Control and Operation of the LHCb Readout Boards Using Embedded Microcontrollers and the PVSS II SCADA System

S.Köstner[1] on behalf of the LHCb Online Project

***Abstract*— LHCb is one of the four experiments at the Large Hadron Collider. Before final reconstruction of the data in PC farms, high speed preprocessing is performed on a set of a few hundred custom electronics boards employing large modern field programmable gate array (FPGA) driven electronics. The local control of these boards is achieved via an embedded microcontroller which is connected to a large Local Area Network. After a brief introduction to the hardware we summarize the implementation of the entire layered software architecture for the readout boards and its integration into the Experiment Control System, which is built upon a common control framework based on an industrial SCADA system. Abstraction of different access modes and separation from the modeling of the components in the control system allow the reuse of various components on different hardware types. Each board has several hundreds of registers and memory blocks, so the optimization of write and read accesses is crucial for the start up configuration of the experiment. To facilitate the control of a large number of readout boards, finite state machines are introduced, where the states and transitions are well defined over the whole set of used boards.**

***Index Terms*— micro controller, PVSSII, SCADA, experiment control system, LHCb, TELL1**

LHCb-PROC-2006-035
04/11/2006

I. Introduction

IN the LHCb experiment [1] there will be 15 different types of electronics DAQ and trigger boards, in total some 400. A common readout board, the TELL1 [2] board, based on FPGA technology to adopt to the various specific requirements of each subdetector is used. Each of these boards will have a few hundred registers to be monitored continuously. In addition there are several memory blocks with some hundred megabytes for look up tables or similar functions to be uploaded in short time. LHCb has chosen to equip these boards with a micro controller which is accessed via a dedicated Local Area Network. The advantage of this approach is that each board has its own control path and the intelligence is decentralized. Thus the bottleneck of a crate controller is avoided as well as the possibility that a badly behaving device can block the entire chain, which improves scalability and robustness of the entire system. The hardware is briefly introduced. The focus of this paper is the integration of the readout boards into the Experiment Control System and the description of the various software layers.

[1]CERN, Geneva, Switzerland

Corresponding author: Stefan.Koestner@cern.ch

Fig. 1. Top side of the SM520PCX embedded PC.

mds
August 13, 2002

II. On-Board Microcontoller

For local board control the SM520PCX from Digitallogic [3] is used, wich is a complete embedded PC [4], based on the i486 compatible AMD ELAN 520 micro controller as shown in Fig. 1. It is running under Linux with a frequency up to 133 MHz. To provide an interface to the on-board chips, a small glue card has been created which is built around a PLX PCI930 PCI bridge, JTAG and I2C controllers. It also includes a bus switch which electrically isolates the glue-card from the carrier boards, when the PC is rebooted. When reading from memories attached to the local bus from PCI, transfer performances better than 20 MB/s were achieved.

III. Experiment Control System

The Experiment Control System of LHCb is built upon a common control framework (JCOP) [5] based on the industrial SCADA system PVSS II [6]. It will handle the configuration, monitoring and operation of all experimental equipment under the various running conditions of the experiment. From the software point of view, the framework adopted a hierarchical, tree-like, structure to represent the structure of subdetectors, subsystems and hardware components. This tree is composed of two types of nodes: Device Units which are capable of accessing the hardware to which they correspond and Control

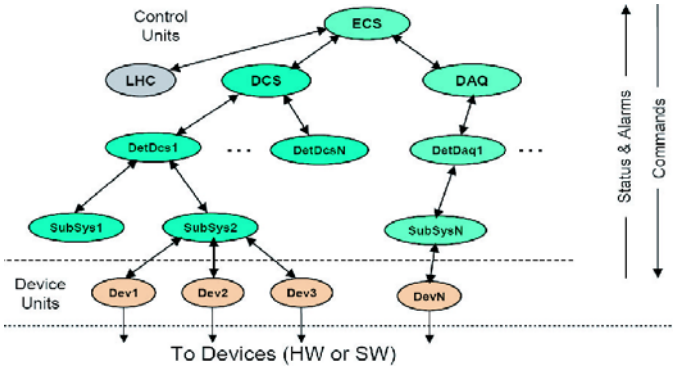

Fig. 2. The hierarchical architecture of the Experiment Control System (ECS) of the LHCb experiment.

