

# Distributed control system for TRIDAQ boards

(published: Proceedings of SPIE, 2003, vol. 5125, session 3)

Wojciech M. Zabolotny<sup>a</sup>, Krzysztof Pozniak<sup>a</sup>, Ryszard Romaniuk<sup>a</sup>, Marcin Bartoszek<sup>a</sup>,  
Michal Pietrusinski<sup>b</sup>, Ignacy Kudla<sup>b</sup>, Krzysztof Kierzkowski<sup>b</sup>, Grzegorz Wrochna<sup>c</sup>, Jan Krolikowski<sup>b</sup>

<sup>a</sup>Institute of Electronic Systems, Warsaw University of Technology,  
ul. Nowowiejska 15/19, 00-665 Warszawa, Poland

<sup>b</sup>Institute of Experimental Physics, Warsaw University, ul. Hoza 69, 00-681 Warszawa, Poland

<sup>c</sup>Soltan Institute for Nuclear Studies, ul. Hoza 69, 00-681 Warszawa, Poland

## ABSTRACT

This paper presents a distributed control system developed for **TriDAQ** boards prepared for Resistive Plate Chamber (RPC) detector in Compact Muon Solenoid (CMS) Muon Trigger system to be used in Large Hadron Collider (LHC) in CERN.<sup>1</sup> The control system is based on Linux servers embedded in **TriDAQ** boards. The communication between embedded servers and managing/data processing host is assured by the Ethernet network. The embedded controllers offer sufficient computational power to perform board diagnostics and preprocessing of acquired data. Presented solution assures both high flexibility (due to configuration by the network) and high performance (due to highly parallel architecture).

**Keywords:** Embedded systems, Linux, CERN, LHC, CMS, Data Acquisition Systems, Distributed Systems

## 1. INTRODUCTION

The **TriDAQ** (Trigger and Data Acquisition) boards are important components of the RPC Trigger system of the CMS detector, which is being prepared for LHC experiment in CERN.<sup>1</sup> These complex electronic systems contain many FPGA circuits and require sophisticated control and diagnostics to work reliably. Management of **TriDAQ** boards includes downloading of FPGA circuits, testing of onboard connections and hardware using the Boundary Scan (BS) technology, and receiving and processing of the acquired data.

The first proposed solution was based on VME interface - the single VME host should service 17 **TriDAQ** boards. The shortage of this architecture was high load of VME bus and host.

The advance of technology enabled us to propose another solution, where the embedded PC-compatible controllers are used for management of **TriDAQ** boards. They can take over many functionalities of the managing host, and working in parallel can improve performance of the whole system. Instead of complex VME bus, simple network connections will be used to connect all **TriDAQ** boards with the managing computer.

## 2. SELECTION OF SYSTEM'S COMPONENTS

Development of such distributed control system was associated with many problems. Main of them were:

- Choice of the proper network solution
- Choice of the proper hardware platform
- Choice of the proper operating system

