EMBEDDED PCS FOR ELECTRONICS CONTROL IN LHCB

F. Fontanelli[1], B. Jost[2], G. Mini[1], N. Neufeld[2], M. Sannino[1]
[1]INFN, Genoa Italy, [2]CERN, Geneva, Switzerland,

ABSTRACT

LHCb has a large number of complex electronics boards, which need to be controlled, configured and monitored. Instead of using a bus-based solution like VME, LHCb has decided to equip each board with an embedded micro-controller card for this purpose. A commercial component based on an i386-compatible microcontroller from AMD has been chosen. This combines the advantages of commercial-off-the-shelf hardware with the widespread knowledge of the i386 family, cutting down developing needs and shortening the learning and support curve.

INTRODUCTION

Traditional board-control has used shared buses like VME or Fastbus over a backplane of a crate to control individual electronic boards. This approach has two intrinsic disadvantages:

1) Scalability: one crate-processor / controller must access all boards; there is no easy way to share the load.
2) Robustness: one faulty board can block access to all devices on the bus. This problem can be aggravated when the bus is not only used for control purposes but also for data acquisition, as is the case for many smaller setups albeit not for the upcoming large HEP experiments.

To address 1) one can localise the board-control and intelligence by equipping each board with a microprocessor. Many different types of processors exist, each with their respective pros and cons. We will discuss our choice in the next section. In this way the centralised crate processor is offloaded of most of its tasks, like high frequency polling, monitoring etc. The next logical step is to abandon the shared bus altogether, this addresses point 2) above. The individual microcontrollers are then connected directly to a local area network (LAN), in much the same manner as the crate-controller is in the traditional setup. An incidental benefit is that the backplane design of such a crate can be greatly simplified because it can be reduced to power distribution and possibly a few global control lines.

THE CREDIT-CARD PC

When selecting a microprocessor to embed it for board-control one is confronted with an enormous choice. In the embedded world RISC architectures like PowerPC and ARM are very popular, because of their very low power dissipation. Consequently these are found in many hand held devices such as MP3 players or cell-phones.

While open-source software is usually readily available for these devices, standardized distributions of major operating systems (OS) are usually not so well supported. Also, a microprocessor is not yet a complete computer; other devices such as a memory controller, LAN controller, possibly graphics are needed. In the embedded world there are usually no standards for the choice and combination of these components, so the OS will have to be tailored to the complete system.

In order to minimize the hardware and software development effort in LHCb we have therefore decided to use an embedded Intel IA32 compatible microcontroller. Moreover we do not put the microcontroller directly on the board but we use an embedded PC module designed by Digitallogic [1], Switzerland. This device, called SM520PCX, is a complete personal computer (PC) on a credit card (CC) size plug-in card, nicknamed Creditcard-PC (CCPC). Centered around the AMD ELAN 520 microcontroller [2], it includes an Ethernet controller by Intel, a standard memory module as used in portable computers, and can be equipped with many optional components, such as Universal Serial Bus (USB) controller, graphics adapter.

The ELAN520 is a i486 compatible processor running at up to 133 MHz, with all standard components of a PC and some additions useful in an embedded environment extra precise hardware timers, a hardware watch-dog for automatic reboot in case of an unresponsive system and sever General Purpose IO (GPIO) lines. The low-power (below 2 W) SM520PCX is only one of a family of pin-compatible embedded PCs from Digitallogic, which offers modules with processors up to a 700

MHz Pentium III. Plugged on a carrier-board they require only power and connections to the required physical connectors, in our case only the RJ45 connector for the Ethernet connection of the CCPC.

The SM520PCX comes with a Flash-Ramdisk, which however we do not use, because network boot and diskless operation over the Network File System (NFS) are very fast and ensure a coherent software environment throughout.

Interfacing to custom electronics boards

A PC is not ``naturally'' suited to directly control resources on an electronics board. Most custom chips in LHCb are accessed via I2C or JTAG (IEEE 1394). JTAG is also widely used for in-situ programming (ISP) of configuration devices such as EEPROMs and Flash-RAMs. For high-speed data-transfer and register access to FPGAs or memories these two methods are unsuitable. A parallel bus is needed. Classically PCs offer two choices, the aged ISA bus and the more modern PCI. ISA is disfavored nowadays in the PC world, and is also not very popular with electronics engineers, for quite a few reasons, not the least being that it is a non-multiplexed bus.

