Generic and Layered Framework Components for the Control of a Large Scale Data Acquisition System

Stefan Koestner, *Member, IEEE*, Dominique Breton, Daniel Charlet, Flavio Fontanelli, Markus Frank, Clara Gaspar, Guido Haefeli, Richard Jacobsson, *Member, IEEE*, Beat Jost, Giuseppe Mini, Niko Neufeld, Ricardo Nogueira, Cedric Potterat, Patrick Robbe, Mario Sannino, and Ioana Videau

***Abstract*—The complexity of today's experiments in High Energy Physics results in a large amount of readout channels which can count up to a million and above. The experiments in general consist of various subsystems which themselves comprise a large amount of detectors requiring sophisticated DAQ and readout electronics. We report here on the structured software layers to control such a data acquisition system for the case of LHCb which is one of the four experiments for LHC. Additional focus is given on the protocols in use as well as the required hardware. An abstraction layer was implemented to allow access on the different and distinct hardware types in a coherent and generic manner. The hierarchical structure which allows propagating commands down to the subsystems is explained. Via finite state machines an expert system with auto-recovery abilities can be modeled.**

***Index Terms*—Embedded controllers, experiment control system, finite state machines, long distance protocol, SCADA.**

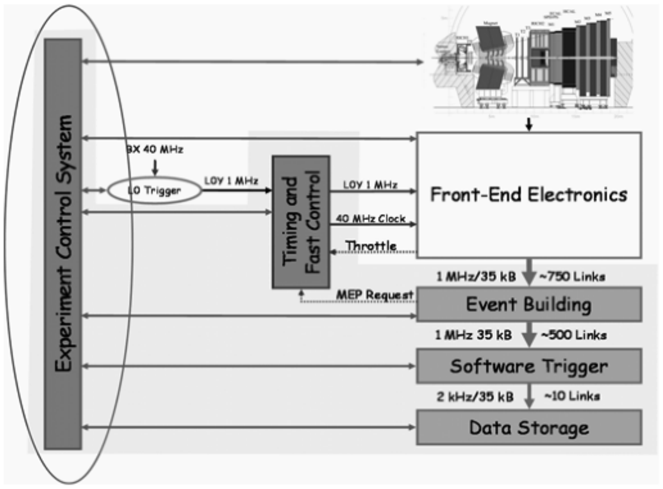


Fig. 1. Visualization of the scope of the experiment control system as being used for LHCb.

I. Introduction

To KEEP development time of the control systems for LHC at a minimum, CERN made the decision to adopt an industrial Supervisory Control And Data Acquisition (SCADA) system and to provide a common Framework to be used by all experiments at LHC. This Framework developed by the Joint COntrols Project (JCOP) [1] provides besides others the possibility to model a hierarchical control system using control units to send and propagate down commands or monitor the sub-tree below as well as device units acting directly on the hardware.

In LHCb these tools are used to build the Experiment Control System (ECS) which is the only interface for the operator to handle the configuration, monitoring, and operation of all experimental equipment. This reaches from experimental infrastructure like magnet, cooling and ventilation over detector operation like gases, high and low voltages to Data AcQuisition (DAQ) and trigger which includes readout electronics, timing, and event filter farm. Operations can be as simple as writing single registers to hardware or more sophisticated as downloading firmware to FPGAs. The relationship between ECS and the detector is shown in Fig. 1. Using the provided framework components it is guaranteed that a coherent and homogenous control system is developed by the users being in charge for their sub-detectors.

At LHCb there are two main possibilities foreseen to access custom electronics. In the proximity of the interaction point, where radiation is high, the radiation tolerant Serial Protocol for the Experiment Control System (SPECS) [2] is used. SPECS is a long distance serial protocol connecting a master in the counting house with its radiation tolerant slaves on the detectors. Electronics boards outside the radiation area, e.g., used for data acquisition, are accessed via an embedded Credit Card sized PC (CCPC) [3]. The CCPC has access to the board's registers through a glue-card [4] and is connected to the ECS via a dedicated Ethernet LAN.

Both systems communicate with the ECS using the Distributed Information Management (DIM) protocol [5], which is also part of the JCOP Framework. An additional abstraction layer is put in place to separate the hardware access mechanism from the control system, allowing for a register name based interface, which is independent from the actual hardware type in use.

In this paper we present the hardware layer as well as the structured software layers needed to transport an event in a coherent manner from any hardware register to the control system,

Manuscript received May 21, 2007; revised October 29, 2007.

S. Koestner, M. Frank, C. Gaspar, R. Jacobsson, B. Jost, N. Neufeld, and R. Nogueira are with the Centre Europeen de Recherche Nucleaire (CERN), Geneve 23, 1211 Geneve, Switzerland (e-mail: stefan.koestner@cern.ch).

G. Haefeli and C. Potterat are with the EPFL Lausanne, CH–1015 Lausanne, Switzerland.

D. Breton, D. Charlet, P. Robbe, and I. Videau are with the Laboratoire de l'Accelerateur Lineaire, F-91405 Orsay, France.

