



Front-end Software for the DØ Silicon Tracker

Q. Jia, D. Buchholz, S.Y. Jun, Y. Li, R. Snihur, T.L.T. Thomas and R. Tilden

*Northwestern University
Evanston, Illinois 60208*

J.A. Wightman

*Iowa State University
Ames, Iowa 50011*

J.F. Bartlett, H. Lan and L. Paterno

*Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510*

January 1997

Presented at the *IEEE Nuclear Science Symposium and Medical Imaging Conference*,
Anaheim, California, November 2-8, 1996

Disclaimer

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Distribution

Approved for public release; further dissemination unlimited.

Front-end Software for the DØ Silicon Tracker¹

Q. Jia, D. Buchholz, S.Y. Jun, Y. Li², R. Snihur, T.L.T. Thomas, and R. Tilden
Northwestern University, Evanston, IL 60208

J.A. Wightman

Iowa State University, Ames, Iowa 50011

J.F. Bartlett, H. Lan³, and L. Paterno

Fermilab, Batavia, IL 60510

Abstract

Fermilab's DØ experiment is constructing a new silicon microstrip detector as part of its upgrade detector. This will have nearly 800,000 instrumented channels and combined with the rest of the tracker accounts for nearly one million channels. Being able to monitor, calibrate, and diagnose problems with this many channels is a daunting challenge. We propose to use distributed processors to "spy" on the data as it is collected. These processors will be resident in the VME data acquisition crates and will be able to access the data over either VME or a secondary bus which is independent of the main data acquisition path. The processing of the monitor data will take place in these local processors. Communication with the online cluster will be over ethernet and will employ a graphical interface for user control. The design uses a client/server architecture in this network of processors. We describe the software and hardware which has been tested as part of the verification of this design.

I. INTRODUCTION

The design of the front-end software for the DØ tracker includes support for diagnostics, calibration, and monitoring of the approximately one million channels of readout electronics. All these processes are capable of being run while portions of the detector are being tested as well as after it is installed and taking data. A design criterion for the diagnostics is that they be capable of being run in situ without removing any electronics. The calibration must also be done under the same conditions. The monitoring must be done while the normal data acquisition is underway, but it must operate in such a way as not to impact the data acquisition adversely by causing any deadtime. Although the upgraded DØ detector will not be operational until 1999, the construction of the readout electronics and the silicon microstrip detectors has begun.

This paper discusses not only the proposed solution for the final detector but the work in progress which must support the production testing of all assembled silicon microstrip detectors with their bonded readout chips and the test beam efforts for measuring the performance of these detectors.

II. MOTIVATION

A major component of the DØ upgrade involves replacing the present tracking system with a combination of a silicon vertex detector [1] and a scintillating fiber tracker [2]. The new tracking system replaces the ~ 7000 instrumented channels of the existing tracker with $\sim 1,000,000$ channels. With this many channels to monitor, we can no longer continue as in the past where we read events in the host computer from the Global Shared Common, the data pool available to all the monitor programs, and analyzed them to monitor the detector performance with the results displayed in a series of histograms for the detector shifter to examine. Trying to maintain one million histograms in a single location is a very daunting task.

To illustrate the scope of the problem we consider the following simple example where we round off some of the numbers for simplicity. We assume we have 1M channels that we wish to analyze by histogramming the ADC spectrum for each channel. The SVXII chip [3] uses an 8-bit ADC, so we have 256 possible values. We assume that we only need to allocate 2 bytes per histogram channel, and we assume that HBOOK [4] is used to create and maintain the histograms. With HBOOK we need 40-50 words (again assume single precision) for header information, like the identification, the title, number of channels, and lower and upper bins. Therefore, we need ~ 300 words (600 Bytes) per readout channel for the histogram or ~ 600 MBytes for all of the histograms. Trying to handle this in a central location would mean a fairly high page fault rate in the processor, at the very least. It can be argued that this number is an upper limit on the memory requirements since not every possible ADC value needs to be histogrammed; also, we might be able to reduce the number of channels monitored but it would be short-sighted to preclude the possibility of monitoring all channels at this point in the design.