PCI on the other hand, due to its signaling, which relies on reflective coupling, has impractical trace length restrictions (our electronics boards are typically 9U x 400 mm). We have therefore chosen to provide a simple local parallel bus, where local here means not going out of the board.

Neither of these can be found on PC based micro-controllers, and also on other micro-controllers, their are usually only one or two I2C chains and only few and/or slow JTAG chains, while ISP in an environment where frequent reprogramming is necessary, definitely requires a high-speed JTAG chain.

We have therefore decided to build a small adapter or "glue-card" to create the required interfaces

$$C_B = \frac{q^3}{3\varepsilon_0 mc} = 3.54\mu e\mathrm{V/T}$$

and connect them via PCI. It also includes a bus-switch to electrically isolate automatically the CCPC from the carrier-board, when the PC is rebooted. The hardware details of this are described in [3].

SOFTWARE

The micro-controller and glue-card solution presented so far will be used to control, configure and monitor over 400 boards of approximately 15 different types. The research and development work around these boards is done in many laboratories across Europe. We have therefore tried to provide simple software distribution mechanism, which allows small easy installations for a few boards, as well as larger multi-server setups and at the same time guarantees that updates of software are centrally transmitted to all installations, so that problems are not only fixed where they occur, but also pre-emptively applied at other sites.

Embedded software

We have chosen Linux as the OS of the micro-controller. The main reason is the easy customizability of the kernel for embedded, disk-less operation. In some early applications in our collaboration however Microsoft Windows has also been used, which allowed using commercial software packages. This was very useful to speed up initial debugging and development, because no software had to be written. All required functionality has since been ported to Linux.

On top of a slightly customized kernel we run the standard CERN Linux distribution, which is derived from the joint Fermilab and CERN Scientific Linux project [4].

The custom software, drivers and user-libraries to access the resources on the system are packaged and distributed in the same manner as the base-system, using the standard Redhat package management system (RPM) [5]. This is described in the next section. Except for the device drivers no special software, cross-compiler or anything else not usually available on a standard Linux PC is required. Editing, compiling and debugging excepting of course accessing the actual board hardware, can be done on any Linux PC. Most developers run all development tools on a server, which is sharing the files with the embedded PC via NFS. This is much more comfortable because build times are much shorter and a graphical environment can be used, which might be somewhat heavy on a CCPC.

Obviously the header files for all dedicated CCPC software are not installed in the standard directories of the host PC. The packages are organised such, that it is sufficient to set up a single environment variable, which will be used by the Makefiles to pick up the header files and libraries.

Within certain limits the Makefiles even allow for running the development environment on a machine, which has a different Linux distribution installed. As long as the build tools are not too different, such a cross-build is perfectly possible. This is regularly done in several collaborating labs with Linux distributions such as Fedora or Suse.

The software is partitioned into several libraries, which mirror the main hardware resources of the CCPC. There are libraries to access the I2C buses, the JTAG chains and the local bus. There are also libraries to access basic components of the glue-card such as the bus-switch and the GPIOs and specialised libraries for the in-situ programming of various types of FPGAs and configuration devices. Care has been taken to implement these libraries as efficiently as possible, in order to fully exploit the maximum performance provided by the hardware. For example, internally all libraries use direct memory mapping of the hardware registers thus bypassing all operating system overheads. In this manner the performances in Table 1 can be achieved

Hardware Interface	
Local bus read / write	10 – 20 MB/s (depending on the clock of the local bus defined by the carrier board)
I2C read / write	40 kBit/s (maximum speed of I2C bus)
JTAG scans	2 MBit/s (in gated mode)
JTAG programming	4 min for an EPC16 (16 MBit FLASH Ram) including erasing and verification

Table 1: Performances of the LHCb CCPC

Management software

In a large collaboration it is necessary to disseminate the software quickly and transparently to all teams. Only in this manner can effective peer-help be achieved, because one can rely on a common software base. In our case this task is made more difficult because the teams are geographically very widely distributed. There is thus no common intranet, like in some large companies. A shared NFS server for the software for example, while from a performance point of view possible, is not an option in the age of firewalls, net-filters and the like.

