DEVELOPMENT STATUS AND DEPLOYMENT OF THE NEXT GENERATION NSCL DATA ACQUISITION SYSTEM

R. Fox, E. Kasten

1. Introduction

As the NSCL began the coupled cyclotron construction project, it was clear that the existing data acquisition system would be inadequate. The existing data acquisition system was based on an embedded kernel running in antiquated front end processors with a mix of C and assembly language code[1]. The online analysis and recording component was based on largely non-portable FORTRAN and C software written for the VMS operating system. Nuclear electronics had advanced past the assumptions on which the front end system was built. We also anticipated the need for a more flexible analysis package than the ones used over the lifetime of the K1200.

This article will describe the status of the new NSCL data acquisition system. The configuration of development and deployment systems will be described. The map of this article is as follows:

- The section *Status and Configuration* will describe the hardware and software configuration of the system as it now stands.
- The section *Deployment Environments* will describe significant instances of the system at its current development level within the NSCL.
- The section *What needs to be done* will describe work in progress to meet the next set of anticipated needs.
- The *Conclusions* attempt to distill our experiences with the development and use of this system in hopes that they may guide others considering such an undertaking.

2. Status and Configuration

This section describes the configuration of the system. Subsections describe:

- The hardware configuration of the system.
- The software components of the system.

2.1. Hardware Configuration

Our goals in choosing hardware for this system were:

- Choose only commercial components
- Build the system with cost effective speed upgrades in mind
- Avoid expensive software components
- Allow existing hardware to be re-used where possible

From the start, it was clear that the Intel/PC platform would be an ideal implementation platform. Components and software are cheaply available for this hardware. This hardware choice forced the operating system choice into one of the following:

- Windows based systems like Windows Nt or Windows 98
- Real-time Kernels
- Linux

Performance measurements for Windows based systems quickly convinced us that they were not suitable. We have had experience developing code and forcing users to develop code within Real-time kernels and were reluctant to inflict this on future users. Measurements of Linux performance initially indicated no insurmountable performance problems.

The lure of Linux is to offer a fully functional operating system at all levels of the acquisition system. Using UNIX at all levels of the system allows code to be easily moved from “embedded” to “normal” parts of the system. The freeware nature of Linux and its large development community lets us leverage both our development and the user’s development efforts on top of a rich, powerful, yet inexpensive base.

Existing NSCL data acquisition systems used VME-based controllers to access CAMAC. In addition, VME components were used to readout LeCroy Ecl-Line compatible electronics. VME based ADCs, now available from e.g. CAEN, are also an attractive way to package the digitization electronics. These factors led us to specify that the interfaces to the experiment would be through VME-bus modules.

The PC platform could then either be an embedded VME-bus PC-compatible module, or it could be a “normal” PC coupled to a VME bus. A survey of available VME-bus PC-compatible modules indicated that:

- These modules tend to lag behind the performance point of “normal” PCs.
- These modules tend to be more expensive than a “normal” PCs.
- These modules tend to be built around an internal PCI bus architecture with PCI to VME bus bridges connecting them to the VME bus.

On the basis of these factors, we decided to use ordinary PC’s connected to the VME bus via a SBS/Bit3 Model 618 PCI to VME bus bridge. The adapter connects the VME and PC together via a fiber optic cable which has the added benefit of decoupling the noisy digital electronics of the PC from the desired quiet of the VME bus. Linux driver software for this device was available from NIKHEF[2], and was modified by Eric Kasten to reflect our needs. The device and its associated driver support user level programmed control access to the VME as well as DMA transfers to and from VME ”memory-like” devices.

For applications which require CAMAC access, a CES model CBD8210 CAMAC branch highway driver can be plugged in to the VME crate. This system is shown schematically in Figure 1.

