SLAC-PUB-6236
June 1993
(I)

A Small Unix-based Data Acquisition System

D. Engberg and T. Glanzman
Stanford Linear Accelerator Center
Stanford, California USA 94309

Abstract

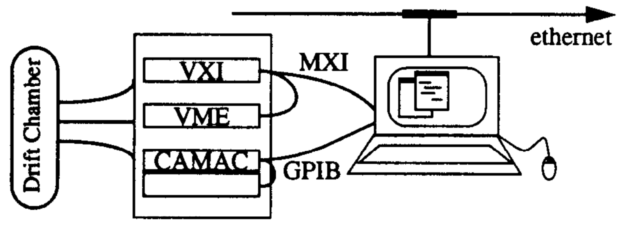
The proposed SLAC B Factory detector plans to use Unix-based machines for all aspects of computing, including real-time data acquisition and experimental control[1][2][3]. An R&D program has been established to investigate the use of Unix in the various aspects of experimental computation. Earlier R&D work[4] investigated the basic real-time aspects of the IBM RS/6000 workstation running AIX. The next step in this R&D is the construction of a prototype data acquisition system which attempts to exercise many of the features needed in the final on-line system in a realistic situation. For this project, we have combined efforts with a team studying the use of novel cell designs and gas mixtures in a new prototype drift chamber[5].

I. FUNCTIONAL REQUIREMENTS

The prototype drift chamber is a 90 sense wire device requiring both time and amplitude measurements. FADC information on a limited number of channels is also required to study pulse shapes. Triggers are generated by the coincidence of a cosmic ray through a pair of scintillators placed on opposite sides of the chamber. Enabling and resetting of the trigger must be accomplished under computer control. The data acquisition system must be capable of handling an event rate of ~1 Hz; be able to sense and report errors; format the data into usable structures; log data to tape/disk; provide real-time data access to an analysis process; and, provide an operator run control interface. This system must be operational by late June 1993. In addition to the requirements imposed by the drift chamber itself, we have imposed a number of requirements specifically to test features of the Unix and data acquisition environments which will be needed for a future on-line system. These extra requirements include: functional separation of processes; distributing processes across machine boundaries; making use of standard network protocols for data transfer; use of interprocess communication mechanisms both for data and control; ability to play back logged data through the system; use of new external I/O buses; creation of a simple GUI for the control program; and, use of an object-oriented programming language.

II. HARDWARE ARCHITECTURE

Hardware components for this system begin with the digitization of analog signals and are shown schematically in Fig. 1. All 90 sense wires are instrumented with LeCroy 2249 gated ADCs and LeCroy 2228 TDCs resident in two CAMAC crates. This choice was largely governed by economic considerations and the ready availability of this equipment at SLAC. CAMAC is connected via a LeCroy 8901 crate controller to a GPIB bus and finally into an IBM RS/6000-520 workstation via a National Instruments MC-GPIB interface. The CAMAC crates also contain crate verifier and dataway display modules for diagnostics. FADC measurements are made by two STRUCK DL515 VME modules. Each module contains four channels operating at up to 250 MHz with 8-bit resolution. The VME crate is connected to the workstation via a National Instruments VME controller, MXI cable and MC-MXI interface. All of the National Instruments equipment (including the GPIB interface) comes with driver software and configuration utilities. Finally, triggering and trigger control will be done using a custom design VXI module based upon a Interface Technology development board with a register-based daughter board. The VXI crate is daisy-chained to the VME crate via the MXI cable and a National Instruments slot 0 controller module.