We have based our software distribution therefore on local installations, which are synchronised using HTTP and a simple automatic update mechanism. This requires a minimal administration effort by the users and can be even fully automatised.

In this manner all security updates and software patches, which concern packages not directly managed by us - and this is the vast majority - benefit from the effort of the respective support teams. It is thus not riskier to put any of the embedded controllers on the LAN or Internet, than it is to put a standard desktop PC running Scientific Linux. This allows direct remote accessibility of the CCPCs and hence the carrier-boards, a fact that greatly eases remote, rapid debugging and support.

The system relies only on a (central) web-server and a few simple scripts, which are documented and available at our web server [6]. The high level package-management of the RPM packages is done by the powerful YUM [7] tool, which in particular adds automatic resolution of package dependencies to the RPM functionality. YUM is a good candidate to replace the current default package manager on Scientific Linux, apt.

High-level software

At a higher level all control functionality in LHCb is integrated in the Experiment Control System (ECS), which is based on a commercial SCADA system, PVSS. This system provides alarm-handling, finite-state machine modelling, graphical user interfaces and the like. More details on the interfacing of the embedded software to PVSS and the ECS are described in [8].

CONTROLLING THE TELL1 BOARD

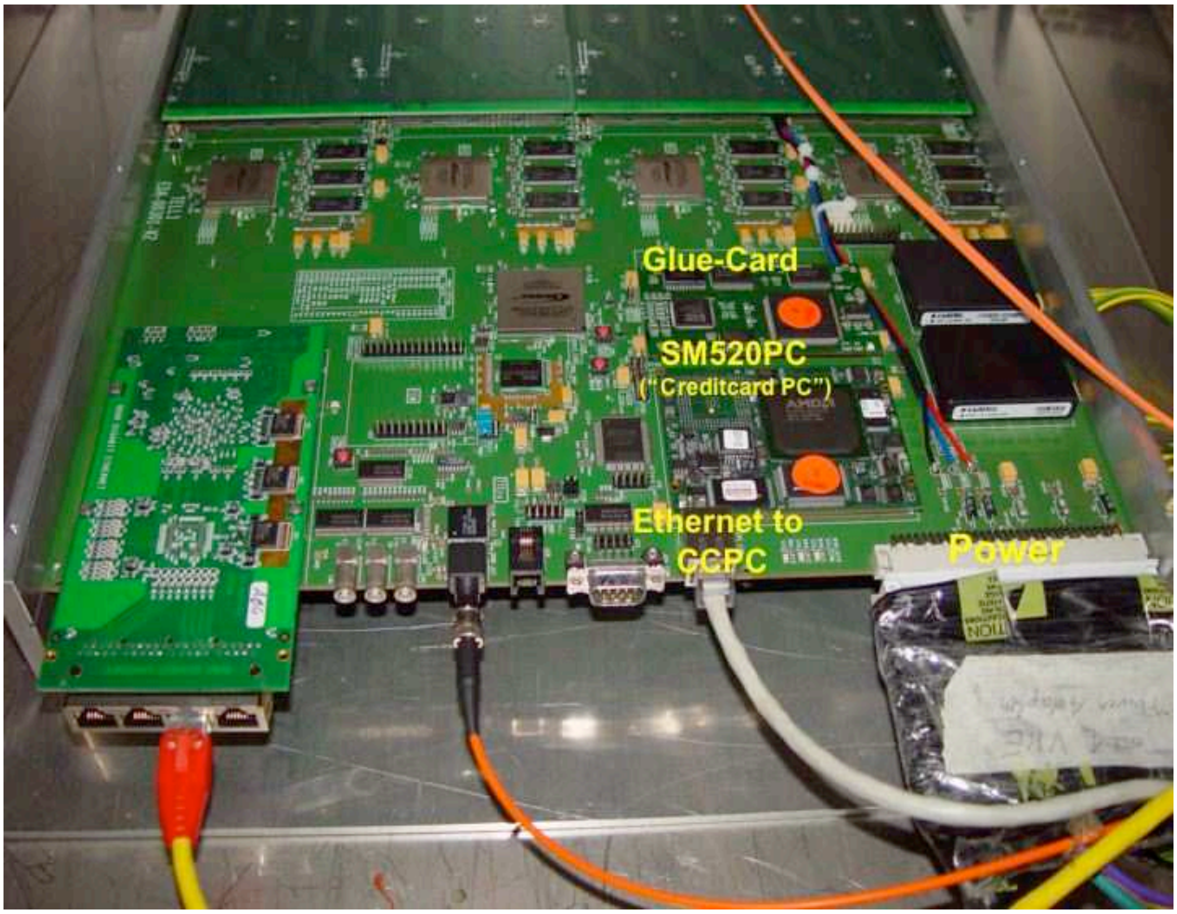

Figure 1: TELL1 readout board with CCPC and glue-card and the LAN connection of CCPC