F. Fontanelli, P. Mini, and M. Sannino are with the INFN Genoa, I-16146 Genoa, Italy.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2007.913471

where its impact can trigger an alert or a completely automatic recovery—or the other way round—to transport a completely abstract command down the various layers till it finally writes to registers at hardware level.

II. The PVSS SCADA System and the JCOP Framework

Over a period of more than 3 years a working group set up by the CERN Controls Board has evaluated various industrial products capable to cope with the requirements of controlling high energy physics experiments [6]. In 2000 the decision was made in favor of PVSS [7], [8], which is a commercial software package for developing SCADA applications from an Austrian company, ETM AG in Eisenstadt.[1]

PVSS is a German abbreviation for 'process visualization and control system'. It offers a highly modular design with specific managers (processes) for different tasks developed upon a client-server architecture. The communication is based on a standard TCP/IP message interface and is event driven, which means that it is only active on demand. For the development of user-specific applications PVSS provides a platform independent scripting language.

It is a procedural high level language using the same function set as ANSI-C together with some additional PVSS internal functions that become handy especially for string manipulations. It allows for developing GUIs (in version 3.6 the QT environment is integrated) as well as scripts running in the background. For more complex tasks an API interface/manager allows to interface to a C++ library for extra functionality.

In High Energy Physics where experiments can have more than a million of readout channels, scalability is an important issue. PVSS offers a device oriented, structured namespace to create and manipulate complex devices. There is no built-in limit neither for the number of devices nor the number of elements of a device. The data is held in the memory of the event manager and the real time database from where it is accessible to all managers. Attributes can be set e.g., for processing and alerts. An interface to an Oracle database to record archiving data can be chosen inside PVSS which also eases the accessibility needed later on for trending.

Due to the architecture of communicating processes as visualized in Fig. 2, PVSS can take advantage of multi-CPU systems which allows for load distribution. Systems may also be distributed across machines and various autonomous systems may be connected with each other. This allows also for a hot-standby redundancy. PVSS is a cross-platform system not just with respect to development but also in the sense of operating truly mixed systems. Like this, one can profit from the advantages of different platforms.

To enlarge the functionality of PVSS and to better adapt it to the needs of the LHC experiments and machine, the JCOP Framework has been developed. It provides additional interfaces to commonly used devices including high voltage power supplies and CAN bus. In addition a dedicated Framework interfaces to an Oracle based configuration database to define and operate on recipes. Recipes are a set of parameters (registers) to

[1]ETM Corporation, Eisenstadt, Austria, [Online]. Available: http://www.etm.at/english/index.htm.

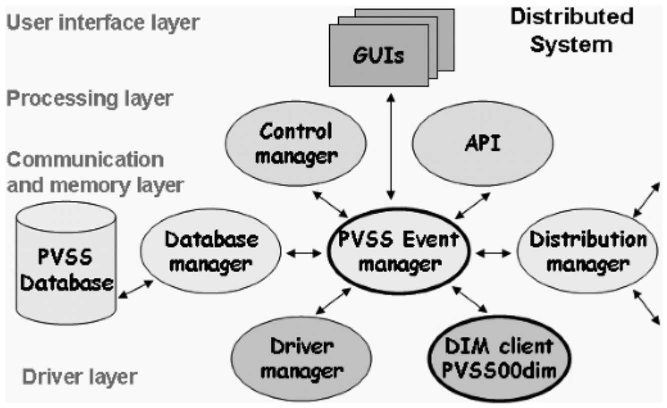


Fig. 2. Schematics of the modular design of PVSS showing the interconnected processes which allow to build a distributed control system. The central unit is the Event-manager. Different projects can communicate with each other via the Distribution-manager. Communication to the hardware happens through the Driver-managers.

be uploaded before data taking is started or any specific action on the experiment is requested. The JCOP Framework provides also some Application Programming Interface (API) managers for communication protocols like DIM.

The ECS of LHCb will be implemented as a hierarchical control system with a single run control as top node. The JCOP Framework allows for modeling of such hierarchical structures by providing an interface to the SMI++ framework [9] which allows to define rules for finite state machines. The main aim of the JCOP Framework is that users will end up with a coherent and homogenous view of the control system for the entire experiment.

III. The SPECS Protocol and Embedded Credit Card Sized PCs for Accessing Custom Electronics

The SPECS is an evolution of the ATLAS SPAC (Serial Protocol for the Atlas Calorimeter) designed for the configuration of remote electronics elements. An intermediate, mezzanine board is used to translate the SPECS long-distance protocol, used for transmitting data between the counting room and the radiation area (about 100 m), into the short-distance protocol (a few meters), used by the front end electronics. It is a 10 Mbit/s serial link with two signals in each direction (clock and data). SPECS is a single master multi-slave bus (with up to 32 slaves) as sketched in Fig. 3.

The SPECS master board hosts 4 SPECS masters and is implemented on a standard 32-bit 33 MHz PCI board, which can be plugged into a PC. The core of the system is implemented on the Altera CYCLONE FPGA. The slave is designed as a portable VERILOG code and is integrated in an ACTEL APA 150 flash FPGA. This technology guarantees immunity against single event latch-ups (SEL) and was tested for radiation hardness up to an integrated dose of 40 krad [10] which corresponds to the lifetime of the experiment including generous safety margins. Irradiation tests were performed using Krypton ions with an energy of 73 MeV/A. Hardness to single event upsets (SEU) can be achieved by appropriate redundancy of registers due to triple voting and the use of one hot state only. It is not foreseen to refresh the firmware remotely in regular intervals.