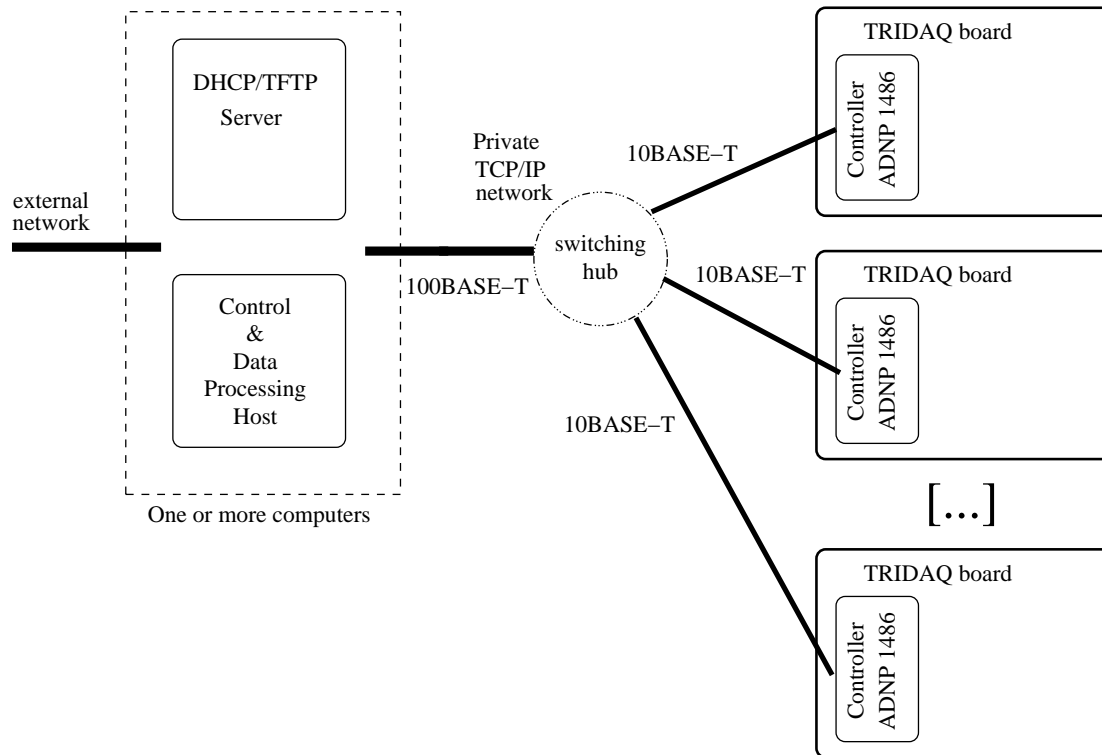
All these problems are tightly connected, and eg. choice of the network solution places limitations on the available hardware platforms.

---

Further author information: (Send correspondence to Wojciech Zabolotny or Ignacy Kudla)

Wojciech Zabolotny: E-mail: wzab@ise.pw.edu.pl, Telephone: +48 22 6607717, Fax: +48 22 8252300

Ignacy Kudla: E-mail: kudla@fuw.edu.pl, Telephone: +48 22 8242346



**Figure 1.** Block diagram of **TriDAQ** boards' control system

## 2.1. Network architecture for distributed control system

To assure simple and reliable network connections between all **TriDAQ** boards and managing hosts we have decided to use the Ethernet based TCP/IP network. We have considered following properties of such solution:

- Well established, stable standard
- Scalable solution - different speeds, ranging from 10 Mb/s to 1Gb/s available
- Availability of many implementations of TCP/IP based protocols

Finally we have decided to use the 10BASE-T Ethernet to connect **TriDAQ** boards to the switching hub and 100BASE-T Ethernet to connect hub with managing host (where the load is much higher).

Availability of remote boot/configuration protocols like DHCP/TFTP allows us to improve the flexibility of the system. The firmware for controllers and their configuration may be kept and managed centrally, on the managing host (server). It is especially useful in the development phase, because upgrade of the firmware consists only in change of files on the server's disk. The final architecture of proposed network is shown in the Fig. 1.

## 2.2. Choice of the hardware platform

The essential question to be solved is the choice of the hardware platform for our embedded controller. When choosing it we took in consideration following aspects:

**Size** The embedded controller should be small and flat enough to be conveniently placed on the printed board mounted in the CAMAC rack.

**Network compatibility** The embedded controller should be Ethernet and TCP/IP compatible, to allow easy interconnection between all **TriDAQ** boards

**Computational power** The embedded controller should offer enough computational power to allow autonomous performing of hardware tests and data processing - reducing the load of the managing host and network.

**Multitasking OS compatibility** To simplify the programming of the controller it should be able to work under control of multitasking (or even realtime) operating system.

**Availability of development tools** To allow effective development of the software, it is essential, that good compilers and debuggers are available for the chosen platform.

**Availability of external bus** - necessary to communicate with peripherals installed on **TriDAQ** board.

After overview of available hardware offers, we have selected five candidates for more thorough analysis.