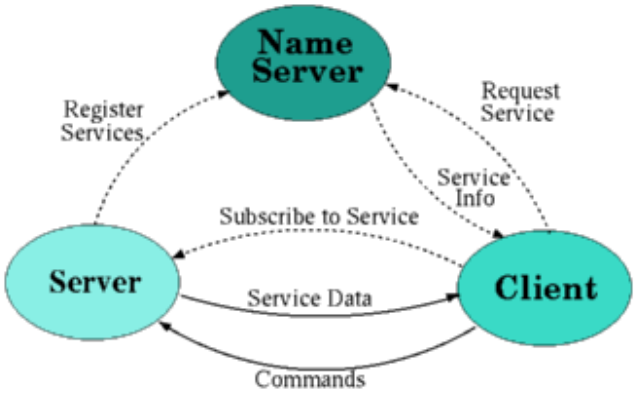

Fig. 3. The internal mechanism of the Distributed Information Management system.

Units which can monitor and control the subtree below them. These Control Units model the behaviour and the interactions between components. At the bottom of the tree there are the devices to be controlled (e.g. readout boards), these are grouped into subsystems, then onto subdetectors. Subdetectors are grouped by area of activity, DAQ or DCS and their states are combined with information received from external systems in order to arrive to a combined, decision making, top-level entity as shown in Fig 2. In LHCb the top of the hierarchy corresponds to the full experiment, allowing the user to have an integrated view of the experiment status and to interact with the different subsystems, the DCS, DAQ, etc.

In this hierarchy commands flow down and status and alarm information flow up. Control and device units are typically implemented using Finite State Machines (FSM), which is a technique for modeling the behaviour of a component using the states that it can occupy and the transitions that can take place between those states. SMI++ [7] has been integrated into PVSS for this purpose. Both PVSSII and SMI++ can run on mixed environments and allow for the implementation of large distributed decentralized systems, which is important due to the large scale of the system in terms of I/O channels in the order of millions.

The recovery from known error conditions can be automated using the hierarchical control tools based on subsystems states. Since SMI++ is also a rule-based system, errors can be handled and recovered by implementing rules like 'when system configured start run' and 'when system in error reset it'. In conjunction with the error recovery provided by SMI++ full use will be made of the powerful alarm handling tools provided by PVSSII.

IV. Implementation of the Readout Boards

Communication between the electronics boards and the various clients of the control system is achieved using the DIM (Distributed Information Management system) protocol [8], which is a portable lightweight publish/subscribe system. A DIM server has been written running on the embedded microcontroller of the electronics boards. This server is generic and performs all the required actions on the various different boards e.g. write and read operations on registers and memory blocks, FPGA programing, monitoring of registers, etc. The server is started automatically and publishes several DIM services, which can be single items, arrays or structured items on the DIM Name Server (DNS) who keeps their coordinates and from where they can be requested by the clients. Services can be subscribed to either on change or on a time basis. In addition commands can be send from the client to the server. Both, services and commands, can invoke a callback function. A usual procedure is that a command, requesting a read or write operation, is sent to the server where in the callback function the interaction with the hardware is executed and the data is sent back to the client by updating a service calling again a callback function on the client side. The basic mechanism is demonstrated graphically in Fig 3. The advantage of this design is its portability as clients can be installed on any machine by just specifiying the DNS node. It is robust as a crashed server can easily republish its services on the DNS node.