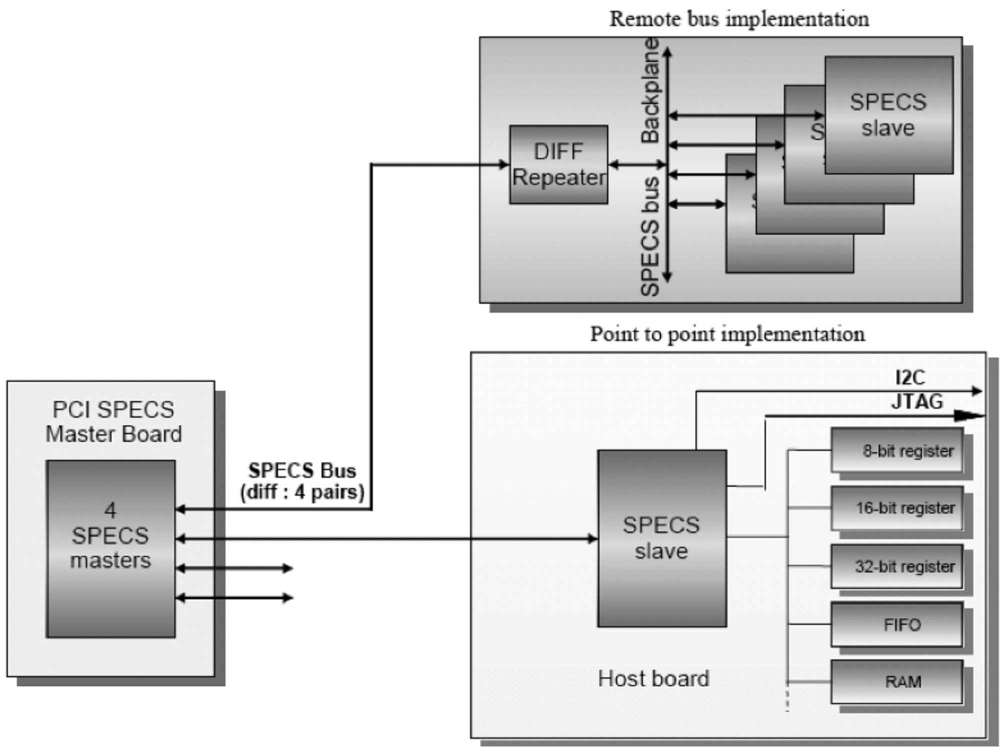


Fig. 3. The SPECS protocol: Two possible implementations demonstrating the master and slave architecture.

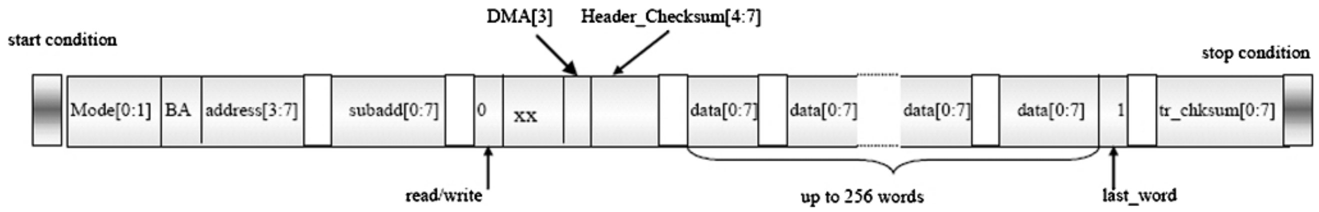


Fig. 4. An example of a SPECS frame: The SPECS protocol is implemented similar to the I2C protocol but without acknowledgements to increase the data throughput. Checksums are added to the header and trailer to detect corrupted data.

The SPECS slave is hosted on a mezzanine board. The whole board provides beside others a local and long distance I2C bus, a JTAG chain, a 16 bit parallel bus, a decoder for the channel B of the TTCrx [11] signal, a DCU [12] chip with 6 ADC channels of 12 bit resolution together with one temperature channel and a PROM which will allow the ECS system to obtain some information about the front end element housing the mezzanine card.

The protocol used for SPECS is similar to the I2 C protocol with start and stop conditions implemented as a specific transition of the data line when the clock line is at a high level. An example of a complete frame is sketched in Fig. 4. The 10 MHz clock is just active when data is transferred. A maximal SPECS frame length of 256 data words was chosen. To ease debugging by oscilloscope the data words are separated by missing clock cycles. Each word consists of 9 bits where the 9th bit tags the last data word of the frame. By avoiding the acknowledgements from the slaves much higher data rates can be achieved. The SPECS cannot correct for errors but if an error is detected the slave can send an interrupt after which the data is sent again. Two error detection schemes are implemented depending whether the error occurs in the data or the address. Four bits of redundancy follow the header of the frame, which contains the address of the slave and other parameters. In addition one byte of redundancy is sent at the end of the frame which allows the detection of a data error. If an error is detected, the slave ignores the whole message, and sends an error interrupt to the master straight away.

