Unix-based Data Acquisition system for $(g-2)_\mu$

I. Kronkvist, P. Cushman, L. Duong, C. Timmermans[1]

University of Minnesota, Minneapolis, Minnesota 55455

Abstract. The Data Acquisition system used in the new BNL $(g-2)$ experiment is based on VME and CAMAC. VME-based CPU's, running the VxWorks operating system, read out the detector electronics and a central computer, running Red Hat Linux 5.0, controlled the data flow and monitoring. The data acquisition software is based on UNIDAQ, developed initially for the SSC project. It consists of several programs, each performing a well defined task, and communicating with each other through message queues and shared memory. This functionality is expanded to a multiple CPU environment using RPC's.

INTRODUCTION

The $(g-2)_\mu$ magnet is a monolithic "C" magnet, 7112 mm in radius, with a magnetic field of 1.451 T, excited by four superconducting coils [1].

A proton beam from the Brookhaven AGS is directed at a nickel target to create a secondary beam of pions. The pions with the desired momentum are selected and transported down a 116 m beam line, in which some of them decay to muons. The muons pass through momentum selection slits and are injected into the $(g-2)$ storage ring. In the ring, the muon's decay electrons will be detected by 24 calorimeter stations, placed around the ring in the "C" gap of the magnet.

The beam from the AGS is cycled every 2.6 s. Each cycle can fill the storage ring up to 12 times. In 1997 the number of fills per cycle was 8, and in the 1998 run it was 6. The fills spaced 30 ms apart. This gives about 2.3 s between the last fill in a cycle and the beginning of the next cycle.

DETECTORS AND READ-OUT ELECTRONICS

Analog signals from the 24 lead-fiber scintillator calorimeters provide both energy and time information. A Wave-Form Digitizer (WFD) [1], which is VMEbus slave module, digitizes and stores analog input data at a rate up to 200 MHz. It is

[1] Currently working for University of Nijmegen, Nijmegen, The Netherlands

basically a scope trace with 2.5 ns sampling rate. The same signals and their sum are sent to CAMAC ADC and TDC modules for calibration purposes. The sum is also sent to a Multi-hit Time-to-Digital Converter (MTDC) [2] which provides the timing information. The MTDC has a sampling rate of 400 MHz. The calorimeter read-out is shown in figure 1.

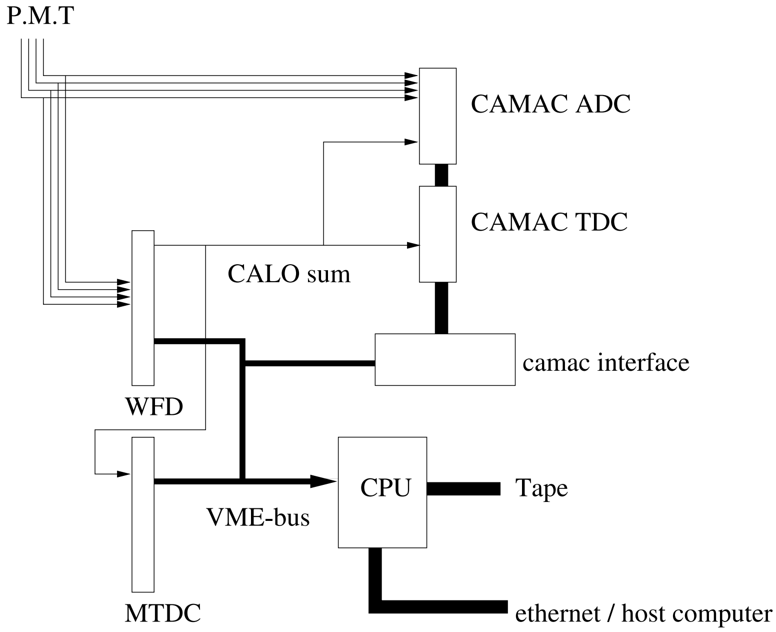

FIGURE 1. Read-out of PMT signals from the calorimeter stations.

On seven of the calorimeter stations, there are Front-Scintillating Detectors (FSD), which consist of 5 radially oriented scintillator fingers covering the entire front of the calorimeters. The FSD's are mainly used for pile-up rejection, and are read-out by MTDC's.

Five of the calorimeter stations contain a Position Sensitive Detector (PSD), which is a scintillator hodoscope composed of one x- and one y-plane, each with 7 mm segmentation. There are 32 x-tiles and 20 y-tiles in the two planes. The PSD provides rejection of spatially separated hits which occur within the same time window. The PSD x- and y-information is read out by custom designed electronics, and the anode sum of both planes is read out by a MTDC. Four of the PSD's are triggered by the calorimeter on which they are mounted, and the fifth PSD is triggered using a coincidence of the x- and y-planes.

