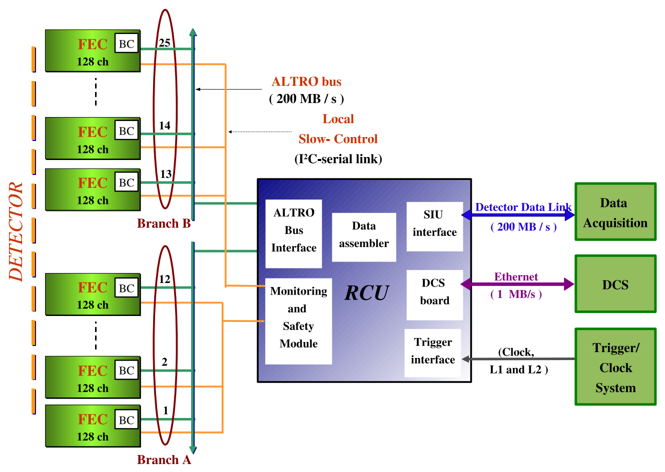


Figure 1: Components of the TPC Front-end electronics and data flow.

Fig. 1 shows an overview of the electronics. Between 18 and 25 FECs are connected to one RCU board via two bus systems. A fast 200 MB/s bus, the *ALTRO* bus, is intended to transport the event-data and the configuration data between RCU and FECs. The Slow Control bus allows monitoring and controlling the cards without interfering with the data readout process. The event-data is converted from an analog to a digital signal by the ALTRO chips ([3]) on the FECs. The ALTRO provides functionality for digital data processing and has the ability to reduce the data volume significantly. The data is then handled by the RCU and shipped out through an optical link that is sited on the SIU card. The data readout chain and the SIU card, as well as the RCU motherboard are not investigated further in this article. Details about the electronics can be found in [4].

III. DCS BOARD EMBEDDED COMPUTER

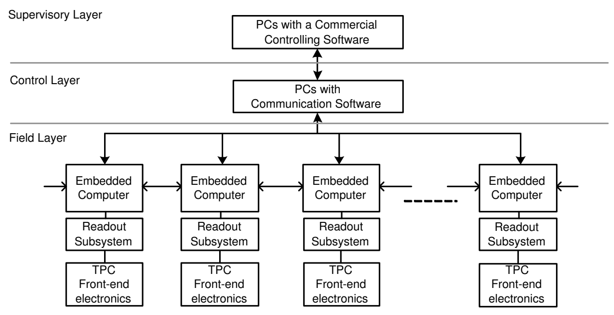
The low level nodes of the presented system are single board embedded computers, the DCS boards. The board is used in several detectors of the ALICE experiment and flexibility has always been a major concern. The core of the system is an Altera EPXA1, containing a 32bit ARM processor with cache and MMU (Memory Management Unit). Among other features there are 100k gates of PLD (Programmable Logic Device) available.

In addition to the FPGA, the board hosts a radiation tolerant 8 MB Flash ROM, 32 MB SDRAM, an Ethernet interface, an ADC (Analog-Digital Converter) for voltage and temperature monitoring, a JTAG connector, as well as dedicated datalines to the RCU board connector. All these components make the DCS board a custom-made, fully-functional computer. The components of the DCS board make it capable to run a lightweight version of Linux. A detailed description of the functionality can be found in [5].

The Linux operating system combined with the PLD and the direct access to the bus systems of the Front-end electronics is the chiefe cause for the flexibility of the system. Registers and memory on the RCU board are accessible from the operating system on the DCS board, either directly or indirectly. For that purpose memory mapped interfaces are defined and can be accessed via device drivers which define an interface between software and hardware. A change in the firmware usually only requires adaption of the driver. This modularization enables design changes without harming the whole system.

The design of the RCU board is based on an FPGA with firmware update possibility and enough space to host other tasks in addition to the readout. Among others, it contains modules for data assembling, bus interfaces, a Monitoring and Safety Module and trigger interfaces ([6]). The DCS uses the RCU board as a sub-node to the DCS board. Specialized tasks which are adapted to the underlying hardware run on the low-level nodes while the high-level control tasks are running on the node itself.