¹This work was supported in part by the U. S. Department of Energy.

²Now at Hughes Information System, Landover, MD 20785

³Visitor from IHEP, Beijing, P.R. China

In addition, with the Silicon tracker expected to have an occupancy of $\sim 3 - 5\%$, using the events from the data pool does not guarantee high sampling statistics. Finally, it is not entirely clear whether using fully triggered events (Level 1 – Level 2 – Level 3), like those in the data pool, might not bias the results.

III. PROPOSED SOLUTION

Our proposed solution encompasses both hardware and software and is discussed in the following sub-sections. First we address the question of how the hardware can handle this proposed solution.

A. Hardware

As discussed above, monitoring all of the channels of the Silicon tracker requires a very large number of histograms. The simplest way to handle a large number of histograms is to do so in a distributed environment; that is, we can have many processors handle the task rather than demanding a single, central processor do the entire job. The detector readout system provides a very natural distributed environment with the readout spanning some number of VME crates. In particular, the present layout of the Silicon tracker readout uses 10 VME crates. To do the processing we propose to use an embedded processor in each of the front-end crates which solves the problem of shipping large amounts of data around the DAQ system. This is technically feasible as processor boards based on the 680x0 family of processors which include VME interfaces are readily available in today's market with next generation processors, like the PowerPC, just now becoming available. Therefore, we should be able to obtain the necessary local processing power to do the job.

We can make a very conservative estimate of the sampling statistics assuming old technology, a 68040 processor. With this configuration we can expect to make 10K histogram entries per second so that at the end of a 4 hour run we should have over 1400 entries per histogram, assuming 100K channels to be monitored in the crate. This corresponds to 100 entries in a little over 15 minutes, which should be sufficient statistics to determine if the channel is out of tolerance. This estimate only gets better with time as we use the PowerPC or its successor.

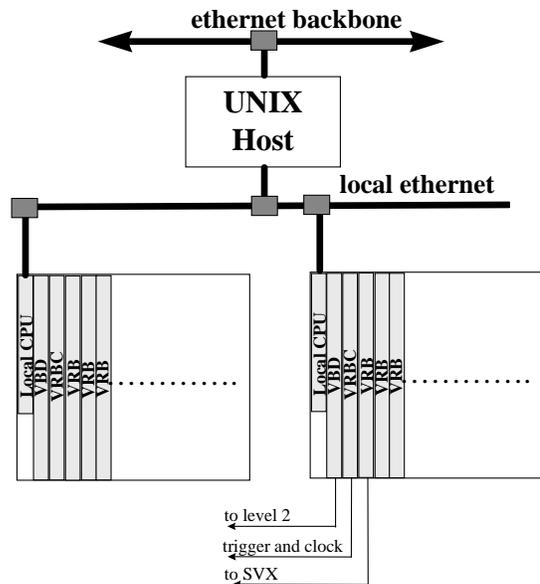
One potential concern with this solution is the incorporation of a processor in the front-end digitization crates due to the presence of high-speed clocks. High-speed clocks can potentially produce high-frequency noise which can be picked up by an ADC. This is not a problem for the silicon tracker since digitization is performed in the SVXII chip and not on the readout board. The standard network interface is to Ethernet so that all communication with these crates will be over Ethernet.

There remains the challenge of how to get the data into the VME-resident processor without impacting the DAQ system. One design criterion we set for ourselves is to find a solution that does not add any deadtime to the experiment. Using some representative design parameters from the silicon detector, like a maximum of 128K channels per crate and 10KHz Level 1 accept rate, there are not sufficient unused VME cycles to meet

our sampling requirements. Therefore, we are forced to look to a private databus. This requirement for a monitor interface has become a part of the design specification for the readout board, the VME Readout Buffer (VRB) [5]. At present the design uses IEEE P-1394 [6] and data FIFO separate from the VME data FIFO's.

