PROCESS TO PROCESS COMMUNICATION OVER FASTBUS IN THE DATA ACQUISITION SYSTEM OF THE ALEPH TPC¹

A. Lusiani

CERN, Div. PPE, CH-1211 Genève 23, Switzerland and
Scuola Normale Superiore, piazza dei Cavalieri 7, I-56100 Pisa, Italy

Abstract

The data acquisition system of the ALEPH TPC includes a VAX/VMS computer cluster and 36 intelligent Fastbus modules (ALEPH TPPs) running the OS9 multitasking real-time operating system. Dedicated software has been written in order to reliably exchange information over Fastbus between the VAX/VMS cluster and the 36 TPPs to initialize and co-ordinate the microprocessors, and to monitor and debug their operation. The functionality and the performance of this software will be presented together with an overview of the applications that rely on it.

I. INTRODUCTION

Since August 1989 the ALEPH detector has been collecting data on e^+e^- collisions provided by the LEP storage ring at CERN.

The ALEPH data acquisition (DAQ) system [1,2] includes a large number of different microprocessor-based custom designed modules arranged in a "tree-like" read-out structure. These modules first process the data coming from the various parts of the detector in parallel and then build the ALEPH event out of all these pieces. Each module reads its data through Fastbus and is in turn read out through Fastbus by a module sitting in the next higher level of the tree. A VAX/VMS computer cluster supervises the operation of the microprocessors and receives the events which are finally delivered to the mass storage.

A significant part of the ALEPH DAQ system resources is devoted to the main tracking detector [3], a large time projection chamber (TPC) with about 50,000 channels, each of which produces 512 bytes of data per event. These digitized data are processed by 36 intelligent Fastbus modules named 'time projection processors' (TPPs [4]). The TPPs are based on the 68020 microprocessor running the OS9 multitasking real-time operating system; they make up more than one half of the total number of microprocessor-based custom designed modules in the ALEPH DAQ system.

Whereas the detector data flow through Fastbus from the digitizers to the VAX, the operation of the DAQ system requires also that initialization data and control messages flow in the opposite direction in order to initialize and steer all the digitizers and all the microprocessors.

According to a generally accepted scheme, ALEPH had planned to use Ethernet as a separate communication path for control information. However, a number of practical

constraints² at the time when the TPC read-out software was designed made it preferable to use again Fastbus as the communication medium for the management of the 36 TPPs.

Therefore, dedicated software has been developed in order to exchange information through Fastbus between the VAX/VMS cluster and the TPPs. Relying on this common software layer, a number of VAX applications have been written in order to accomplish all the tasks that are needed to centrally initialize and co-ordinate the TPPs and to monitor and debug their operation.

II. THE COMMUNICATION SOFTWARE

The ALEPH TPC DAQ system was designed to be fully managed from the VAX. It necessitates that proper software routines exist to initialize the TPPs, to download data and executable modules, and to start, stop and steer the TPP tasks. Moreover, it is desirable to have a way to retrieve information from the TPPs in order to get messages and other data from the microprocessor tasks and in order to monitor the remote operations.

All the required functionality can be achieved by using Fastbus as a communication medium. The interference with the Fastbus usage during data-taking is naturally minimal, since the system initialization and most of the system management take place outside data-taking. Furthermore, the Fastbus bandwidth can be exploited to minimize the system initialization time.

Two nodes in the VAX/VMS computer cluster are interfaced to Fastbus. The memory and the registers of the 36 TPPs can be accessed via Fastbus through their slave ports, which are connected to Fastbus as shown in fig. 1.

The TPP initialization is accomplished by accessing a dedicated Fastbus CSR register that resets the module. The required communication functionality is achieved as described in the following.

A first basic routine is able to send a message from a VAX process to a TPP process and, at the same time, can retrieve information on whether the message was received by the TPP and by the selected process. In the case of successful delivery, a reply from the destination process can also be returned.

A special application, the TPP server, runs on all the TPPs and is used to remotely perform the basic functions of the TPP operating system by means of messages addressed to it over Fastbus.

Finally, a library of routines has been written, which can get messages from the TPP tasks by accessing three dedicated memory buffers where the output stream of an

¹Presented at the "Eight Conference on Real-Time Computer Applications in Nuclear, Particle and Plasma Physics", Vancouver, June 8-11, 1993.

²E.g., the lack of a solid commercial implementation of communication software over Ethernet for OS9.

Figure 1: Schematic diagram showing the main hardware components involved in the communication. Two nodes of the ALEPH online VAX cluster are interfaced to Fastbus; 36 TPPs are connected to 4 Fastbus cable segments, each servicing 9 modules.

application can be redirected by means of a special device driver.

