The DAQ System for CLEO III

A. Wolf [a,1], C. Gwon [a], K. Honscheid [a], J. Lorenc [a], R. Wanke [a], E. Lipeles [b], A. Shapiro [b], A. Weinstein [b], F. Würthwein [b], A. Bean [c], D. Coppage [c], C. Darling [c], B. Forrest [c], T. Noor [c], C. Strohman [d], V. Fadejev [e], J. Staeck [e], I. Volobouev [e]

[a] *The Ohio State University*

[b] *California Institute of Technology*

[c] *University of Kansas*

[d] *Cornell University*

[e] *Southern Methodist University*

In 1998 the CLEO detector collecting data at the CESR storage ring will be upgraded. A new Data Acquisition system will be installed during the upgrade to record triggers up to 1000 Hz with a mean data size of 25 KBytes per event.
In this largely improved system, we will exploit the new CBLT VME transfer mode (defined by VIPA) to readout crates with about 15 9U digitization cards. Secondly, we decided to design and build a FASTBUS-VME interface card with the dimension of a FASTBUS card, and a 2-slot VME backplane inside. With this interface, all hardware and software will be homogeneous at this point of the DAQ data pipeline. It will also perform additional sparsification and channel tagging "on the fly". This new system will be monitored and controlled by a networked system of CPUs. To achieve a dynamical and simple connection method between the various processes we chose the CORBA standard communication method. The port to PowerPC based VME boards was done in close collaboration with Visigenic Corp.

Key words: Data Acquisition; VME; VIPA; CBLT; Readout; FASTBUS; Interface; Slow Control; VxWorks; CORBA;

[1] corresponding author

1 Introduction

The design of the CLEO III data acquisition [1] system is guided by the need to reduce cost and complexity of both hardware and software, while achieving the required performance of 1000 Hz event rate. This is accomplished by implementing an integrated plan based as much as possible on *commercially available components* and by making use of *standards* wherever reasonable. We invested particular effort to make the CLEO III data acquisition uniform and homogeneous:

A standard VME slave interface [3] providing the first implementation of the MCST and CBLT operations (VME-P) [4] will be used by all VME data boards. Using a CLEO designed FASTBUS controller, we allow the FASTBUS and VME based systems to use identical read-out and control software, significantly cutting software development time. A software layer based on the CORBA standard [5] and common to all platforms – VxWorks [7] (VME/FASTBUS), UNIX, and NT – allows the system to be designed and to operate independent of the underlying hardware/OS structure. In close collaboration with Visigenic Corp. [6] we achieved the first implementation of the CORBA 2.0 standard on a real-time operating system.

The overall design of the DAQ system follows the traditional, hierarchical approach. Using state–of–the–art components gives us the needed performance, but also provides us with the most cost-effective solution. At the "front end" about 35 crates digitize the incoming data in parallel. The (sparsified) data is buffered locally in each data board in at least 7 event buffers. From there a dedicated CPU in each crate moves the data with a readout time under 500 μs into a large capacity memory buffer on the CPU board, performs an additional sparsification step (if needed) and sends the data to a central event builder facility. This event builder has 35 point-to-point links to each crate using the Fast Ethernet standard. Inside it uses the PCI bus to provide the necessary band width for combining all event fragments into one event record and sending it to a software trigger level, executed on a (UNIX) multi-processor workstation. From there the accepted events will be written to a mass storage device.

2 VME Readout with CBLT

Two of the major detector components of CLEO III, the RICH detector and the the Silicon Vertex detector, will use specially designed VME data boards for digitization. The trigger system will also use VME crates to house its electronics and provide a readout connection via the VME bus. Simulations show, that we must use less than 500 μs to transfer the data from the data board memory into the buffer on the VME CPU (called "Datamover"). Therefore we decided to built a *VME interface* For these data boards that uses several extensions (defined by VIPA [2]), following the *VME-P extensions* draft.

Additionally, we implemented on-board data buffer management with similar depth and behavior to the internal buffers used by the FASTBUS modules in other crates. This creates a more uniform readout system, and its design can be *re-used* by all detector components using the VME bus for readout.

We are using the following new features:

- *MCST*: This address mode allows the parallel execution of commands by *multiple slave boards.* We are using this mode to broadcast the command "read next event" to all slaves to prepare the internal buffer management of all boards for data transfer (e.g setting of buffer pointers and prefetching of the first data word) at the same time. Optionally, this command can be used to skip events, if needed. The `IACKIN/IACKOUT` daisy–chain lines are used to forward the "command acknowledge" to the VME master. This chain is managed by tagging each board either as "first", "middle", or "last" board in the chain, so it is properly started, forwarded and ended. The "last" boards ends the cycle by issuing the `DTACK` signal for the CPU.
- *CBLT*: This special block transfer mode allows multiple slave boards to send data in one overall transfer cycle. The advantage of this readout approach is that the CPU does not need to know the specific distribution of the event data in the slave boards. Secondly, this mode is compliant with VME64, so only the slaves have to have special hardware to implement this transfer. Again the `IACKIN/IACKOUT` lines are used to pass the "readout token" between neighboring slave boards. The CPU does not notice those token transfers, but only subsequent `DTACK` signals. The transfer is completed with a `BERR` signal by the last module.