2.2. Software components

One of the more successful features of prior NSCL data acquisition systems has been the flexibility with which sources and sinks of event data can be added to the system. Our software component goals are to:

- Extend the dataflow architecture to support distributed routing
- Extend the dataflow architecture to support the routing of non event data.
- Provide a class library and programming framework to support

Last year we described the generic dataflow architecture[3]. Another article in this annual report[4] describes the online/offline data analysis software we have produced. Significant advances described in this report include:

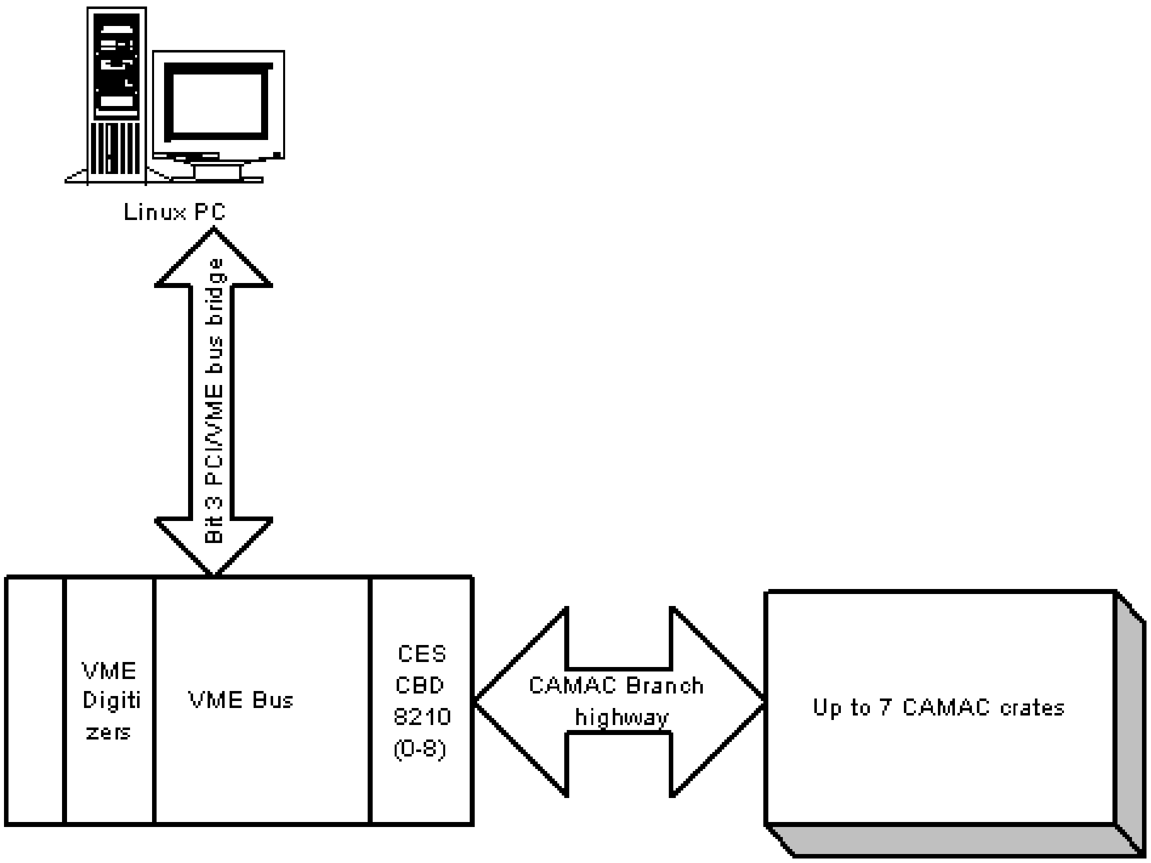

Figure 1: Hardware configuration of the system

- A Readout Framework.
- Data logging software.
- New features being incorporated in the data routing software

2.2.1 Readout Framework

Our initial goals for this development project were to provide a readout framework which would support easy migration of existing "MASH" code to the new system. This turned out to be relatively easy. An initial version of this readout skeleton is now available and in use in test environments.

MASH software accessed the CAMAC by directly addressing the CES branch driver relative to a base address. The NIKHEF SBS/Bit3 adapter supports mapping arbitrary blocks of VME address space to process space via the mmap(2) Linux system service. Performing this mapping in the initialization segment of the readout skeleton allows users to transport their CAMAC readout code unchanged to the Linux system.

A subset of the MASH command interpreter was embedded into the readout skeleton to support run control. The entire application code is a subclass of the DAQROCNode class, and thus is able to make use of the complete set of buffer management services available to producers of data.