Electronics set-up

The DAQ system is composed of a total of six 9U and one 6U VME crates plus three CAMAC crates, figure 2. Each of the 9U crates also contain a MVME167 Front-End CPU (FE) which configures the electronics and reads out their data buffers after each AGS fill. This data is stored in a local memory buffer on the CPU. The 6U crate holds the main Event-Builder CPU (EB), a VMIC7588 pentium processor, and also contains a 2917 KBUS interface for reading out the CAMAC calibration crates. CAMAC is read out after each AGS fill. The 9U crates are connected to the 6U crate through two MXI-2 daisy chains, three 9U crates in each chain. After the last fill in each AGS cycle, the EB reads out the data buffers of the FE's over these MXI-2 chains. It then concatenates the event and writes it to a SCSI mounted DLT tape drive.

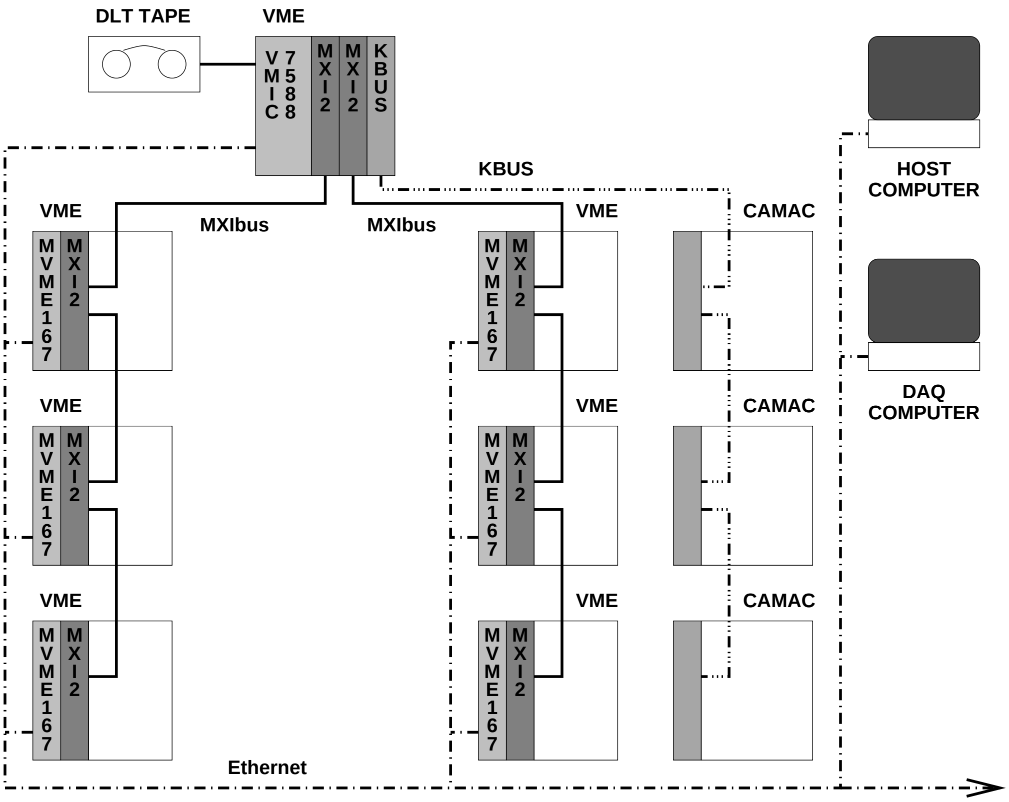

FIGURE 2. VME based DAQ system.

At boot-up, the 7 CPU's load their operating system over an ethernet link from a host computer. The disks on the host and DAQ computers are NFS mounted on each CPU and all processes necessary for the DAQ are automatically spawned.

UNIDAQ

UNIDAQ [3] is a distributed Data Acquisition system for UNIX-flavour platforms. It consists of many processes that can run on one or more machines, communicating with each other through message queues and shared memory. Data is passed from process to process using a buffer manager, NOVA [4]. Critical event variables are saved in the shared memory and stored on disk.

The UNIDAQ programs are based on a template which handles messages and commands. There is a `Run Control` process that handles distribution of the commands to the processes affected by them. User interaction is handled using an `Operator` process, which is a graphical user interface, based on TCL/TK, that allows a user to start/stop runs and change event variables.

For the $(g-2)$ experiment, the UNIDAQ system is run on three different platforms:

- The MVME167 Front-End CPU's which run VxWorks 5.2.
- The VMIC7588 Event-Builder CPU which runs Tornado 1.0.1 (VxWorks 5.3).
- The DAQ computer, a dual pentium machine that runs Red Hat Linux 5.0.

A schematic view of the different processes is shown in figure 3. These are also explained below.