On the counting house side, where radiation damage is not an issue, embedded credit card sized PCs are attached to the FPGA based DAQ and trigger boards. These embedded PCs provide the necessary local intelligence and allow to access the various components of a board. Each micro controller has an isolated access path over a local area 10/100 Ethernet network to the Experiment Control System. This design contributes to the robustness of the entire system. In opposite to the usage of a proprietary bus, such as Fastbus or VME, where the cards are interconnected inside the crate, it cannot happen that a failure of a defect card or crate controller could affect other cards. In addition this design helps to decentralize intelligence and thus avoid bottle necks.

For local board control the SM520PCX from the Swiss company DigitalLogic AG[2] was chosen, which is a complete em-

[2]Digital Logic AG, Luterbach, Switzerland, [Online]. Available: http://www.digitallogic.ch

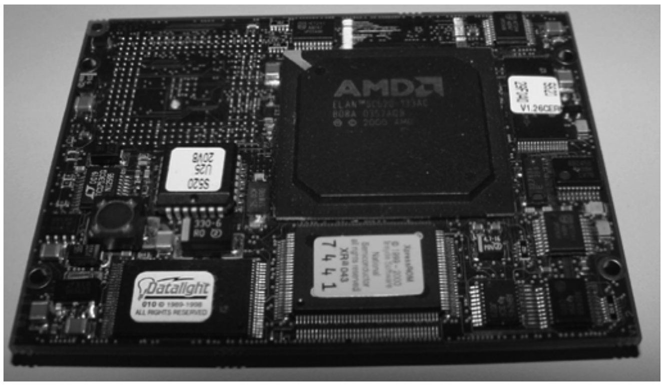


Fig. 5. Embedded Credit card sized PC. Top view of the SM520PCX.

bedded PC, based on the i486 compatible AMD ELAN 520 micro controller running at 133 MHz as shown in Fig. 5. It comprises all necessary hardware for diskless operation on a plug-in board of 85 x 66 mm ^{2}.

As operating system Linux has been chosen as it can be operated easily via remote connection, it can be booted via network and the kernel can be customized, such that unnecessary functionality can be removed to save valuable resources. On top of a slightly modified kernel the Scientific Linux [13] distribution of Fermilab and CERN is running. Except for the device driver no special software or cross-compilers are needed, which allows to run development tools under a faster and more powerful server sharing the file-system with the embedded PCs via NFS. Software distributions and bug-fixes are managed centrally based on simple and automatic update mechanisms.

In the case of LHCb three interfaces are sufficient to access all components on a board. These are I2 C, JTAG and a parallel bus. To provide an access to these interfaces on board, the embedded PCs are directly attached to a glue-card which is built around a PLX PCI9030 PCI bridge, JTAG and I2C controllers. A small FPGA is used to map the control registers of these controllers into the address space of the local bus. The glue-card contains a JTAG and an I2C hub to provide 3 independent JTAG chains and 4 independent I2C buses respectively. The PLX provides in addition support for interrupts and several General Purpose I/O (GPIO) lines. The glue-card also includes a bus switch which electrically isolates the glue-card from the carrier boards, when the PC is rebooted. A remote download of the firmware is possible via the fast JTAG chains. When reading from memories attached to the local bus from PCI, transfer performances better than 20 MB/s were achieved.

IV. Communication Layer and Abstraction of Hardware Register Access

A. *Distributed Information Management—DIM*

For communication between the embedded micro controllers or the SPECS master cards with the Experiment Control System the DIM protocol was chosen. DIM is a portable lightweight publish/subscribe system based on TCP/IP. It requires a DIM Name Server (DNS node) to which all DIM servers publish

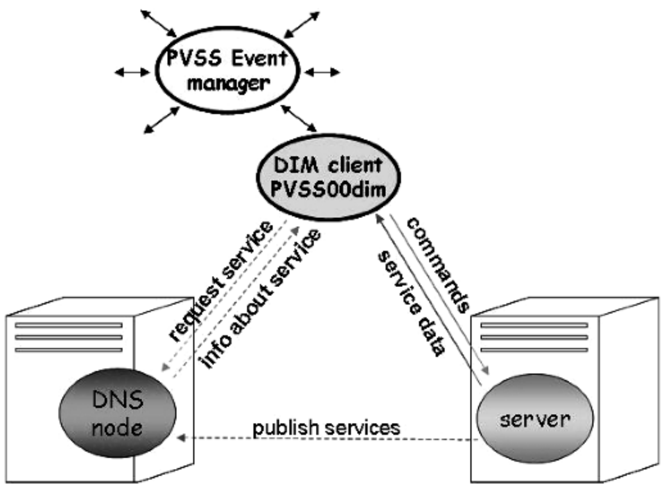


Fig. 6. DIM protocol. A PVSS API manager acts as a DIM client and connects the ECS to servers running on the hardware components. The database manager holds an image of the hardware in form of well structured PVSS datapoints.

their available services. DIM clients can later on request services from the DNS node. The DNS node hands over all necessary information about where to find this service to the client, e.g., IP address, so that a direct peer to peer connection can be established between server and client. Commands can be sent from client to server and services are sent from server to client. The advantage of this design is its portability, as clients can be installed on any machine by just specifying the DNS node. It also contributes to robustness as crashed servers can easily re-publish their services on the DNS node.