2.2.2 Data logging software

Our concepts of data logging for the new system are heavily influenced by the fact that coupled cyclotron experiments will primarily be done with radioactive beams. The resulting low event rates require us to record data to disk rather than tape. Disk based event files allow the user to make changes to the analysis setup and immediately replay old data. In tape based analysis, an iteration would require either dismounting and analyzing the current tape, or a time consuing accumulation of new data.

It will be possible for data taken to be periodically staged to tape either manually or automatically. We will allow staged event files to be either purged or not at the user's discretion.

At present the recording portion of this system has been completed. Individual runs can be recorded to a directory in file names which clearly identify both the run number and the blocking factor. SpecTcl, can easily switch between online and offline modes, allowing users to change analysis conditions, play back the current run to end of file and then continue analyzing raw data.

2.2.3 New Data Routing Features

Initial versions of the data routing server supported only a 'reliable' delivery system. Current versions of the system support 'unreliable' delivery, which is suitable for monitoring software which does not need to be on the critical performance path of the data flow. In our past systems, this delivery method was termed "rate based sampling."[5]

It is our intention to use common data routing and distribution mechanisms to manage all of the data flows of a data acquisition system. In[6], Nomachi et al. characterized the data flows of a data acquisition system as follows:

Event Primary high performance data flow for event data.

Command Data flow for control messages between either human or programmed controllers and components of the data acquisition system.

Status Data flow which describes time varying state information.

Nomachi goes on to describe performance requirements of status flows as dependent on two axes: Monitoring rate and update rate. We feel that in fact, only the monitoring rate is important, since it doesn't matter how quickly a quantity is varying if, in fact, nobody cares about it. Furthermore, one must be assured that high rate monitoring always is getting up-to-date values.

To support both command and status data flows, the server is being augmented with support for small out of band messages. These messsages bypass the queueing associated with event data and are delivered immediately to software attached to the dataflow. Commands can then be sent to software in the system by dropping a "message in a bottle" and sending it out of band along a port of the data distribution system. Status flow updates, and requests for updates can be accomodated in the same manner. In both cases a clearing house is needed to dynamically keep track of the composition of the system so that it is known along which paths to send these messages and from where to expect responses.

3. Deployment Environments

SpectroDaq is being developed in an incremental fashion. As each major feature set is completed, a new version is released. Currently, several groups are using SpectoDaq in the NSCL, including the Gamma-ray spectrometer development group, the coupling line diagnostics group, L. Sobotka et al. in Washington University (older releases). Detector testing at the NSCL is largely being done with current versions of the system.

User experience with the system has been largely positive. People have been able to transfer older readout software to the new system with minimal pain. New software is easily written and integrated. Experimentalists enjoy being able to symbolically debug their software at source level. The rapid rapid turnaround of a host development environment, which removes the extra program load step of an embedded cross development environment greatly enhances productivity.

User feedback has been very helpful. Users have found errors which have slipped through our testing procedures. They have also contributed useful comments which have been incorporated in subsequent releases of the system. The iterative design and release methodology has also ensured that users have had a usable system at an earlier stage than other methodologies (e.g. waterfall) would have provided.

4. What needs to be done

This section describes:

- Specific work to be done.
- Our strategic directions for system configuration
- Our strategic directions for building user extensible software

At present a great deal of specific work must be done to make this system a complete data acquisition system. While the basic data distribution system provides sufficient power to serve as a control and status flow subsystem, we must provide specific support in the application framework for these flows.

The application framework will also be extended to support specific virtual member functions for the receipt of data along each of these paths. In this manner, particular software can be built by simply overriding default no-op methods. This is in keeping with modern object oriented software development strategies.

Finally we must resolve the following software installation and configuration issues:

- The software is somewhat sensitive to the versions of library software installed on the host linux box. Many Linux distributions appear to be rather chaotic regarding the library versions they installl.
- We now have about 5 installations within the NSCL of this system. Keeping them all up to date as new versions of the software are released is a difficult problem. rdist helps to get the program files to the appropriate systems, however it does not ensure that servers get restarted at times which are appropriate to the scheduled use of each individual computer.

