Upgraded DAQ System for the HyperCP Experiment

C. G. White [a,*], T. D. Jones [b], W. R. Luebke [a], Y. C. Chen [d], K. C. Cheng [d], W. S. Choong [c,b], D. M. Kaplan [a], K. Nelson [e], H. A. Rubin [a], B. Turko [b], P. Zyla [b]

[a] *Physics Division, Department of Biological, Chemical, and Physical Sciences, Illinois Institute of Technology, Chicago, IL 60616*

[b] *Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720*

[c] *Department of Physics, University of California, Berkeley, CA 94720*

[d] *Institute of Physics, Academia Sinica, Nankang, Taipei, Taiwan*

[e] *Department of Physics, University of Virginia, Charlottesville, VA 22901*

Abstract

The 1999 run of Fermilab E871 (HyperCP) recorded 80 Terabytes of data on 19,000 8mm tapes with a typical trigger rate near 100 kHz. The readout electronics digitized data from 21,000 input channels with a fixed deadtime of 1.7 μs resulting in a typical livetime of 80%. The typical instantaneous rate of data generated by the HyperCP spectrometer was 36 MB/s. These data were written to tape at a sustained rate of 27 MB/s.

Key words: Hyperon, HyperCP, E871, CP Violation, DAQ, acquisition
PACS: 29.90.+r, 29.30.Aj

1 Introduction

The origin and nature of CP violation is a mystery with CP violation having been observed only in decays of neutral K mesons [1]. Therefore, it is of interest to search as sensitively as possible for CP violation occurring outside the kaon sector [2]. The HyperCP experiment [3] aims to achieve 10^{-4} sensitivity to

* Corresponding author.
Email address: whitec@iit.edu (C. G. White).

possible hyperon CP violation. This is two orders of magnitude beyond the most sensitive current limits [4].

Hyperon CP violation can be studied most sensitively by comparing the Lee-Yang [5] α parameter characterizing the parity violation in the decay of a hyperon species with that for the corresponding antihyperon species [6]. To do this, HyperCP will reconstruct $\sim 10^9$ hyperon and antihyperon decays in the cascade decay modes $\Xi^- \to \Lambda^0\pi^- \to p\pi^-\pi^-$ and $\overline{\Xi}^+ \to \overline{\Lambda}^0\pi^+ \to \overline{p}\pi^+\pi^+$. This data sample will also give an interesting range of sensitivity to various rare decays [7], and to possible direct CP violation in charged-kaon decay as well [8].

In order to achieve our goals, a high-rate, high-speed data acquisition system was built and operated by the HyperCP collaboration. The first run of HyperCP took place between March and September 1997. A second run collected data from June 1999 through January 2000. Between the first and second runs the spill cycle was changed from 20 seconds of slow spill every 61 seconds to 40 seconds of slow spill every 81 seconds. This change necessitated an upgrade of the data acquisition system. The original DAQ system has been described elsewhere [9]. Here we detail the readout electronics and the DAQ hardware and software employed during the second run of HyperCP.

2 HyperCP Spectrometer

The HyperCP spectrometer is shown schematically in Figure 1. An intense unpolarized beam of Ξ^- ($\overline{\Xi}^+$) hyperons was produced at $0°$ by 800 GeV protons striking a metal target, with the secondaries momentum-selected by means of a curved magnetic channel set to 160 GeV/c with a 25% FWHM momentum bite. Following a 13 m evacuated decay pipe the hyperon decay products were detected in a high-rate magnetic spectrometer using multi-wire proportional chambers (MWPCs). The necessary rate capability was determined by the $\approx$15 MHz of charged particles emerging from the channel. To switch between hyperon and antihyperon running, the polarities of the hyperon and analyzing magnets were reversed, selecting either positive or negative secondary beam. To maintain equal rates in the detectors, different target lengths were used for the two modes. This approach resulted in nearly identical acceptance and efficiency for hyperons and for antihyperons, essential for minimizing biases.