A. The basic message-passing routine

The VAX processes use the TFNLIB library [5] in order to send messages to the TPP processes. In the fundamental routine, the message content is combined with information on the sending and destination processes and then written through Fastbus in a dedicated memory area of the relevant TPP. The TPP hardware is able to generate an interrupt for any Fastbus write access on the TPP slave port. This interrupt is handled by a dedicated device driver, which is described elsewhere [6]. Here the data consistency is checked, and the body of the message is delivered to the specified TPP process. Either an OS9 signal or an OS9 event can be used in order to make asynchronous reception possible. Acknowledgement information is written in another dedicated memory area, which can also accommodate a reply from the TPP process. This area is polled through Fastbus within the same routine that originated the transaction until the TPP answer is ready. The VMS wait call is invoked before every read operation: the time to wait has been set such that generally the TPP answer is ready on the first poll. After a maximum number of polls without an answer a time-out error is returned.

By using the VMS cluster-wide lock system service, communication procedures cannot be interrupted by any other VAX process running in the cluster and communicating with the same TPP: in this way several VAX processes can communicate with the same TPP at the same time

without any interference.

The described procedure conveys a message from the VAX to a TPP in an asynchronous way. The success of the delivery is checked within the same routine that originates the transmission by means of polling; furthermore, a reply from the addressed task can be retrieved if it is available within the polling time.

A more elegant and flexible scheme would require an additional asynchronous message-passing mechanism from the TPP to the VAX in order to get reception acknowledgements and replies. Although such a mechanism could in principle be engineered by using Fastbus service requests, it was estimated that its benefits would not necessarily improve the performance of the applications envisaged for the management of the TPC DAQ system. In addition, further software overhead on the VAX would have been required in order to handle the Fastbus service requests, and the development of complex software protocols would have been needed as well in order to insure a reliable communication.

It should also be noted that the absence of full message buffering limits the suitability of this communication software for a general purpose usage; however, for the applications in the TPC DAQ system, this does not affect neither the performance nor the functionality.

B. The TPP server

The TPP server [6] is an application that is started on all TPPs during their bootstrap procedure. Several TFNLIB library routines interact with the TPP server via messages over Fastbus in order to remotely perform a number of remote operations from the VAX:

- OS9 commands can be executed remotely in the TPP;
- OS9 executable and data modules can be located, created, downloaded and uploaded;
- OS9 signals can be sent to OS9 processes;
- BOS banks³ in OS9 data modules can be located, downloaded and uploaded.

Large amounts of data can be quickly transferred through Fastbus: the TFNLIB library sends a message to the TPP server in order to locate the data position in the TPP memory. In this way the actual data transfers go directly from the TPP memory to the VAX memory and *vice versa*.

C. Messages from the TPP tasks to the VAX cluster

Warning and informational messages issued by the TPP tasks can be retrieved on the ALEPH online VAX cluster.

To this aim a special device driver has been written, to which any OS9 output stream in the TPP can be redirected. This device driver stores the messages in a memory buffer contained in an OS9 data module. When the buffer is full, the oldest messages are dropped as soon as the new ones arrive.

This device driver offers the possibility of using different OS9 data modules to buffer different streams of messages. In the TPP, three separate streams are used to collect respectively error, informational, and system messages.

³ ALEPH data are formatted in BOS banks inside the TPP. BOS is described in: V. Blobel, "The BOS System Dynamic memory management FORTRAN77 Version," Internal Report, DESY R1-88-01 January 1988.

Figure 2: Schematic diagram showing the hardware and software components involved in the communication between processes in the VAX cluster and TPPs in the TPC data acquisition system.

On the VAX, a library of routines has been written in order to return the TPP messages one at a time as they are issued. To obtain this, the TPP message buffers are repeatedly read out by means of the TNFLIB routines that upload OS9 modules.

Storing messages in the TPP memory and reading them out through Fastbus means that debugging can proceed without any serious degradation of the TPP performance.

For the TPC, the essential information on data acquisition failures is combined together with the data being acquired. Therefore, it is not necessary to insure that the buffers are read frequently enough not to lose any informational message. As a desirable side effect, the system reacts in a well behaved manner in the case of a flood of messages.

D. Reliability and robustness considerations

Special emphasis has been placed on the reliability and on the robustness of the communication:

- the data integrity is controlled by means of checksum words that are added to the data and checked at the destination;
- elaborate error handling and error reporting ensure that sufficient diagnostic information is conveyed to the user when the communication fails;
- whenever possible, any operation that fails is repeated a reasonable number of times, so that sporadic hardware failures can be recovered;

- information on the automatically recovered failures can be logged so that any degradation of the performance of the hardware can be identified.