IV. SOFTWARE ARCHITECTURE



Supervisory Layer
PCs with a Commercial Controlling Software
Control Layer
PCs with Communication Software
Field Layer
Embedded Computer
Readout Subsystem
TPC Front-end electronics

Figure 2: Schematic view of the software architecture

A. *Functional Layers*

A sketch of the system is given in Fig. 2 showing the principles of the architectural layout. From top to bottom this is:

- The Supervisory Layer
  The Supervisory Layer consists of a number of PCs and provides user interfaces to the operator. It also interfaces to external systems and services, e.g. the LHC.
- The Control Layer
  The Supervisory Layer communicates with the Control Layer mainly through a LAN network. This layer consists of PCs, PLCs (Programmable Logic Cells) and PLC like devices. The Control Layer collects and processes information from the Field Layer, as well as sending commands and information from the Supervisory Layer to the Field Layer. It also connects to the Configuration Database.
- The Field Layer
  The Field Layer consists of all field-devices, sensors, actuators and so on. The DCS board and the readout electronics are located in this layer.

The tasks of the three layers are carried out by dedicated programs. They work in parallel, feeding the operator with useful information concerning the status of the system, or responding to commands given at the top-level. The components which cover the task within the three functional layers are shown in a more detailed view in Fig. 3.

A SCADA (Supervisory Control And Data Acquisition) system acts in the Supervisory Layer, through which the operator can access and monitor data points related to the hardware devices. A commercial controlling software, PVSS (Prozess-Visualisierungs- und Steuerungs- System by ETM[1]), has been chosen for the ALICE experiment. The DCS is not restricted to this specific controlling software but can feature any SCADA system.

The PVSS connects to the *InterComLayer*, a specific communication software acting as the Control Layer and connecting the hardware devices in the Field Layer to the controlling system in the Supervisory Layer. The system uses the communication framework *DIM* (Distributed Information Management System, [7]), which is based on the client-server principles. Several abstraction layers have been introduced:

- PVSS and InterComLayer communicate through a specific interface, the Front-End-Device (FED), which is common among different sub-detectors within the ALICE experiment. The InterComLayer implements a server which the PVSS can subscribe to as a client.

- Each hardware device implements a Front-End-Electronics-Server (FeeServer), which the InterComLayer subscribes to as a client.

The InterComLayer connects to several FeeServers and pools data before distributing it to the SCADA system. Vice versa the InterComLayer distributes configuration data to the FeeServers. In addition it implements an interface to the Configuration Database containing all specific configuration data for the hardware devices. The concept of the InterComLayer and the FeeServer is presented in detail in [8]. A few features will be outlined here.

B. *Communication protocol*

Communication between all layers is based on the DIM protocol. DIM is an open source communication framework developed at CERN. It provides a network-transparent inter-process communication for distributed and heterogeneous environments. TCP/IP over Ethernet is used as transport layer. A common library for many different operating systems is provided by the framework. DIM implements a client-server relation with two major functionalities.

- Services: The DIM server publishes so called services and provides data through a service. Any DIM client can subscribe to services and monitor their data. The DIM clients get notified about current values via a callback from the DIM server.

- Commands: A DIM server can accept commands from DIM clients. Server and client have to agree on the format of the command.

A dedicated DIM name-server takes control over all the running clients, servers and their services available in the system. Each server registers at startup all its services and command channels. For a client the location of a server is transparent. The control system benefits from the features of the DIM framework, e.g. independence of machine architecture, process recovery and load distribution.

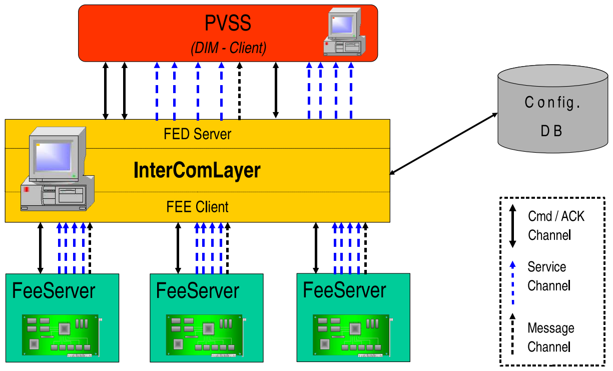


Figure 3: Software components and data flow

C. *The Front-End-Electronics-Server*

Fig. 3 shows a detailed view of the software components and the data flow. The DCS as described in this article is based on so called Front-End-Electronics-Servers (FeeServers) which run on the DCS board. A FeeServer abstracts the underlying Front-end electronics to a certain degree and covers the following tasks:

1. Interfacing hardware data sources and publishing data

2. Receiving of commands and configuration data for controlling the Front-end electronics

3. Self-tests and Watchdogs (consistency check and setting of parameters)

The core of the FeeServer is device-independent. It provides general communication functionality, remote control and update of the whole FeeServer application. Some features are related to the configuration of the data publishing. In order to reduce network traffic, variable *deadbands* have been introduced. Data is only updated if the variation exceeds the deadband. The core can be used for different devices, i.e. different detectors of the ALICE experiment.