The IEEE P-1394 is a high-speed serial bus that is gaining attention from the electronics industry. Some features of this standard are live insertion of the modules, simultaneous asynchronous and isochronous operations, and compatibility of different bandwidth devices on the same bus. An interface chip set is presently available from Texas Instruments [7] that supports 100Mbs (Mega-bit/sec) operation with 200 Mbs due for release shortly. Plans are to have a 400 Mbs interface available within a year. With several of the industry giants looking favorably on P-1394 and proposals to increase the bandwidth to as much as 1.6 Gbs, it appears that we will be adopting an interface with a relatively long lifetime.

There is an additional advantage to be realized with using an embedded processor in each VME crate. We could allow it to control downloading the SVXII chips via the Port Card if we also incorporate a 1553 interface in our crates, see [1] for details. Since there will be a very standard download for each chip for global data runs, this will simplify the download procedure as only a simple command to download is necessary. The pattern could be resident in a local database in the front-end thus obviating the need for sending it with the download command. Figure 1 shows a schematic of the layout of the host connections to the front-end processors and their connections to the silicon readout.



VME crates with VxWorks +EPICS+DART

Fig. 1 Schematic of host connection to the front-end processors in VME crates along with the connections that continue to the SVX readout chips.

B. Software

Now that we have specified a workable hardware solution we address the question of the software environment. We have chosen to use VxWorks[8] for our real-time kernel since the Fermilab Online Support Department (OLS) fully supports it for both the Sloan Digital Sky Survey as well as for the DART project, the DAQ system being used in most of the fixed-target experiments in the 1996–1997 run.

However, the real-time system must be able to communicate via the current controls system protocol. At present we use the ACNET protocol developed by the Fermilab Accelerator Controls group, but are evaluating EPICS [9], the Experimental Physics and Industrial Control System, as a replacement. One feature of EPICS is that it uses VME-based processors running VxWorks with TCP/IP communication over Ethernet. Thus, no matter which solution for the controls system is adopted, either ACNET or EPICS, we will use VxWorks.

We are using the VxWorks cross development system which includes a C compiler for the 68040 processor on the Motorola processor boards, MVME162, 166, and 167. Wind River Systems[8], the developers of VxWorks, also supports C++, but we have opted to use C for now. However, we have not precluded upgrading to C++ in the future as we are undertaking an object-oriented design of the front-end software.

IV. SOFTWARE DESIGN

We have divided the software into two basic categories, “system” processes and “user” processes. The “system” processes provide the software framework since they are independent of the specific sub-detector. The “system” processes will be restarted whenever the node is restarted. The “user” processes are the ones that are sub-detector-specific and will only be started when requested. Both are explained in more detail in the following sub-sections.

A. “System” Processes

The first of these is the DAQ Front-end Process which makes the node have the look and feel of one of the control system nodes. It will speak the control system protocol, whether that will be ACNET or EPICS. This means that it will also handle all of the network communication which will be over Ethernet in the control system protocol. It will also handle all of the alarm system functions, like the alarm scanner, the local database of nominal values and tolerances, and connection to the DØ alarm system. If EPICS is adopted, then it will fill the role of the DAQ Front-end Process. In connection with this we have requested that COOR (the main DØ runtime control program) broadcast all Begin/End Store and Begin/End Run messages so that we can allow the front-end software to respond to these events if necessary.

The second of these processes is the Event Manager. This process will be responsible for obtaining the data from the readout board and distributing it to the “user” processes that request it. The Event Manager software will provide a standard Application Programmer’s Interface (API) for the “user” processes. We expect this data collection to be interrupt driven.

To make this process more efficient, the Event Manager will use an interrupt service routine to signal when data is available. The Event Manager will maintain a circular buffer of pointers to individual data buffers, so that when a “user” process requests a buffer of data, it will simply be given a pointer to the buffer. In this way we avoid having to move the data any more than is necessary. It will be the responsibility of the “user” process that requested the buffer to return the pointer when it is finished processing the data. This implementation of the Event Manager allows acquisition of the events to be asynchronous with the processing of the events.

In addition to these processes which will be running whenever the node is running, we will also have supporting software systems available to the “user” processes. One of these is the histogramming system. At present we support 1- and 2-dimensional histograms with user-specified lower and upper limits and equal bin widths. Complementing the histograms are distributions which are running sums for calculating the statistical moments of the distributions being histogrammed. This precludes having to loop through the histograms every time one of these quantities is requested. Currently, we calculate the mean and variance for the distributions.