III. APPLICATIONS USING THE COMMUNICATION SOFTWARE

A non exhaustive list of applications relying on the communication over Fastbus is presented in the following.

A. Initialization and steering

The ALEPH run controller steers the TPC initialization task which, in turn, uses the TFNLIB library to bootstrap the TPPs, to download executable and data modules, and to control the TPP data acquisition tasks. A total of about 5 Mbytes of data must be downloaded after power up. The whole initialization procedure, which requires a minor share of the VAX computing resources, takes 1 minute; this is considered to be fully satisfactory.

The TPPs are controlled through an additional layer of software, the TPP state manager library [7], which minimizes the transfer of data to the TPPs by remembering their state and by repeating the operations that failed only on the relevant TPPs.

B. Remote control via Fastbus

The TPPCOMM program can remotely execute OS9 commands through Fastbus and can retrieve their output on the user terminal. An application has been written to perform a complete test of the TPP hardware from any user terminal, through TPPCOMM.

C. Calibration and test programs

The about 50,000 analog channels of the ALEPH TPC are shaped, digitized, processed and buffered by 768 triple width Fastbus modules named TPDs [8], which are also in charge of equalizing the individual response functions by means of adjustable calibration constants.

The calculation of the TPD calibration constants is a major operation and is accomplished by a VAX application that uses the TFNLIB library in order to run 36 TPP tasks in parallel.

Similarly, another VAX application has been developed in order to thoroughly test the 768 TPDs so that their reliable operation during data acquisition can be certified.

D. Presentation of the TPP messages

The TPPLOG facility collects the TPP messages into a large buffer that can be browsed by the user at the terminal. Messages kept in one of the message buffers of the TPPs can be logged into a file by the TPPLOG.FILE facility.

A similar facility, TPPLOG.SOURCE, can be used to continuously log into a file all the messages produced by all TPPs in one stream. This file is updated one line at a time and can be browsed by the facility TPPVIEW while it is updated. In this mode of operation, a single VAX process reads the TPP messages, which can then be accessed by many different users of the ALEPH online cluster, even on the nodes without a Fastbus interface.

E. Monitoring of the TPP operations

By using the TFNLIB library, the TPPMON facility reads continuously a dedicated OS9 data module in the specified TPPs. In this way, a large amount of information on the data acquisition performance can be displayed for monitoring and debugging purposes.

IV. CONCLUSIONS

The ALEPH TPC data acquisition system relies on a set of dedicated software packages that allow exchange of information over Fastbus between the ALEPH online VAX cluster and 36 TPPs. A number of applications are successfully using this communication software in order to initialize, steer, monitor, calibrate and test the TPC data acquisition system. Adequate communication speed, robustness and reliability were achieved by using the available Fastbus bandwidth and a minor share of the VAX and TPP computing resources.

Although it should be noted that the absence of some features, like full message buffering, may limit the convenience of this approach in other situations, communication over Fastbus has provided an entirely appropriate solution for the case addressed in this paper.

V. REFERENCES

- [1] ALEPH collaboration, **The ALEPH Handbook** (CERN, April 1989);
D. Decamp et al., "The ALEPH Handbook," Nucl. Instr. and Meth. **A294**, 121 (1990).
- [2] ALEPH Dataflow Group, "ALEPH Data Acquisition System Hardware Functional Specifications," ALEPH internal note, DATAQ 85-21, CERN, 1985;
W. von Rüden, "The ALEPH data acquisition system," IEEE Trans. Nucl. Sci. **NS-36**, 1444 (1989).
- [3] S.R. Amendolia et al., "The FastBus Read-out System for the ALEPH Time Projection Chamber," IEEE Trans. Nucl. Sci. **NS-36.5**, 1514 (1989);
S.R. Amendolia et al., "Data acquisition for the ALEPH TPC: a case study for LEP," Nucl. Instr. and Meth. **A289**, 569 (1990).
- [4] S.R. Amendolia et al., "The MC68020-based Fast-Bus read-out processor for the ALEPH Time Projection Chamber," IEEE Trans. Nucl. Sci. **NS-34.1** 128 (1987).
- [5] A. Lusiani, "TFNLIB: TPP Fastbus Network Library," ALEPH internal note, ALEPH 93-075, TPC-GEN 93-03, CERN, 6 May 1993.
- [6] A. Lusiani, "TPP software for the communication with a VAX cluster through Fastbus," ALEPH internal note, ALEPH 93-074, TPCGEN 93-02, CERN, 6 May 1993.
- [7] B. LeClaire, "The TPP State Manager," ALEPH internal document, CERN, September 1989.
- [8] B. Lofstedt et al., "The Time Projection Digitizer Module," CERN-EP-EL F6821-5A4, 1986.