4.1. Strategic Directions

As we have built prototypes of major components of this system, we have been struck by two recurring needs:

- The need to build components which are easily extended both at compile time and at run time.
- The need for meta-descriptions of the system configuration.

4.1.4 Extensible Components

Components we provide are often used in ways we did not anticipate. This is a good thing. We intend to use the Tcl/Tk scripting language as a base command language for all components of the system. This allows us to support run-time extensions of the functionality of the software and its user interface via Tcl/Tk scripting. It also allows support for compile time extensions of the command set via C++ wrapper classes around the Tcl command registration procedures.

Tcl/Tk scripting provides a common basis for automating tasks within the data acquisition system. The Tk component provides powerful GUI creation and modification tools available to all interactive components.

Providing Tcl/Tk extension command packages which support introspection allow the user to extend, at run time, the functionality of the software via Tcl procedures. A Tcl management component has been added to the data routing server. In another report article we describe the Tcl/Tk scripted data analysis software we are developing.

The application framework will be extended in the future to support embedded Tcl/Tk interpreters. This will allow users to easily write their own Tcl/Tk scriptable components.

4.1.5 Meta-Descriptions

All aspects of a data acquisition system can benefit from a meta-language description of the system. In past systems, databases have been developed as well as specialized definition languages to either populate or replace these databases. In keeping with modern information interchange trends, we intend to use XML (eXtensible Markup Language) to describe our system and experiment configurations.

The parallels between Tcl/Tk as a command language and XML as a configuration language are clear. Both languages provide an extensible framework. Tcl/TK provides a base language and syntax for scripting. XML provides a generic syntax for self describing documents. Document Type Definitions (DTDs) further define the actual set of tags accepted by XML. XML's Style Language (XSL) allows XML supporting web browsers to display XML information in a standard way.

Figure 2 shows an example of how part of an experiment might be configured using XML. In the next year we will be working to define DTD's for sections of experiment and system configuration. Our goal is to be able to define system configuration and experiment in a single XML document.

```
...
<EXPERIMENT NAME=2000-107>
...
<MODULE TYPE=''CAEN7XXX'' NAME=''ADCS''>
   <BUS>VME</BUS>
   <SLOT>2</SLOT>
   <CHANNELS>
      ADC1
      ADC2
      ADC3
   </CHANNELS>
</MODULE>
...
<READOUT>
<TRIGGER SOURCE=''1''>
   <READ>
      <MODULE NAME=''ADCS''>
       <CHANNELS>
           ADC1
           ADC3
        </CHANNELS>
      </MODULE>
   </READ>
</TRIGGER>
...
</READOUT>
..
</EXPERIMENT>
...
```

Figure 2: Sample section of XML experiment definition

5. Conclusions

While the next generation NSCL data acquisition system is usable in its present pre-release form. There is a great deal we want to do to make the system more usable during the commissioning phases of the coupled cyclotron. User experiences with the system have been largely positive.

The system supports data flow involving an arbitrary set of data sources and sinks. We also have a readout skeleton which can take data and submit it to the data routing system. A data analysis package usable in both the online and offline environment leverages on the Xamine display server and Tcl/Tk to provide a powerful scripted data analysis environment.

We will be turning our attention to the control and status flows next. The data flow system built by E. Kasten has enough flexibility to handle these data flows without much modification. Our work is therefore deciding how to use this system and how to encapsulate this usage for the experimenter.

Small scale experiments and detector development projects have been using our system already. This use has provided valuable feedback to us in the form of bug reports and usability comments.

References

1. *A multitasking multisinked, multiprocessor data acquisition front end*
R. Fox et al. IEEE TNS-36 no. 5 (Oct. 1989) pp1562-1567
2. See: http://www.nikhefk.nikhef.nl/~natalia/projects/vme.html
3. See: http://www.nscl.msu.edu/research/1998_Annual_Report/Kasten.pdf
4. *Development and Status of SpecTcl* R. Fox et al. 5. *Progress on the Data Acquisition System at NSCL*
R. Fox et al. IEEE TNS-32 no. 4 (August 1985) pp 1286-1289)
6. *UNIDAQ* Procedings of CHEP94 Nomachi et al.
online at: http://www-online.kek.jp/ online/Unidaq/chep94