A simple trigger was employed to minimize potential systematic effects that might mimic a CP asymmetry. Our primary physics trigger ("cascade trigger") required a coincidence of at least one particle on the right of the spectrometer with at least one on the left. This was done using the "pion" and "proton" hodoscopes, which were located sufficiently far downstream of the analyzing magnets that the proton and pions from the hyperon decay were all separated from the secondary beam. To suppress low-energy backgrounds, the trigger also required a minimum energy deposit in the hadron calorimeter. Nevertheless the trigger rate was dominated by background due to interactions of beam particles with the spectrometer material. During slow spill extraction, the most probable rate of cascade triggers was 50 kHz during positive mode, and 60 kHz during negative mode. The most probable total trigger rate, including all calibration and secondary physics triggers, was 90 kHz during positive running and 95 kHz for negative running. The typical instantaneous data rate generated by the HyperCP spectrometer was 36 MB/s.

The muon detection system consisted of proportional-tube planes interspersed with iron absorber. Muon triggers were formed with signals from four planes of scintillation hodoscopes located between and behind the absorbers. These data provided sensitivity to rare decays with muons in the final state.

3 Overview of the Data Acquisition System

The acquisition of so large a hyperon sample required a highly capable data acquisition system (DAQ), designed for 100 kHz trigger rate and greater than 23 MB/s sustainable data rate to tape. Our system was an upgrade of the system used during the 1997 run [9]; however, the details differ dramatically due to substantial hardware changes and entirely new software. The major features included:

- High speed readout system;
- 960 MB data buffer (VDAS) [10];
- 68040-based Motorola MVME167 processors;
- Fast yet robust DAQ software;
- 1 MB/s Exabyte 8705 tape drives [20];
- Silicon Graphics workstations running Fermilab DART software [11].

Raw data were digitized by a latch system and passed to the VDAS along five parallel data paths. Event building was accomplished using three standard 6U VME crates operating in parallel, each with three MVME167 processors running the VxWorks 5.3.1 operating system [12]. Within a crate, on command from one MVME167 processor, an event buffer interface (EBI) read an event fragment from the first of five parallel VDAS modules, converted the protocol to VME and output the data onto the VME backplane. The MVME167 processor then stored the data in an internal buffer. Each path was thus read with the five fragments assembled within the processor buffer to form a complete event. Assembled events were written to Exabyte 8705 tape drives via MVME712 interface modules.

4 Front-End Digitization System

The major task of the front-end system was digitization and momentary storage of data from the 20,394-wire MWPC system. In addition it read out the trigger hodoscopes, calorimeter, and muon system. To achieve sufficiently low deadtime at up to 100 kHz average event rate, all systems satisfied a digitization and readout time specification of $\lesssim 2\,\mu$s/event.

4.1 Latch System

Readout of the MWPCs, calorimeter, muon system, and trigger hodoscopes was conducted using an adaptation of the Nevis Laboratories MWPC Coincidence Register (CR) system [13]. Having already built and operated nearly 10,000 channels of this system in Fermilab Experiment 789 [14], we chose for reasons of cost-effectiveness to augment that existing system for HyperCP rather than to build a new system from scratch. The HyperCP latch system consisted of 37 crates. Each was built using CAMAC-standard card cages housing inexpensive two-layer printed circuit boards communicating via a custom ECL backplane. Each crate accommodated up to twenty-three 32-channel latch cards plus one Buffered Encoder board used as a crate master for readout and sparsification of the data. These crates were divided among 5 parallel data paths balanced (as well as possible) based on the average number of hits per event. Each path consisted of a variable number of latch crates chained serially to a Compressor module, followed by an interface module linked through an optical fiber to a VDAS data buffer (Figure 2). In total, 654 latch cards were employed instrumenting 20,928 channels. Except where mentioned, the latch system was designed using Motorola 10k-series ECL logic.