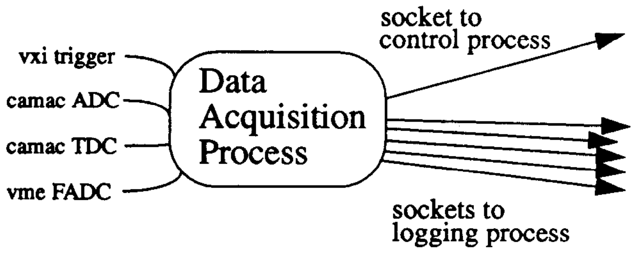


Fig. 1. Data Acquisition Architecture

The computational heart of this system is an IBM RS/6000 model 520 workstation equipped with 64 Mbytes of RAM, 670 Mbytes of local disk space, an internal 40 Mbyte/s Microchannel I/O bus, and is running the AIX (version 3.2) operating system. This machine is connected via ethernet to SLAC's public ethernet circuit and operates under both NFS and NIS for access to remote disks and centralized user accounts respec-

*Work supported by Department of Energy, contract DE-AC03-76SF00515

tively. One other machine, an IBM RS/6000 model 320H, which is also connected to the public ethernet is being used to support the multiprocessor aspects of the system.

III. SOFTWARE ARCHITECTURE

The software is written in the C programming language and consists of five processes: data acquisition; control; logging; analysis; and, playback. This scheme appears in Figs. 1 and 2. The data acquisition process is responsible for initializ-

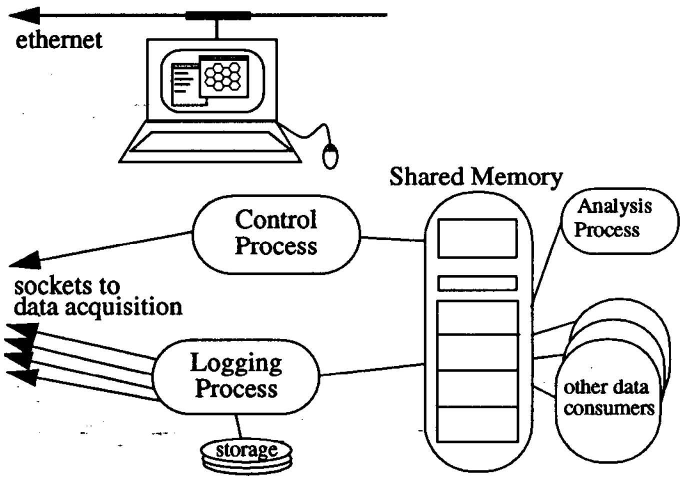


Fig. 2. Control and Analysis Architecture

ing and testing the external I/O buses and modules; controlling the trigger; reading out event data; and, sending event data to the logging process via an array of standard TCP/IP sockets. Electronics configuration (*e.g.* CAMAC crate population) and physical to logical channel mapping are done with two short plain text files. The logging process receives event data into a standard Unix shared memory segment which is large enough to hold several events, then logs the data to disk or tape. The analysis process samples events from the shared memory segment, performs reconstruction and creates a one event display (using the public-domain xgks graphics library[7]). Synchronization of reading and writing to the shared memory segment is done through the use of standard semaphores. The playback process may be used to read previously logged data and place it into the shared memory segment, thus making it available to the analysis program. The control process is used to initialize the entire system by spawning all other processes and provides a user- interface to run control. (A copy of the current user's manual containing additional details is available via anonymous ftp as file pub/dragon/dcproto/README from ftp.slac.-stanford.edu.)

Event and other data is organized into C (language) structures transferred to other processes or written to mass storage using "cheetah". See Fig. 3. Cheetah[6] is a C library package which provides the capability to read and write C structures to and from standard I/O channels (*e.g.* disk, tape, or sockets), a feature missing from the ANSI C language. A major feature of the cheetah package is the self-documenting dataset. When