The registers of each building block on the readout boards are bundled and abstracted as a Datapoint Type, which is the internal representation of the SCADA system. Datapoint Types are complex structures from which datapoints can be derived which are connected to the PVSS internal memory data base. 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. Each board is represented as an instantiation of a datapoint type, which reflects the entire state of all controllable resources on the board. A special PVSS API manager allows associating DIM commands and DIM services to data points. PVSS allows also associating event triggered functions with data points. Whenever the content of a data point element changes, a function may be called in a PVSS script or API manager to perform a set of actions associated with the change. 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

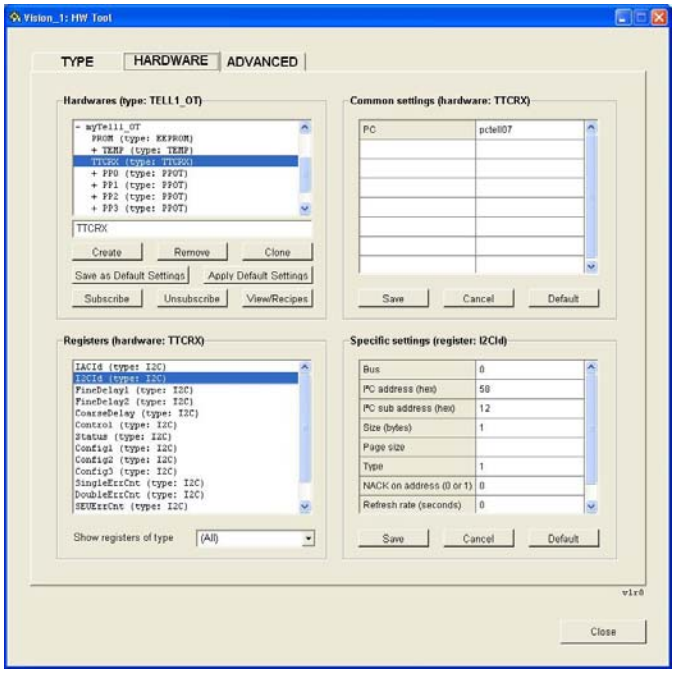

Fig. 4. Look and feel of the FwHw tool acting as an abstraction layer.

from the configuration database. In addition a masked write access which allows to modify specific bit fields is provided.

In order to facilitate the modelling of hardware as PVSS datapoints a tool was introduced, FwHw [9]. This tool offers a graphical user interface allowing to assign crucial information like bus address or register length to the various register types. The tool atomatically creates a well defined datapoint structure connected to the DIM API manager and sends as well instruction to the server to store the register's information in a list. Upon subscription of the registers the server publishes the appropriate services and is ready to receive commands establishing the communication between server and client. The server treats each register in the appropriate manner thus allowing to hide diversity and complexity of the various hardware and bus types. 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. If a register content changes on the hardware a DIM service can be launched updating the appropriate datapoint. Thus polling at fixed rates can be avoided and traffic on the local area network is kept at a minimum. In addition the tool allows for defining recipies. Recipies are a bundle 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.

To allow for further abstraction in an intuitive way the readout board is modelled as a finite state machine being a device unit in the sense of the control system's hierarchical structure as described above. Each board can find itself in predefined states, as shown in Fig. 5, from which they can transit into another. A state transition of a device unit can be

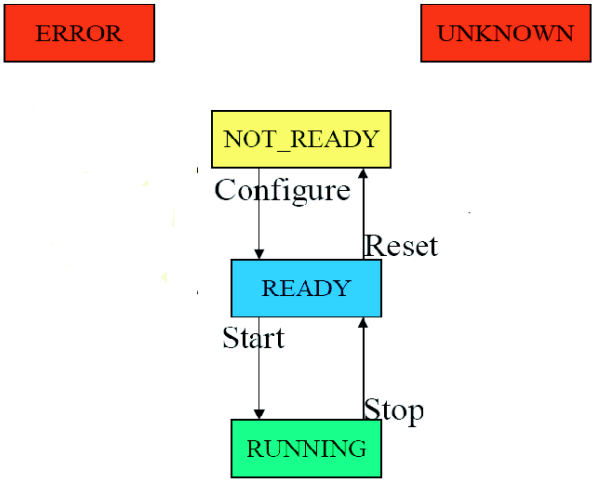

Fig. 5. Transition scheme as defined for finite state machines of type DAQ.