4.1.1 Latch Crates

The latch cards (Figure 3) accepted differential-ECL inputs via a front-panel card-edge connector and received gate signals from the backplane. Inputs were conveyed to the latch card front panel via Ansley multiconductor ribbon cables [15], which were sufficiently oversized in length to provide trigger delay. A Prompt Reset signal (not used in HyperCP) supported multiple trigger levels. The Group Reset signal was used to reset one latch card at a time during the readout cycle (described below). To allow easy interchange of modules, card addresses were hard-wired into the backplane by extraction of pins from the (card-edge) backplane connectors. Each latch card drove all five address signals, but only those whose pins were intact were asserted on the backplane.

In the original ca.-1970 design, token passing was used within the crate to pass control from one latch card to the next. In 1980, a fast readout scheme was implemented for Fermilab E605 [16], in which token passing was eliminated in favor of priority encoding via a Fast Carry Card (FCC) mounted to the crate backplane (Figure 4). The FCC received Carry-Out ($\overline{\text{Empty}}$) signals in parallel from all latch cards having non-zero data and issued a Carry-In ($\overline{\text{Enable}}$) to the first such card.

Gate signals were fanned out to the individual slots using gate fanout boards mounted on the backplane. Two versions of gate fanout were used, depending on the age of the crate in question. The newest crates had twin gate fanout boards mounted at either end of the FCC which were also used to terminate the backplane data bus. Older versions had a single gate fanout board mounted below the FCC. In either case, input NIM gates were converted to ECL and distributed to individual slots. The twin fanout design allowed for common gating of 12 adjacent slots, or, when more fine tuning was desired, three groups of 4 adjacent slots. The older design accepted three input NIM signals, fanning each out to eight slots. Although HyperCP used a 20 ns gate width, gate signals as short as 10 ns have been used reliably. Following receipt of gates, the highest-priority latch card with nonzero data put out its 32 data bits in parallel on the crate backplane.

The Buffered Encoder (BE) directed readout of the latch cards through the backplane and fast carry logic. Readout was initiated with the receipt of an asynchronous Start pulse from the trigger system, which was sufficiently delayed to ensure gating and settling of signals along the backplane. The amount of delay varied depending on the time required for the detector signals to arrive at the latch card. At the next rising edge of the readout clock, distributed from a central clock module to the entire latch system, the backplane data lines were latched within the BE. A Group Reset pulse was then sent by the BE along the backplane resetting the enabled latch card. The next card with data was then enabled through the fast carry logic and asserted its data on the backplane. The latched data bits, having been converted to TTL logic levels, were then entered into 256-deep, 7ns CMOS FIFOs coincident with the latching of the next backplane data word. A block diagram for the BE module is shown in Figure 5.

The clock period was specified by the time required to issue a Group Reset, to complete the resetting of the enabled latch card, to enable the next latch card via the Fast Carry logic, and to allow for settling of new data on the backplane. Recently manufactured crates have been shown to run reliably with a 50 ns clock period; however, the HyperCP latch system relied upon an assortment of crates manufactured at different times for various experiments. A clock period of 85ns was chosen for the system based on the reliable performance of the slowest crate. The total number of clock cycles required to read out a crate equaled the number of latch cards with data plus one, the extra cycle being used to generate a $\overline{\text{Done}}$ pulse and to insert a Complete word into the FIFO. The Complete word was used for event synchronization between crates and as an event separator within the FIFO.

Synchronization between crates was ensured through the use of a 4-bit Event Synchronization Number (ESN) recorded as part of the Complete word. The ESN came from the ESN Source Module which contained a 4-bit counter incremented by each live trigger. The ESN was distributed to the Buffered Encoders over ribbon cable and latched by the BE with each Start pulse. The ESN was used to certify that event fragments from different crates pertained to the same trigger.