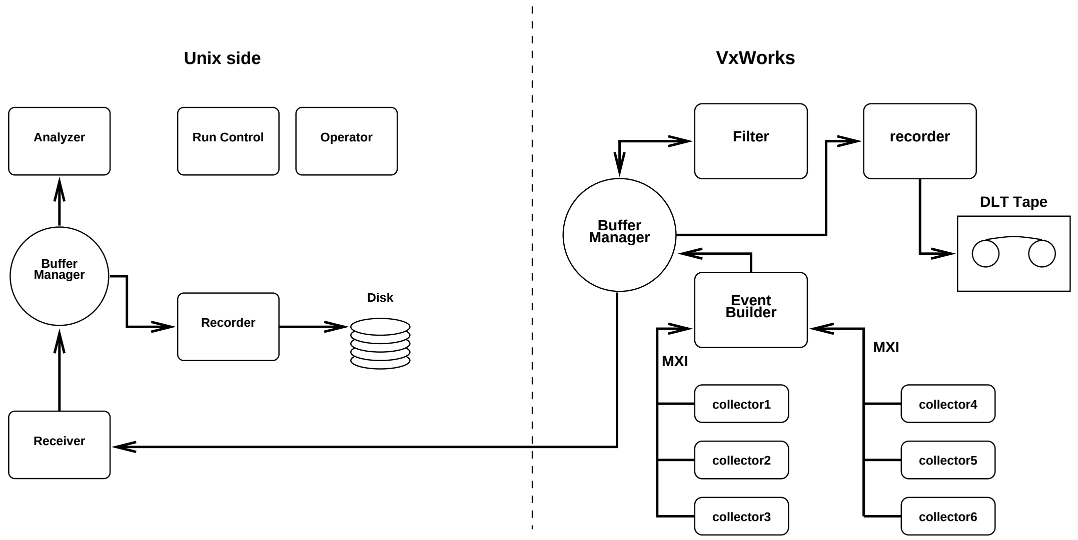

FIGURE 3. The UNIDAQ processes.

VxWorks processes

For reading out the 9U crates, the Front-End CPU's each run a version of a `collector` process. This collects the data from the electronics modules according to a downloaded configuration file. The data is then stored in a local memory buffer, and a pointer to this memory location as well as the size of the buffer is supplied to the main Event-Builder CPU.

The EB also runs a `collector` process which reads out CAMAC event data after each AGS fill. At the end of each AGS cycle, the EB then reads out the memory buffers of each FE, and both CAMAC and VME data is concatenated to build an event, which is then sent to an event buffer. A `recorder` process takes the events from the buffer and writes them to a SCSI mounted DLT tape drive. There is also a `filter` process which filters out a certain percentage of the VME data and sends it back to the buffer to be received by the DAQ computer.

35 μs before each fill, a signal is sent from the AGS that beam is on its way. This sends a start signal to the WFD and MTDC modules, which begin digitizing their input. After 600 μs, a stop signal is sent out to the electronics, and they stop their digitization. This stop signal is also fanned out to the FE's and triggers their 9U read-out sequenze. A slightly delayed stop signal is sent to the EB to trigger its read-out sequenze. Since the data for all fills in a cycle must be stored, the Event-Builder must keep track of the fill number in order to read out the Front-End CPU's at the end of the cycle. In case all fills are not present in a cycle, there is a trigger time-out of $\approx$ 40 ms for the EB. If no fill has arrived within this time, the fill is considered non-present and the fill-counter is incremented to check if there should be more fills to come in the present cycle.

Linux processes

The DAQ computer contains the `Run Control` and `Operator` processes. In addition, it also runs a `receiver` process which receives a certain filtered percentage of the data over ethernet, from the VMIC7588. The receiver puts these events in a local event buffer. It has a `recorder` process that takes these events and writes them to a data file on disk. It is possible to run an `analyzer` process on the DAQ machine to check the quality of the data. This program exists in two different versions; one that reads the events directly from the event buffer and gives an online quality check, and one that reads a data file and thus gives the opportunity to compare data from different runs.

Configurations of UNIDAQ

All nodes present in the UNIDAQ system must be defined in a text file, listing the machine name and its operating system. This file is used by the `Run Control` process for distributing commands to processes on other nodes. `Run Control` needs to know both the name of the machine to which it should send commands and the operating system on that machine. The latter is used for formatting the command in a way that the receiving operating system can interpret.

A TCL/TK based utility program was created to configure the read-out electronics in both the VME and CAMAC crates. This program is used for defining addresses and slot numbers for the electronics modules, as well as the input signal to the channels of each module. For some of the electronics modules, there is also the possibility to set thresholds and offset values for the input channels. The configuration is saved in seven files, one for each of the seven VME crates. At the beginning of a run, the configuration files are automatically downloaded to the crate controllers, and only the modules defined in these files will be read out during data taking.

The configuration files for each run are saved in a database, and subsequently used in the offline reconstruction of the data for defining which parts of the experiment were read out for a specific run.

REFERENCES

1. Muon $(g-2)$ Design Report, 3rd edition 1995, unpublished.
2. "A Deadtimeless Multihit TDC for the New g-2 Experimen", W. Earle *et al.*, Fourth International Conference on Electronics for Future Colliders, LeCroy Corp, 1994
   "A New Multihit Digital TDC Implemented in a Gallium Arsenide ASI", E. Hazen *et al.*, IEEE Trans Nuc Sci, Vol. 40 No. 4, p. 1125, August 1994
3. UNIDAQ source code and manuals can be obtained from the world wide web at **http://mnhepw.hep.umn.edu/t̃immer/personal.html**
4.