Generic DIM servers have been written to run on the micro-controllers of the embedded PCs and the host PCs of the SPECS master cards from where they have access to all the low level libraries containing the drivers to communicate with the hardware components. Thus they can perform all necessary actions on the various different boards e.g., write and read operations on registers and memory blocks, FPGA programming, monitoring of registers, etc. The JCOP Framework provides a special API manager (PVSS00dim) which can act as a DIM client. A special set of functions can associate datapoints, which is the internal representation of the SCADA system, to a DIM command or connects them to subscribed services. The servers are started automatically and publish their services on the DNS node running on a support PC who keeps their coordinates and from where they can be requested by the PVSS clients. Services can be subscribed to either on change or on a time basis. The communication process between the hardware and ECS via DIM is demonstrated in Fig. 6 which is an extension of Fig. 2.

Framework components including a graphical user interface and basic functions to write to and read from the various registers are provided for SPECS and CCPC. The basic mechanism is that the whole representations of a register (bus type, address, register width, data, etc.) are written to specific datapoints associated to a hardware type. These datapoints are connected to the DIM API manager in such a way that as soon as the data is written to the datapoint elements a DIM command is launched sending the information to the server. On the server side a callback function is called. From within this callback

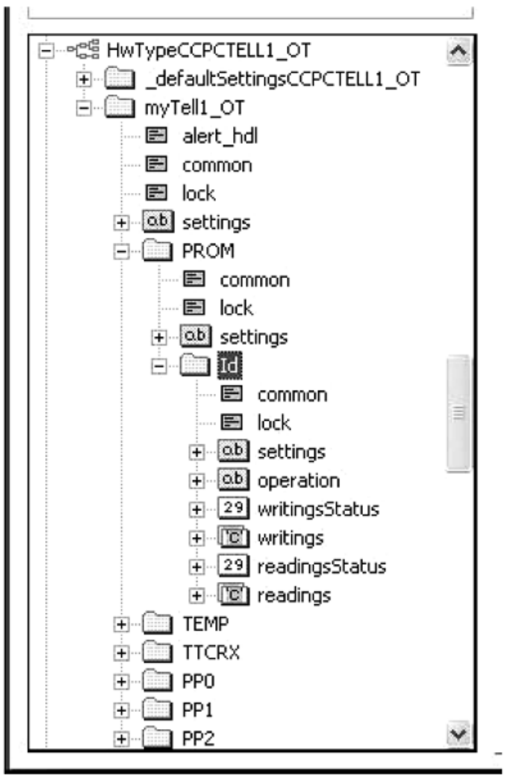


Fig. 7. A datapoint structure representing a real hardware device inside the ECS. The datapoints are always up to date with the real hardware—depending on the refresh rate.

function the hardware is accessed and an associated service is launched sending the hardware information back to the client where it is written to a datapoint element. A change on this element can again call a PVSS callback function or raise an alert.

B. Abstraction Layer—FwHw Tool

An interface which requires the complete register information to be passed as parameters is cumbersome and error prone, in addition it implies that software developers must have a profound knowledge about the hardware. Inserting an intermediate abstraction layer based on register names allows also to model and mirror the hardware inside the ECS in an intuitive manner.

As already mentioned the key-element to store data inside PVSS is a datapoint. It is distinguished between datapoint types, which define the internal structure, and datapoints which can be seen as instantiations of a datapoint type. As PVSS has a non-flat namespace complex structures can be created. Each register can be seen now as such a datapoint containing a well defined set of datapoint elements connected to the DIM API manager, as shown in Fig. 7. These registers can now be grouped according to the components they belong to. The components are further grouped to finally form a complete electronic device. Each instantiation of such a structured datapoint type finally represents a hardware device in reality, e.g., electronic board.