**Ethernut** A simple ATmega 103 based board with 128 kB FLASH, 32 kB SRAM and 10BASE-T Ethernet interface<sup>2</sup>

**uCsim** A SIMM compatible board with Motorola DragonBall 68EZ328 processor, with 2 MB FLASH and 8MB DRAM, and 10BASE-T Ethernet interface<sup>3</sup>

**MZ-104+** An embedded PC with PC/104+ bus interface, dual 100/10BASE-T Ethernet interface and up to 64 MB of DRAM<sup>4</sup>

**Advantech CPC2245** An embedded PC with 32 MB DRAM, SVGA interface, 10/100Base-T Ethernet interface, and a CompactFlash solid state disk socket<sup>5</sup>

**ADNP 1486** An embedded PC with 128-pin QIL compatible footprint, equipped with 16 MB DRAM, 4 MB FLASH, Elan SC410 CPU and 10BASE-T Ethernet interface<sup>6</sup>

First of them was rejected inspite of very low price because of lack of mature operating system, and low computational power. The second one didn't allow to implement an external bus. The third and fourth ones because of implementation of almost all functionalities of standard PC were too big and required too much power. The finally chosen hardware - ADNP 1486 was selected because of the following advantages:

1. Easy connection to the motherboard - the whole module imitates an integrated circuit in QIL 128 package.
2. Very small power consumption (ca. 0.3A at 3.3V) (no need for active cooling)
3. The reasonable amount of RAM already on board (16 MB)
4. The flashdisk already on board (4 MB)
5. Preinstalled ROM DOS with serial console support (convenient for hardware debugging), possibility of Linux installation (with or without net booting)
6. Almost full ISA bus available at the outer pins
7. PC-compatible - many operating systems and software tools available
8. 486-compatible AMD Elan CPU working with 100 MHz clock offers sufficient computational power
9. Relatively low price (550 DM)

### 2.3. Choice of the operating system

The proper operating system for our embedded controllers has to meet many requirements

1. Stable
2. Well integrated TCP/IP network services
3. Minimal costs (preferably free, no royalties)
4. Open Source (to allow possible slight modifications)
5. Good availability of development tools
6. Ability to run on limited hardware resources

As a system matching these requirements we have chosen GNU/Linux, more exactly the Etlinux distribution<sup>7</sup> with following features:

- Very small RAM requirements (minimum 2MB) leaving more memory for data and applications
- Extended Tcl shell, allowing to implement many utilities as the Tcl scripts.
- Very small size of operating system image

However to allow for using Etlinux in our control system it was necessary to modify it slightly. We had to modify kernel 2.0.38, adding the serial console support. Additionally we had to add some additional programs (eg. `dhcpcd` and `rpc.portmap`) to support necessary network protocols.

## 3. INTEGRATION OF HARDWARE WITH OPERATING SYSTEM

In such control system the essential question is the proper communication between operating system and the controlled hardware. It is very important that data transfers are performed fast and with minimal CPU load. Another important aspect is asynchronous notification of applications about the hardware events.

To allow safe and efficient application development a decision was made to implement as much functionalities as possible in the *user space*. It allows to use standard debugging tools, like `gdb` to isolate and eliminate the problems, and decreases possibility that misbehaving application crashes the system.

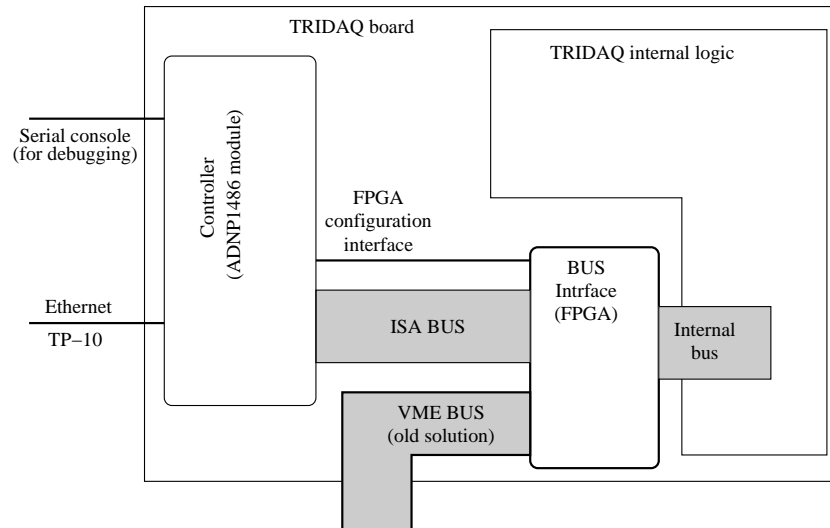
### 3.1. Asynchronous notification about hardware events