The device dependent functionality of the FeeServer is implemented in a separate part, the so-called *ControlEngine* (CE). The CE provides data access in order to monitor data points and executes received commands specific for the underlying hardware. The *ControlEngine* has contact to the specific bus systems of the devices. The access is encapsulated in Linux device drivers.

[1]ETM professional control GmbH, www.etm.at

D. *InterComLayer*

The InterComLayer takes the task of the Control Layer. It runs independently from the other system layers on a separate machine outside of the radiation area. It provides three interfaces (see also Fig. 3):

- *Front-End-Electronics Client* (DIM client) to connect to the Field Layer
- *Front-End-Device Server* (DIM server) to connect to the Supervisory Layer
- Interface to the Configuration Database, using a database client

The InterComLayer connects to all FeeServers. After the connection is established, the InterComLayer subscribes to the services of the FeeServers and controls their message channels. Filtering of messages according to the log-level is performed on each layer to reduce network traffic. The service channels of the FeeServers are pooled together and re-published to the upper layer. By this means the source of the services is transparent to the SCADA system on top.

In order to transport configuration data to the Front-end electronics, the InterComLayer has an interface to the configuration database. Neither database nor InterComLayer know about the format of the data. The data will be handled as *BLOBs* (Binary Large OBjects).

In addition, the InterComLayer provides functionality for maintenance and control of the FeeServers. Servers can be updated, restarted and their controlling properties can be adjusted to any requirements.

V. Handling of configuration data

One major task of the DCS is the download of configuration data to the Front-end electronic. This concerns mainly the configuration of the Front-End Cards. This section will not explain the nature of the data, e.g. specific register configuration for certain behavior of a filter, but will focus on the handling of the data.

The ALTRO Bus Interface module which is implemented in the RCU firmware provides access to the FECs through command sequences. It implements a sequencer which decodes instructions written to the RCU Instruction Memory. The instructions are executed in the form of a single ALTRO instruction (RCU micro instruction) or sequence of them (RCU macro-instruction).

In order to have a better load balancing, the requests to the data base and the data download to the FECs is not carried out by the PVSS in the Supervisory Level. PVSS sends a command to the InterComLayer which fetches the configuration data directly from the data base and sends it to the FeeServer where it is interpreted inside the ControlEngine.

Using the ALTRO Bus Interface a basic instruction includes 3 steps:

- A command sequence is written to the RCU instruction memory
- Additional data (if necessary) is written to the RCU pattern memory
- A certain RCU register is set to trigger the sequencer

The ALTRO Bus Interface will then interpret the command sequence and ship data to or from the FECs.

The configuration data is organized into so called *Configuration Packages (CP)*. Each package corresponds to a basic sequence to write to RCU memory locations. Several such basic operations can be grouped into one *CP*. To the communication software they appear as *BLOBs*.

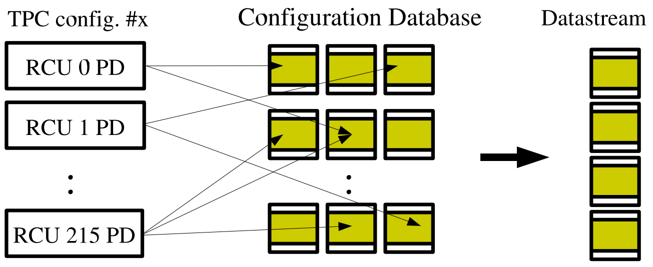


Figure 4: Archiving of configuration data

All *CP's* are archived in the configuration database. The configuration of an RCU is determined by a *Package Descriptor*, which is basically a list of all Configuration Packages needed for a certain configuration and is represented by a table in the data base. A complete configuration for the TPC is a list of *PD's*, one for each of the 216 RCUs (Fig. 4). As soon as the InterComLayer has received a request from the PVSS to load a certain configuration it fetches the result set for the corresponding table from the data base. It than iterates over all *Configuration Packages* and sends them down to the FeeServer where the packages are further processed by the Control Engine.