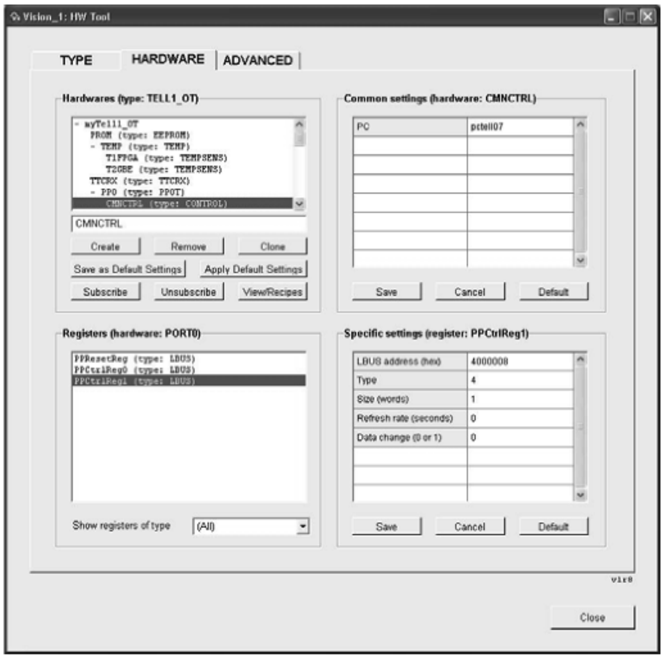
Once registers are created they can be subscribed. This means that all the information about a register which is stored under the datapoint element ‘settings’, as shown in Fig. 7, is sent to the server. On the server side a connected list is created which can be searched for by register names holding all the information in the according structure. At the same time two DIM services are launched (write and read) and one command for each register following a well defined naming convention distinguishing between SPECS and CCPC. From now on communication between server and client is established. A parameter can be written into the registers datapoint element ‘operation’ which launches a DIM command, interpreted at the server side either as write or read command according to the parameter value passed. On the server side the register information is retrieved and action follows according to bus type specification. After the hardware access a DIM service is invoked to send the data and information on the success to the client, where it is stored either in the datapoint elements ‘readings’ (the actual data) and ’readingsStatus’ or ‘writingsStatus’. To decrease traffic a write/read command can be launched which sends the data to the hardware and immediately reads it back.

Sometimes registers consist of various parameters consisting just of a few bits. In order to not overwrite important information a masked write operation is introduced which allows to write just a defined number of bits without touching the rest of the register. DIM services are normally updated at regular intervals to refresh the register values inside the ECS. By looking immediately at server side if the values have changed, the service can be suppressed and thus unnecessary traffic on the local area network can be avoided.

In the design described above all the diversity and complexity of the various hardware and bus types are hidden inside the server. Separation of the access mechanism on the hardware and the modeling of the components in the control system allow the reuse of various components on different hardware types even using different protocols, e.g., a special front end chip can be used as SPECS as well as CCPC device. In the end each board is represented as an instantiation of a datapoint type, which reflects the entire state of all controllable resources on the board. PVSS allows also associating event triggered functions with data points. Whenever the content of a data point element (= register value) changes, a function may be called in a PVSS script or API manager to perform a set of actions associated with the change.

In order to facilitate the modeling of hardware as PVSS datapoints a tool was introduced, FwHw. This tool offers a graphical user interface, as shown in Fig. 8, which allows assigning crucial information, like bus address or register length, to the various register types. The tool automatically creates the well defined datapoint structure connected to the DIM API manager and sends as well instructions to the server. Once the registers are created and subscribed a set of framework functions allows for interaction in an abstract way by just passing the register name as parameter.

In addition the tool allows for defining recipes. Recipes are a set of predefined register settings which can be retrieved from the configuration database and uploaded to the readout boards. Different settings can be applied for different run conditions. Each board has several hundreds of registers and memory blocks to write, so that optimization of write and read accesses is crucial for the start up configuration of the experiment, when all the register settings are retrieved directly from the configuration database.



Hardwares (type: TELL1_OT)

```
- myTell1_OT
   PROM (type: EEPROM)
   - TEMP (type: TEMP)
      T1FPGA (type: TEMPSENS)
      T2GBE (type: TEMPSENS)
   TTCRX (type: TTCRX)
   - PP0 (type: PPOT)
      CMNCTRL (type: CONTROL)
```

CMNCTRL

Common settings (hardware: CMNCTRL)

PC	pctell07

Registers (hardware: PORT0)

```
PPResetReg (type: LBUS)
PPCtrlReg0 (type: LBUS)
PPCtrlReg1 (type: LBUS)
```

Show registers of type (All)

Specific settings (register: PPCtrlReg1)

LBUS address (hex)	4000008
Type	4
Size (words)	1
Refresh rate (seconds)	0
Data change (0 or 1)	0

Fig. 8. Screenshot of the FwHw tool used for modeling electronic devices as PVSS datapoints.

V. A Hierarchical Control System Using Finite State Machines

In order to be able to build up a hierarchical control system each electronic device is modeled as a Finite State Machine represented by a device unit. States have to be defined which are intuitively representing the actual state of the board. States can be reached from another state by well defined transition paths as shown in Fig. 9 for a FSM used inside the DAQ domain. Each domain however may have different states and transition schemes. In LHCb it is distinguished between 4 domains: Detector Control System (DCS), High Voltage (HV), DAQ, and DAQ Infrastructure (DAI). A device unit is always the leave of a hierarchical tree and is directly attached to a datapoint, which is mirroring all the register values and memory blocks. In a script, which is translated to SMI++, it can be defined how the device unit shall react once a defined datapoint element has changed. Thus state transitions can be triggered if some important register values are changing. Also the other way round, actions can be performed on the datapoint upon receiving commands from a control unit. In this way a command can cause direct action on the hardware as writing to the datapoint elements (registers) will trigger immediately a DIM command to be sent down to the hardware server.