The setup for these two address modes is done by configuring a "CR register" in each module, so it knows whether and how to participate in such operations.

The general access to the boards uses the VME64 access mode A32/D64, but our VME interface is also A32/D32 compliant, if this should be necessary. For the D64 access mode we expect a data transfer rate of about 20 MB/s This means at an average event size of 2.5 KB the readout time would be at about 125 μs.

3 The FRITZ Interface

Instead of having a completely different crate controller in the FASTBUS crates, we have built a FASTBUS-VME interface to use the same CPU board at this stage of the readout chain. FRITZ – *Fastbus Readout Interface with data Translation and Zero suppression* – will make the FASTBUS look like VME.

FRITZ consists of a standard size FASTBUS PC card. It has not only the the necessary FASTBUS interface logic and ECL drivers, but also a complete VME interface including a 2–slot VME backplane. The advantages of this approach are:

- *Easy upgrades.* As soon as higher performance VME CPU cards are available, they can be used for FASTBUS as well.
- *One type of controller.* Since all crate controller have same type, maintaining of hardware is much simpler.
- *One OS and development setup.* All software is written with the same type and version of the operating system. Also the development environment is the same, allowing bug fixes to be done easily to all controller software.
- *Uniform event builder connection.* The interface to the event builder is exactly the same, so there is no difference in the management of the data flow between crates.
- *Uniform trigger interface.* The device providing trigger information also sees the same type of crates.
- *Code is almost the same.* All software not directly used to manipulate or readout the data boards is exactly the same. This includes data buffer management and slow control monitoring.

Besides the basic FASTBUS functionality, FRITZ will provide manipulation of the data before storing them into the CPU's memory. A list driven sequencer combined with a VME master interface is moving the data along a multi-step process:

- Reading a data word in block transfer mode from FASTBUS data board, optionally removing non-data words;
- Additional threshold comparison and channel tagging using a memory lookup device;
- Storing data as VME master into CPU memory.

4 Slow Control Network with CORBA

The overall system will consists of about 100 processors, distributed over a large sized building and connected by a network. To monitor and control such a system, many processes are connected together and are exchanging messages or requests. The usual design of such a "Slow Control Network" needs a common software layer, on top of which to build hardware independent modules (programs or libraries), that can send requests and data to all participating nodes. One popular solution is the "BSD socket" library, but it lacks a standardized message format and a run-time setup of connections independent of network addresses.

In the last 3 years, the Object Management Group (OMG) has defined an object oriented communication layer using *Object Request Broker* technology. CORBA – *Common Object Request Broker Architecture* – provides the means to send requests and data in a hardware/platform independent way The endpoints of this communication are *objects* implemented in any of the supported OOPL, like *C++, SmallTalk,* or *Java*. The data communication is designed according to a client–server model, and the ORB libraries are transparently sending all necessary information to processes "instantiating" such an "object interface".

Locating an object is done by its interface type and its object name *at run time*. This makes setting up such a system very easy, and debugging much simpler. Since all object requests are transparently generated in the standard (only language dependent) syntax, the compiler can type-check this request and many problems are already found at *compile time*. Additionally, the system can be tested in a small scale setup. New processes can be designed and tested outside of the "big system", and later integrated without any code change or recompilation. The latest CORBA 2.0 version even provides the means for interoperability between different ORB implementations.

The evaluation of available CORBA implementations led us to an agent based ORB library by Visigenic Corp. (former PostModern) – VisiBroker 2.0. This implementation uses the CORBA standard as native protocol for longer than a year, and it proved to have a very effective way to setup fast connections between objects. It was readily available for the UNIX and NT workstations we were planing to use.

In close collaboration with Visigenic, we ported this implementation of CORBA 2.0 to VxWorks 5.3 for the Motorola PowerPC 604 platform. This is the first implementation on a real–time operating system. We now have implemented CORBA servers on the VxWorks system, to configure any node with the correct object modules and to start all needed tasks. Other objects will be "activated" during this process, so that servers (e.g. to control the data boards or the readout tasks) become available wherever needed.

This method of communication will also be used to work with central services, like:

– The alarm system,
– a run control "object"
– a global data base and logging service,
– the shifter's console, a GUI.

Using CORBA implemented in the *Java* language, we are also building a generic data/histogram presenter, running under any Java–enabled web browser, thus using WWW technologies to disseminate information throughout the CLEO collaboration.

References

[1] CLEO III Data Acquisition System. Technical Design Report; *CLEO CBX-95-104*.

[2] VME for Physics "Implementation Recommendations & Guidelines, Draft Document"; (*http://www-vipa.fnal.gov/vipa/doc2.html*)

[3] VME64 Standard; *ANSI/VITA 1-1994*.

[4] VME64 Extension Draft; (*http://www.forcecomputers.com/technology/papers/v64ext.html*)

[5] Object Management Group, CORBA 2.0 specifications 96-03-04; (*http://www.omg.org/library/schedule/Technology_Adoption.htm*)

[6] VisiBorker 2.0; Visigenic Corp; (*http://www.visigenic.com*)

[7] VxWorks 5.3; Tornado 1.0; Wind River Systems; (*http://www.wrs.com*)