B. “User” Processes

We currently envision three distinct “user” processes, but the system we are designing is flexible enough to allow this list to expand if needed. These are the Online Data Monitoring, Calibration, and Diagnostics.

1) Monitoring

The first of these processes is the Online Data Monitor. This process will operate during a run and will “spy” on the data as it is being collected. It will create histograms and distributions and fill them accordingly. The exact histograms have not been specified, but we anticipate that these will include spectra to study pulse heights and bit frequency, both of the Gray code and the binary ADC output. The alarm scanner will be triggered periodically to check the mean and variance of these distributions against the nominal values and tolerances maintained in the local database. An informational alarm will be generated and sent to the Host whenever the mean is found to be out of tolerance.

For data monitoring we expect that the IEEE P1394 will provide the secondary bus so that the data being “spied” on does not interfere with normal data transfers over the VME backplane. For the testing we have been doing up-to-now we rely on using the regular VME backplane for both data collection and data monitoring. During normal data collection a maximum of one-eighth of any event will be available in the “spy” buffer that is accessible over the secondary bus. The local VME processor will acquire its monitor data from this memory and analyze the event fragment. We are designing the online data monitor to be sensitive to any changes in the operation of the detector and/or readout electronics in real-time. We expect to monitor such distributions as the pulse height spectrum of each channel and will periodically compare

various statistical moments of these distributions to locally stored reference values to signal any changes in the operation of the detector and/or readout electronics in real-time. Any out-of-tolerance conditions will be communicated to the online system through the DØ alarm system. These can be entered into a detector-specific database for incorporation in the off-line reconstruction of the physics data. The ability to monitor the performance in real-time is of paramount importance to the successful operation of the DØ upgrade detector.

2) Calibration

The calibration of the silicon microstrip detectors is accomplished by injecting a known charge into the digitizer readout chip and comparing to the output of the chip. For the silicon tracker using the SVXII chip this is most easily and efficiently accomplished in the readout crate since this chip generates its own calibration pulse. Readout of this data by the processor will be over the VME backplane to take advantage of the increased bandwidth. The processor has full control over the procedure as it selects the amount of charge injected and compares with the digitized readout. Before the calibration sequence can begin, the SVXII chip has to be downloaded with a proper set of values. This is best accomplished by having the local processor having the download control after having been given permission by COOR.

Under normal operating conditions we envision the various bits of information that must be entered into the calibration database, like the various statistical moments, will be sent up to the Host. In addition, there will be an expert mode in which the full event will be sent to the Host for study and debugging.

3) Diagnostics

The third “user” process we envision will be the Diagnostic process. We have developed a unique programming language for writing programs that can access VME-based hardware. This language, which we call TE, resembles certain aspects of Basic, Pascal, and C but is simpler to use. The code can be run interactively through the interpretive mode, or it can be converted to C code and compiled. The current version runs both in our Motorola 68K processors using the VxWorks operating system and in a PC with a Bit3 VME interface. The interactive code can be quickly modified for testing hardware, while the compiled code is run when the highest speeds are required. The design of this software allows us to run diagnostics under the same conditions, including speed of execution, as will be used in the final installation. It also supports diagnostic testing at the test bench facilities. This has been used to debug and test some of the prototype electronics.

While TE has proved to be very convenient for ease of debugging hardware, the final diagnostic programs may very well be code written in C or C++ to take advantage of speed of execution that can be realized when code is produced specifically for a certain processor.

V. USER INTERFACE

Since VxWorks is a single user system, it is not desirable to have the user, be it the detector shifter or the electronics expert, logging into the front-end node to use it. Instead, we will develop and maintain a GUI for the user on the Host which will obviate the need for remembering arcane commands however clear they may be to the original author. We will assure that this software adheres to the DØ online standards, that is, we will communicate with the control program to obtain ownership of resources whenever we want to start the Calibration or the Diagnostic process. This communication will be through the user interface and not with each front-end crate individually. In addition, the user interface will include the ability to display the information returned from the front-end nodes, like graphically displaying histograms, for example.