Control units can act on a number of device units (children) at once and thus allow for a better grouping and structuring of the hierarchical tree. If the state of any child changes rules can be defined according to which the control unit may transit to another state. Control units can also act on other control units finally building up the hierarchical tree with the Run Control as top node from where the commands are sent and propagate down the tree to finally reach the device units, as shown in Fig. 10. Like commands propagate downwards the hierarchical tree, statuses are propagating upwards the hierarchical tree leading to an integrated view of the whole experiment at the top node.

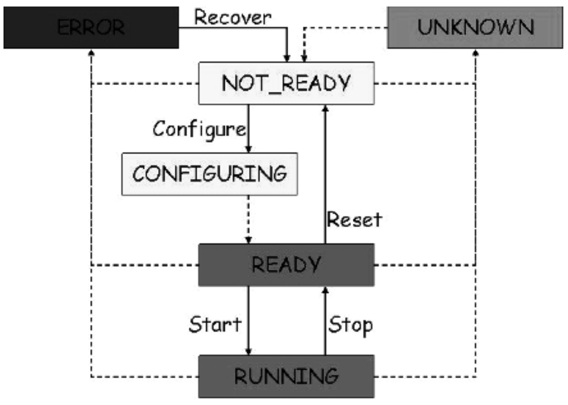


Fig. 9. Finite State Machines. A transition scheme as defined for the DAQ domain.

In this way an error occurring on the hardware can first move the corresponding device unit into state error. This state is then transported upwards by the control units according to the rules defined there and finally reaches the Run Control at the top where the run may be paused till the error is recovered. However, an auto-recovery scenario may be thought of immediately at device unit level. Once the transition has reached the state ERROR scripts can be invoked which check the hardware for standard errors followed by routines to fix them. Upon success the state returns back to NOT_READY.

For DAQ boards the main operations are to download the firmware to the FPGAs, which is allowed from state NOT READY, and the configuration of the registers. The main configuration of the board, basically downloading recipes from the configuration database, happens in the transition from state NOT READY to READY. The content of the recipes can differ according to the run type, which can be specified as a parameter when launching the command for configuration. Little action is required from state READY to RUNNING. The state UNKNOWN is reachable from all states and defines the state when control is lost e.g., a communication problem occurred.

In conjunction with the error recovery provided by SMI++ full use will be made of the powerful alarm handling tools provided by PVSSII. In addition the system allows for partitioning, which is the capability of monitoring and/or controlling a part of the system independently and concurrently with the others in order to be able to make tests, or perform calibration runs on a subsystem. The complexity of such a subsystem is demonstrated by Fig. 11 which shows the sub-tree of the Inner Tracking Detectors. A brief tutorial and some more information about how to build a FSM tree using the Framework components as well as how to use the communication and abstraction layers can be found in [14]. A more general description of the entire ECS of the LHCb detector can be found in [15].

VI. Conclusion

LHCb has chosen to use embedded processors with an isolated access path for board control in the counting house area and the long distance serial link protocol, SPECS, for

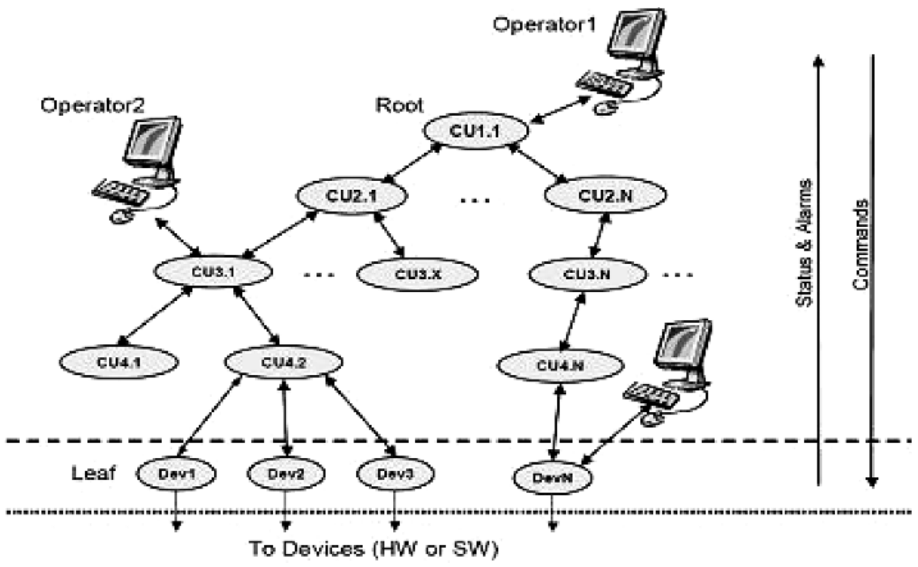


Fig. 10. Demonstration of a hierarchical tree using control units for an intuive grouping and device units as the leaves acting on the hardware. Operator access (e.g., sending commands or viewing the status) can be granted already at device unit level, any control unit or at the top from the Run Control. Partitioning of the system is possible.