The above solution however creates problems with hardware events notification, because hardware interrupts in Linux must not be serviced in user applications. The natural implementation seems to be use of *signals*, but their use in an application is still limited due to asynchronous execution of signal handler. Finally another solution was chosen, where for each hardware device requiring asynchronous notifications, a dedicated *character status device* was created. The application thread servicing the asynchronous notifications may use the *blocking read* or *select* function on the *status device* to remain sleeping until hardware event happens; then it reads the single byte from the device driver, informing about the cause of the interrupt, and may proceed with event's servicing.

This solution assures minimal latencies when servicing interrupts in user space. Certainly *kernel space* interrupt service routines are still required for handling of time critical interrupts (eg. when service routine must not get preempted).

### 3.2. Providing user space access for hardware registers

Similarly *user space* applications are allowed to access hardware registers directly, to allow fast data transfers. A POSIX *mmap* mechanism has been used to achieve it. All the hardware registers are mapped in the memory address space. The application may use the *mmap* function to obtain a pointer for virtual memory area, where hardware registers are available. It allows to use very efficient programming strategies, eg. the C language structures may be used to intelligibly reflect meaning of particular registers and to provide convenient access to them. It is possible to use CPU's protection mechanisms to detect memory access bugs, however it requires that memory assigned for each device is page aligned (4 KB).



**Figure 2.** Simplified block diagram of **TriDAQ** board

### 3.3. Operating system drivers

To simplify development of device drivers for all still being developed peripherals on **TriDAQ** boards, a *skeleton driver* has been prepared. The *skeleton driver* implements the basic functions of each device like: *status device* and *select* function used for asynchronous notifications, the *mmap* function, and both *open* and *close* functions. To add a new hardware device, a developer should only write functions responsible for device initialization, and for interrupt servicing (the rudimentary interrupt handling necessary before transferring control to the user space as described in section 3.1).

### 3.4. Bus interface

The ADNP1486 module uses the ISA bus to communicate with the external world, while the **TriDAQ** boards were designed to be connected to the VME interface, and therefore their internal bus is rather VME compatible. To interface ADNP1486 with the internal bus of **TriDAQ** board a dedicated IP core was created in VHDL language to be implemented in a FPGA chip. This *bus interface* translates the ISA read/write cycles to the internal interface read/write cycles.

### 3.5. FPGA configuration interface

The **TriDAQ** board contains many programmable chips which require configuration during the powerup of the system. Most of them are configured through a special FPGA chip connected to the internal interface, however other method is necessary to configure the *bus interface* and make internal interface available. This is a task to be performed by our controller. Therefore some of its *General Purpose I/O pins* (GPIO) have been used as an interface for configuration of Altera FPGA's in the *passive serial mode*. Additionally a dedicated device driver has been written which allows for configuring of Altera FPGA connected to this *FPGA configuration interface* just by sending (eg. with `cat` command) the standard Altera RBF file to the associated `/dev/altconfelan` device.

## 4. COMMUNICATION PROTOCOLS

The whole system uses different network protocols. Some of them are used during the startup of the system, and the others are used during the normal work, to establish communication between the managing host and controllers of **TriDAQ** boards. The very important issue in network systems is their security. Usually special, cryptographically protected protocols are used for that purpose. In our case all the system creates a small, closed network. Only the managing host (or hosts) provides connection to the external network, which should be firewall protected. Thanks to such structure we may internally use simple unsafe protocols (like TFTP, RPC and so on) to achieve the highest possible performance.