VI. PRODUCTION TESTING OF SILICON

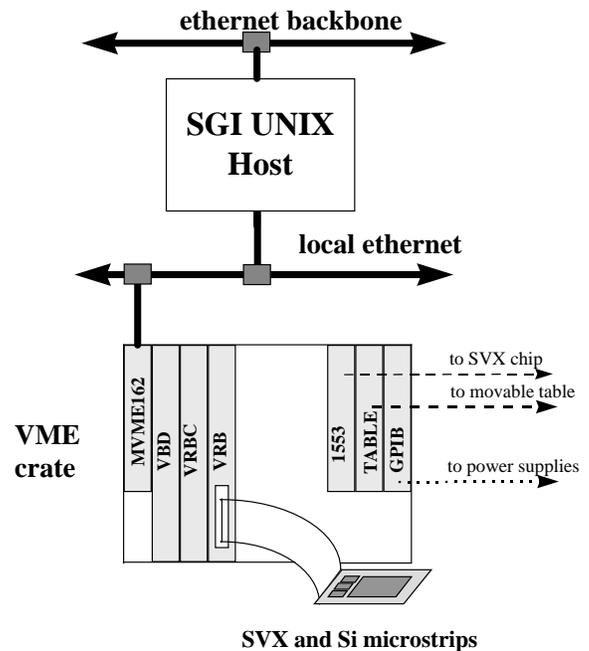


Fig. 2 Schematic layout of the host for production tests of silicon detectors and the VME crate with its associated readout electronics and connections to other standard buses.

After each silicon microstrip detector has been bonded to its substrate and SVXII readout chip, it must be tested for errors and to measure its response. The detectors will be placed on a computer controlled movable table which has a laser suspended above it. The laser can produce a focused spot of light smaller than the strip width of the microstrip. Many of the programs described above as part of the software design for the DØ upgrade have been implemented as part of the production testing of the individual silicon detectors after they are fabricated. A TCL/TK GUI interface for a Silicon Graphics workstation has been tested which allows the user to initiate standard tests and also complete expert level testing. The code for calibration, monitoring, and diagnostics is run as local

code on the VME based 68K processor in a server mode. The workstation is a client to the 68K processor and communicates via an ethernet link. The user on the workstation is able to click buttons to initiate a specific task. The 68K can access GPIB, CAMAC, and 1553 buses via VME interfaces. It can control the movable table through a VME interface card. This is shown in a schematic view in figure 2.

VII. CONCLUSION

Some of the software described above is still in the design stage but substantial portions have been implemented. Those parts necessary to control and monitor the production testing of the silicon microstrips have been written and tested with prototype detectors and readout electronics. The software will continue to evolve as we get more experience with the production testing of the microstrip detectors and in a test beam early next year. The final product will be used for the next data taking stage of the DØ detector which is expected in 1999.

VIII. ACKNOWLEDGMENTS

We gratefully acknowledge the support staffs at each of the participating institutions. Financial support has been provided by the U. S. Department of Energy.

IX. REFERENCES

- [1] "DØ Silicon Tracker Technical Design Report," DØ Upgrade Collaboration, DØ Note 2169 (unpublished).
- [2] "DØ Upgrade Technical Design Report."
- [3] "A Beginners Guide to the SVXII," R. Yarema, DØ Engineering Note 3823.112-EN-399 (unpublished).
- [4] "HBOOK, Version 4.20," CERN Reference Manual Y250 (unpublished).
- [5] "VME Readout Buffer," H. Gonzalez, D. Mendoza, M. Bowden, T. Zmuda, M. Johnson, and E. Barsotti, Document # ESE-SVX-950719 (October 25, 1995).
- [6] On WWW, <http://www.firewire.org/> gives the 1394 Trade Association home page.
- [7] On WWW the Texas Instruments Product Information and Document Search Page (<http://www-s.ti.com/sc/docs/psheets/pids2.htm>) allows you to search for current 1394 support.
- [8] VxWorks is produced by Wind River Systems, Alameda, CA.
- [9] On WWW, http://www.ceba.gov/accel/documents/epics_doc.html.