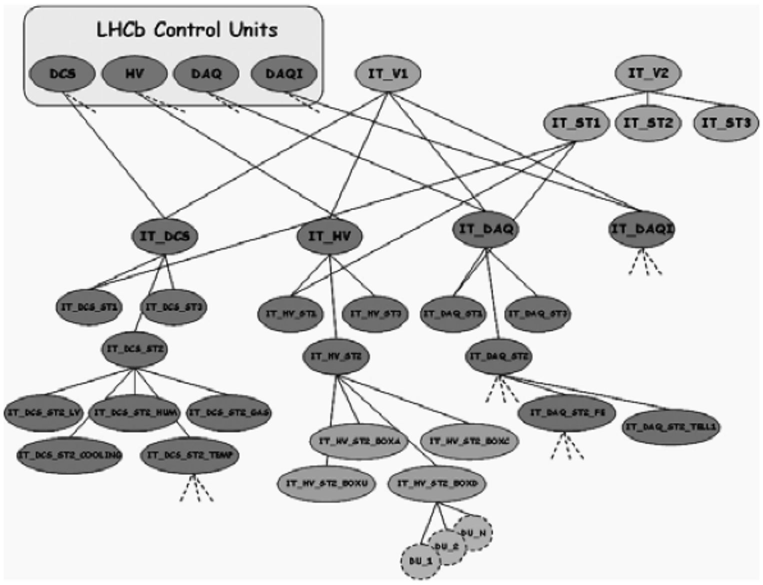


Fig. 11. Schematics of a hierarchical tree of a complete sub-detector. In this case the Inner Tracker.

accessing hardware inside the radiation area. Both choices have proved to be robust and have been extensively tested on various occasions. A well structured ensemble of software layers has been developed which disentangles the hardware layer from the control system. The entire experiment is modeled as a hierarchical tree using finite state machines which allows to operate the whole experiment in an intuitive way, even by non experts.

References

[1] "ERN Joint Controls Project, JCOP." [Online]. Available: http://cern.ch/itco/Projects-Services/JCOP/welcome.html, http://cern.ch/it-cobe/Projects/Framework/welcome.html

[2] D. Breton, D. Charlet, P. Robbe, and I. Videau, "PECS—A Serial Protocol for the Experiment Control System of LHCb," Oct. 2005.

[3] F. Fontanelli, G. Mini, M. Sannino, Z. Guzik, R. Jacobsson, B. Jost, and N. Neufeld, "Embedded controllers for local board control," *IEEE Trans. Nucl. Sci.*, vol. 53, Jun. 2006.

[4] F. Fontanelli, B. Jost, G. Mini, N. Neufeld, R. Abdel-Rahman, K. Rolli, and M. Sannino, "CCPC gluecard application and user's guide," *LHCb 2003-098/LPHE 2005-010*, Jun. 2003.

[5] C. Gaspar, M. Dönszelmann, and Ph. Charpentiere, "DIM, a portable, light weight package for information publishing, data transfer and inter-procss communication," presented at the Int. Conf. Computing in High Energy and Nuclear Physics, Padova, Italy, Feb. 1–11, 2000.

[6] A. Daneels and W. Salter, Selection and Evaluation of Commercial SCADA Systems for the Controls of the CERN LHC Experiments. Geneva, Switzerland [Online]. Available: http://www.elettra.trieste.it/ICALEPCS99/proceedings/papers/ta2o01.pdf

[7] PVSS II [Online]. Available: http://www.pvss.com

[8] P. C. Burkimsher, *JCOP Experience with a commercial SCADA product, PVSS*. Geneva, Switzerland: CERN [Online]. Available: http://cern.ch/itcobe/Services/Pvss/ScadaLab/ConferencesPresentationsEtc/2003Icalepcs/JCOPExperienceWithACommercialScadaProductFinal.pdf

[9] B. Franek and C. Gaspar:, "SMI++—An object oriented Framework for designing distributed control systems," *IEEE Trans. Nucl. Sci.*, vol. 47, pp. 86–90, Apr. 2000.

[10] D. Charlet and F. Machefert, "Calorimeter and SPECS Component Irradiation at PSI," Jun. 2005, LHCb-note, LHCb-2005-046.

[11] J. Christiansen, A. Marchioro, P. Moreira, and T. Toifl, TTCrx Reference Manual [Online]. Available: http://ttc.web.cern.ch/TTC/TTCrx_manual3.10.pdf

[12] G. Magazzu, A. Marchioro, and P. Moreira, DCU User Guide [Online]. Available: http://cmstrackercontrol.web.cern.ch/CMSTrackerControl/documents/Magazzu/DCU2_User_Manual_v2.pdf CERN

[13] Scientific Linux [Online]. Available: http://www.scientificlinux.org

[14] S. Koestner, PVSS/TELL1 Framework Components [Online]. Available: http://lhcb-online.web.cern.ch/lhcb-online/ecs/PVSS_TELL1/

[15] C. Gaspar, B. Franek, R. Jacobsson, B. Jost, S. Morlini, N. Neufeld, and P. Vannerem, "An integrated experiment control system, architecture and benefits: The LHCb approach," *IEEE Trans. Nucl. Sci.* vol. 51, no. 3, Jun. 2004 [Online]. Available: http://lhcb-online.web.cern.ch/lhcb-online/ecs/pdf/RT_2003.pdf, 513