VI. Integration Test

During irradiation tests of the TPC Front-end electronics at the The Svedberg Laboratory(TSL) in Uppsala/Sweden in May 2005, the control system has been tested extensively. The setup is shown in Fig. 5. It consisted of an RCU motherboard, a DCS board and 9 Front-End Cards attached to the RCU. A second setup used one RCU board equipped with a DCS board running without any FEC attached. This setup was mainly used to test the interference between different tasks of the control system.

The FeeServer was running on the DCS board publishing 10 data points per FEC and a few memory locations in the RCU memory. Furthermore, a PC was running the InterComLayer. This integration test didn't involve the SCADA system in the Supervisory Layer. All steering of the setup has been done via command line interfaces. The InterComLayer subscribed to the services provided by the FeeServer and wrote the values into files. In addition, all log messages from the FeeServer were collected, filtered and written to a file.

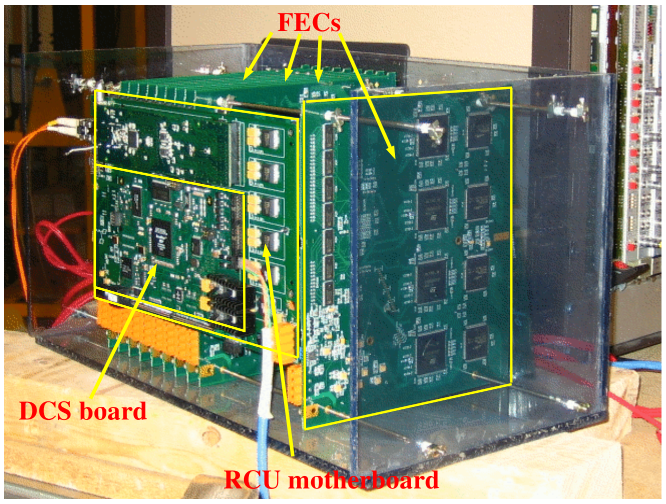


Figure 5: The setup for the irradiation tests at TSL

A few problems occured in the startup phase including problems with the DIM framework which caused the FeeServer to crash under certain circumstances. These has been identified as timing problems and the software has been adapted to avoid them. After solving the problems, the control software ran stable during 3 days until the end of the irradiation test. The integration test has been continued afterwards and no major problem could be observed. There is some work necessary to avoid interference between the FeeServer and the Active Reconfiguration.

VII. Summary and Conclusion

The presented communication software has the task of connecting the User Interface in the Supervisory Layer with the Front-end electronics in the Field layer. The System is based on servers which are running on a custom-made embedded computer, the DCS board, inside the electronics and have access to the specific bus systems.

The DCS board is a fundamental part in the distributed system which allows running complex controlling software under the operating system Linux. Tasks can be processed in parallel on the DCS board and on the connected custom hardware devices. Furthermore, the Linux operating systems on the embedded computers provides flexibility and well known interfaces.

The System is designed to be independent of physical intervention. Software and firmware are easily reconfigurable. Together with the distributed and module-based design with well-defined interfaces, this increases the flexibility and testability of the system.

The system is still under development. The modularity makes it possible to test and review each sub-system on it's own independently of the complete setup, and several tests have been performed with satisfying results.

References

[1] ALICE Collaboration, "ALICE Technical Proposal for A Large Ion Collider Experiment at the CERN LHC," CERN/LHCC 1995-71, 1995

[2] ALICE Collaboration, "Technical Design Report: Trigger, DAQ, HLT, DCS," CERN/LHCC/2003-062.

[3] R. Esteve Bosch *et al* , "The ALTRO Chip: A 16-channel A/D Converter and Digital Processor for Gas Detectors," *IEEE Transaction on Nuclear Science*, Vol. 50 No. 6, Dec. 2003.

[4] L. Musa *et al* , "The ALICE TPC Front End Electronics," in *Proc. IEEE Nuclear Science Symposium*, 2003

[5] H. Tilsner *et al* , "Hardware for the Detector Control System of the ALICE TRD," in *Proc. 9th Workshop on Electronics for LHC Experiments*, 2003

[6] C. González Gutiérrez *et al* , "The ALICE TPC Readout Control Unit," in *Proc. 10th Workshop on Electronics for LHC and future Experiments*, 2004

[7] C. Gaspar *et al* , "DIM, a Portable, Light Weight Package for Information Publishing, Data Transfer and Interprocess Communication," Presented at the Int. Conference on Computing in High Energy and Nuclear Physics, Padova, Italy, 2000

[8] S. Bablok *et al* , "Front-End-Electronics Communication software for multiple detectors in the ALICE experiment," *Nucl. Instrum. Methods* , accepted for publication