directly triggered by the hardware if some datapoints connected to real registers are changing. If a device unit goes into an error state it can automatically try to recover. The main operations are to download the firmware to the FPGAs, which is allowed in state NOT READY. The main configuration of the board, basically downloading recipes from the configuration database happens from state NOT READY to READY. The content of the recipes can differ according to the run type which can be specified as parameter when launching the command for configuration. Little action is required from state READY to RUNNING. The state UNKNOWN is accessible from all states and basically defines the state when control is lost e.g. DIM server has crashed. Also the state ERROR is accessible from all states. However an automatic recovery can be launched allowing to transit to any other state upon success. Device units can also offer a graphical user interface to allow users to interact directly with the hardware or to obtain more precise information about the status of the board or retrieving the current values of crucial registers being monitored. In Fig. 6 a screenshot of such an interface as being used for the TELL1 board is shown.

V. Conclusion

LHCb has chosen to use embedded processors with an isolated access path for board control. This choice has proven to be a robust solution and has been extensively under test on various occasions. A well structured ensemble of software layers has been developed to integrate the control of the readout boards. The implementation up to the expert system level is well advanced and only minor improvements are to be expected. The system will be ready for full scale running at the beginning of 2007 when the detector integration and commissioning will start.

Acknowledgment

I would like to thank the whole LHCb Online Group for their always immediate response and support as well as their

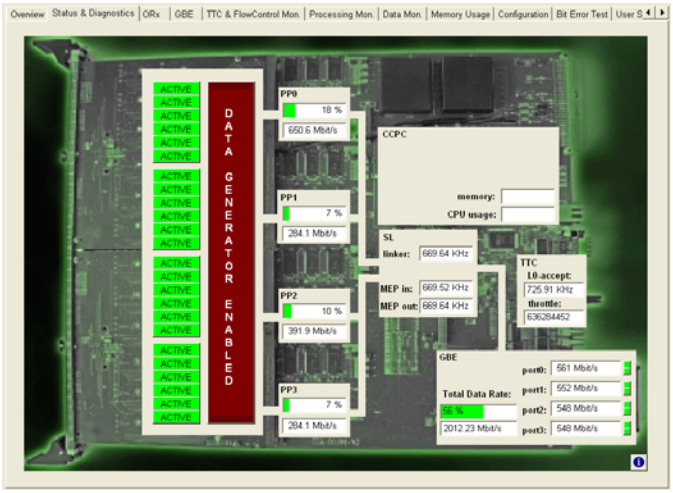

Fig. 6. Screenshot of a user interface as provided by a TELL1 device unit.

dedication and enthusiasm which made this work possible. Thanks must go also to G. Haefeli and C. Potterat for their advice and help in integrating and modelling the Tell1 readout board.

REFERENCES

[1] LHCb Collaboration: *LHCb Reoptimized Detector, Design and Performance*, CERN/LHCC 2003-030.
[2] G. Haefeli, A. Bay, A. Gong, M. Muecke, N. Neufeld, O. Schneider: *The LHCb DAQ interface board*, NIM, A560, (2006), 494-502.
[3] Digitallogic, Switzerland: *http://www.digitallogic.ch*
[4] 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, Num3, June 2006.
[5] A. Daneels and W. Salter: *The LHC experiments Joint COntrols Project, JCOP*, ICALEPCS, Triste 1999.
[6] PVSS: *http://www.pvss.com*
[7] B. Franek and C. Gaspar: *SMI++ - an object oriented Framework for designing distributed control systems*, IEEE Trans. Nucl. Sci., vol. 47, Num2, April 2000, pp.86-90.
[8] C. Gaspar, M. Doñszelmann, Ph. Charpentier: *DIM, a Portable, Light Weight Package for Information Publishing, Data Transfer and Inter-process Communication*, Presented at: International Conference on Computing in High Energy and Nuclear Physics (Padova, Italy, 1-11 February 2000)
[9] FwHw: *http://lhcb-online.web.cern.ch/lhcb-online/ecs/FWHW/default.html*