#### 4.1. Protocols used during the startup of the system

During the startup of the system DHCP and TFTP protocols are used to provide all the board controllers with their firmware and configuration information. An etherboot software suite<sup>8</sup> was used to create the loader program and the operating system image. The operating system is loaded through the network and then runs from the ramdisk.

Such solution assures both high performance and high flexibility. During the development it is easy to provide new firmware, because it is stored on the central server. Additionally it is safe, because the power failure does not lead to controllers' filesystem corruption.

In very specific cases, eg. when additional, more complex software should be used for diagnostic purposes, it is possible to provide NFS mounted directories, or even to use a dedicated operating system image running with NFS mounted root filesystem. We didn't decide to use NFS in the standard working mode, because it would unacceptably increase network load.

Yet another problem may occur during the startup of the system - when all the board controllers try to connect to DHCP/TFTP server, the network may get overloaded, and because of big amount of collisions its throughput may decrease much below the standard 10 Mb/s. To prevent this problem different delays has been introduced in the startup sequence of board controllers.

#### 4.2. Protocols used during the standard work of the system

All the board controllers use Linux as their operating system. However the managing host may use other software platform. To establish efficient network communication between applications running on different platforms we have decided to use the RPC protocol. It allows for convenient bidirectional communication between applications running on boards' controllers and on managing hosts, and is well suited to small memory/CPU resources available on boards' controllers.

For special diagnostic purposes there is a telnet daemon running on each board controller, which allows to run non-standard application and perform nonstandard test interactively.

### 5. TESTING OF THE SYSTEM

The CMS detector is still being developed, and the **TriDAQ** boards are still only prototypes. Therefore testing of the whole system was not possible yet. However all its components have been successfully tested with prototype **TriDAQ** board. The controller booted successfully with the server provided operating system image. The *FPGA configuration interface* worked correctly, and the *bus interface* assured the correct communication with **TriDAQ** peripherals. The device drivers' mechanisms for accessing of hardware registers and for providing asynchronous notifications have been verified to work correctly. Also the RPC based mechanisms for bidirectional interprocess network communication have been tested and verified successfully.

### 6. CONCLUSIONS

The presented control system is a reasonable alternative for standard bus based systems. It is especially well suited for applications, where the data preprocessing may significantly reduce the bus or network traffic, and where the parallel data processing may significantly improve performance of the system.

From the economical point of view that solution may be accepted in systems where the controller's price is only a small part of the whole board's cost. However the situation may change when the Ethernet enabled controllers become more popular and therefore cheaper.<sup>9</sup>

## REFERENCES

1. J. Krolikowski, G. Wrochna, M. Konecki, M. Kudla, A. Ranieri, E. Pietarinen, K. Banzuzi, K. Pozniak, and P. Zalewski, "RPC trigger," in *CMS, The Trigger and Data Acquisition project*, C. E. Board, ed., **I**, pp. 419–480, CERN, 2000. Also available as <http://cmsdoc.cern.ch/cms/TDR/TRIGGER-public/CMSTrigTDR.pdf>.
2. "Ethernut project - an open source software and hardware project for embedded ethernet applications." <http://www.ethernut.de>.
3. "uCsim hardware project." <http://www.uclinux.org/ucsimm>.
4. "PC/104+ ZFx86 SBC with dual ethernet." [http://www.tri-m.com/products/tri-m\\_engineering/mz104+.html](http://www.tri-m.com/products/tri-m_engineering/mz104+.html).
5. "ISA bus 486 mini biscuit PC with VGA/LAN." <http://www.advantech.com/products/CPC-2245.asp>.
6. "SBCs for embedded networking: DIL/NetPC ADNP/1486-3V." <http://www.dilnetpc.com/dnp0008.htm>.
7. "Etlinux documentation." <http://www.etlinux.org>.
8. "Etherboot home page." <http://www.etherboot.com>.
9. A. Stevenson, "High-availability systems bypass bus traffic," *EDN Magazine* **May 10**, p. 82, 2001.