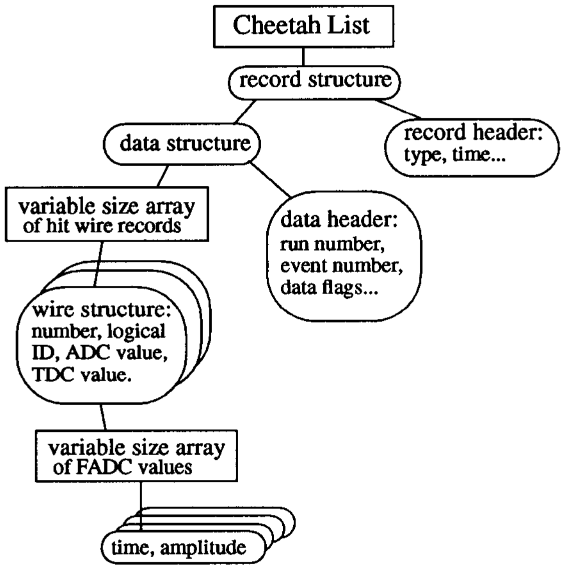

Cheetah List
record structure
data structure
record header: type, time...
variable size array of hit wire records
data header: run number, event number, data flags...
wire structure: number, logical ID, ADC value, TDC value.
variable size array of FADC values
time, amplitude

Fig. 3. Data Structure using C structs

data is written it is accompanied by a complete description of the structures, including user comments from the header file defining the structure. With this feature one can, for example, read an old cheetah-format file and determine its format and content solely from the included dictionary information. Cheetah also provides a number of other conveniences making it easier for programmers to organize data, one example being a list structure which knows it size and automatically expands when new items are added.

IV. PERFORMANCE

Tests have been performed to measure the elapsed time for reading the various CAMAC digitizers. Currently, we achieve a sustained transfer rate of 700-1000 bytes/s, being limited to the problem mentioned above. The CAMAC transfer rate could be improved by replacing either the crate controller with one supporting automatic subaddress incrementing or by replacing the digitizer modules with ones supporting block transfer reads. The ultimate limitation is, of course, that of GPIB which is of order 1 Mbyte/s. A similar test was performed between the workstation and the VME FADC modules resulting in a sustained transfer rate of ~2 Mbytes/s. Another test between a VME memory module and the workstation result in only 0.9 Mbyte/s. National Instruments claims a maximum transfer rate between the RS/6000 and MXI of approximately 6 Mbyte/s. It is not known where the bottleneck occurs in this case.

V. COMMENT

Various problems and limitations surfaced during the development of this system. For example, we experienced a curious incompatibility problem between the vendor-supplied VME/VXI driver software and the X window server which caused the machine to crash. Fortunately, the vendor has been relatively helpful in isolating the problem and developing a work-around for it. Another example concerns the limited speed of CAMAC operations which is due to a "generational mismatch" between the "old" digitizer modules and our "new" crate controllers.

Although we had not intended to design and implement a full "buffer manager", it is clear that we would benefit from such a tool. This would provide a smoother and more general solution to the problem of providing controlled access to shared memory resources, particularly in the context of multiple data consumers and producers. The use of sockets has emphasized the need to code their use with care and has reminded us that they are truly point-to-point. A system of distributed shared memory would be a very useful tool.

VI. SUMMARY

We have developed a working data acquisition system based entirely upon Unix workstations and using native Unix tools. This system supports read-out, control, logging, and analysis for a small prototype drift chamber, fulfilling all of the specified functional requirements. The system was specifically designed to exploit many of the underlying functions needed to implement a full-scale on-line system for a future SLAC B Factory detector. We plan to continue the development of this project by redesigning it in the object-oriented language, C++, and by adding an X window graphical user interface.

REFERENCES

[1] *Workshop on Physics and Detector Issues for a High Luminosity Asymmetric B Factory at SLAC*, SLAC-373/LBL-30097/CALT-68-1697 (March 1991), pp. 575-596.

[2] *B Factories: The Stare of the Art in Accelerators, Detectors and Physics*, SLAC-400 (November 1992), pp. 543-550

[3] D. Aston, et al, *Computing for a B Factory*, SLAC BaBar Note 82 (15 June 1992)

[4] T. Glanzman, *Real-Time on a Standard Unix Workstation?*, SLAC-PUB-5929 (September 1992)

[5] A. Boyarski, P.Burchat, *Drift Chamber Status*, SLAC BaBar Note 76 (8 Jan 1992)

[6] P. Kunz and G. Word, *The Cheetah Data Management System*, SLAC-PUB-5930 (September 1992)

[7] XGKS is available via anonymous ftp from unidata.ucar.edu as pub/xgks.tar.Z