The TELL1 board is the main readout board of LHCb, used by almost all sub-detectors. It consists of a main board equipped with several large FPGAs for preprocessing and formatting of the input data, large amount of memory for buffering, timing and trigger decoding logic. It also contains several daughter cards: analogue or digital receivers and a Gigabit Ethernet card to send the data to the DAQ system.

The hardware as well as the FPGA firmware of this board is configured and monitored via several 100 registers, moreover there Flash-memories for the firmware and lots of chips which are configured via I2C. All these registers are accessed via a CCPC shown in the left of Figure 1.

At start-up a bus-switch will galvanically separate the CCPC from the board, the CCPC boots from the network configures the memory map to access the board and starts a server process, which makes all registers available to the control system. Configuration is then done under the direction of this system. During running the CCPC will monitor registers either interrupt driven or by polling periodically. Changes will be reported, freeing the higher level from the task of detailed supervision at high rate. Also the CCPC is available for interactive login to run specialised debugging tools.

DEPLOYMENT IN LHCB

Installation in LHCb has not yet started, so here we can only give a report on the current plans for installation. There are some 400 CCPCs in LHCb. Almost every sub-detector of LHCb has boards, which use the CCPC. To provide for smooth and parallel operation and commissioning we attempt to avoid sharing of hardware resources between different sub-detectors wherever possible. This can be seen as one aspect of "partitioning". The hardware resource in question is the Control-PC, responsible for booting the CCPCs and running the PVSS project driving the actual board control during operation. Currently we think of not driving more than 20 CCPCs with one Control-PC. This leads to a total number of approximately 30 Control-PCs, because the various sub-detectors are quite unevenly populated with CCPCs.

The Control-PCs will be attached via Gigabit Ethernet to an Ethernet Switch to which also the CCPCs supervised by this Control-PC will be connected. The Control-PC and the CCPCs form a private LAN, which frees us from concerns about security and network broadcasts among others. The Control-PC will in turn be connected via a separate connection to the backbone LAN of the ECS. Direct access to the CCPCs is not possible only via the Control-PC, which acts as an application gateway in the sense of CNIC.

CONCLUSIONS

We have currently experience of running the CCPC-based board-control on systems ranging from a single board on a table for debugging to a full crate of 15 TELL1 boards in parallel – and the size of the system is now growing rapidly. It proved to be very useful that one can simply log into a board like into any standard PC. It is easy to write small, local test programs and later export the most useful functionality to the general control system of LHCb. Local command line based monitoring and centralized control and supervision exist well together.

From a hardware point of view our choice proved to be cheap and robust. There were no hardware problems with any of the over 50 CCPCs in heavy daily use so far. The fact that the CCPC is a commercially available plug-able daughter-card cuts short the design time of boards, at the comparatively small price of having to deal with a physically relatively large building block in the design.

We have no doubt that the full deployment of all CCPCs and their servers by the end of next year and their subsequent operation for several years in LHCb will be smooth, efficient and free of problems.

REFERENCES

[1] Digitallogic, Switzerland, http://www.digitallogic.ch
[2] AMD ELAN 520. http://www.amd.com/usen/ConnectivitySolutions/ProductInformation/0,,50_2330_8634_8635,00.html
[3] F. Fontanelli et al, "Interfacing Credit Card-sized PCs to Board Level Electronics", these proceedings
[4] Scientific Linux, http://www.scientificlinux.org
[5] Redhat Package Manager, http://www.rpm.org
[6] LHCb Online web, http://cern.ch/lhcb-online
[7] Yellowdog Updater Modified YUM, http://linux.duke.edu/projects/yum/
[8] R. Jacobsson, "Controlling Electronics Boards in LHCb with PVSS", these proceedings