The presence of data within the FIFO activated encoding of hits by a fast FPGA. Encoding saved data space since the average hit density throughout the latch system was less than 1 hit per 100 channels. Each active (i.e. non-zero) bit was encoded into a 16-bit data word with 5 bits used for the channel number, 5 bits for the slot number, 4 bits for the crate number, and 2 bits unused. The FPGA output was converted back to ECL and stored in the BE output register. Processing proceeded unless Hold was asserted by the immediately upstream BE or (in the case of the first BE in the chain) the Compressor Module.

In addition to data readout with sparsification, we recorded certain information (e.g. bits recording the state of the trigger logic and the calorimeter's ADC information) that was not sparse. The Buffered Encoder was designed to allow also operation without sparsification. Two BEs were so configured, reading out 56 bytes of fixed-format data. To ensure that the non-encoded latch cards sent their data onto the backplane bus regardless of the state of their inputs, one input bit out of each 32 was forced on, leaving 31 bits per latch card available. This was necessary since there was no addressing information in the event record for the non-sparsified data, thus they were identified only by their position and sequence in the event record.

4.1.2 Multi-crate operation

The output cable and transmission protocol of the BE obeyed the Nevis Processor Bus (P-Bus) standard [17]. The single-ended-ECL P-Bus cable included a 16-bit data space and a 4-bit name space as well as control lines. The data space was used to transmit the 16-bit data word, while the name space identified the Buffered Encoder (i.e. crate).

Each P-Bus segment connected two Buffered Encoders, with the "first-in-chain" BE connected to the compressor module (discussed below). Consequently, each BE contained both an input and output register, as shown in figure 2. Upon initialization all BEs asserted P-Bus Hold backward along the

BE chain such that only the first-in-chain BE was enabled to output data. Following a trigger, all BEs placed the first data word (or Complete, as the case may be) into their output register; however, all but the first-in-chain was then held. Data words passed into the compressor module, one per clock cycle. After outputting a Complete word, the enabled buffered encoder switched to "pass-through" mode, releasing Hold on the next BE in the chain. As the name implies, a BE in pass-through mode simply transferred words from its input register to its output register. Pass-through mode terminated when the Complete word from the last crate in chain (Last Name Complete or LNC) was passed to the output register.

The primary function of the Compressor Module (CM) was event aggregation. The CM aggregated events into "super-events" by suppressing complete words and replacing the LNC with an 8-bit event separator. The LNC from the last fragment was retained and served as the ending boundary for the super-event. The number of events per super-event was set by a switch on the CM with allowed values ranging from 1 to 127. Event aggregation allowed more efficient use of the VME-backplane bandwidth during event building (as detailed below). HyperCP aggregated ten event fragments into each super-event fragment. As a check of readout integrity, the CM also examined the ESN from each complete word and halted the readout should the ESNs from consecutive crates differ.

The CM was also designed to reduce the event size by compressing consecutive data words with an arithmetic difference less than 192 into 8-bit offsets with respect to the previous data word or offset (offsets between 192 and 255 were dis-allowed in order to differentiate compressed data words from standard data words). There was no limit, other than crate boundaries, to the number of consecutive compressed hits. Compression reduced the size of the encoded data by 40%. The overall data compression, including header words and unencoded data (which were not compressible), was 30%. Compression was pursued in anticipation of higher trigger rates and as a hedge with regard to the final data-to-tape rate; however, for the trigger and data-to-tape rates observed during the run, compression of the raw data would not have appreciably increased the total number of events collected. Consequently, to avoid unnecessary complexity, the compression feature of the CM was not used during the bulk of HyperCP data collection.

4.1.3 Calorimeter ADC System

The calorimeter photomultiplier (PMT) signals and several monitor signals were digitized by twelve fast charge-integrating analog-to-digital converters (QDC). A dynamic range of about 10,000 was needed for the calorimeter signals to allow the small response produced by muons as well as the much larger response from high-energy Lambda decay protons to be recorded. The muon response was used for calibration.

The QDCs were implemented as single-width CAMAC modules having two channels per card, mainly to stay within the power dissipation [1] allowed per CAMAC slot. In the block diagram of the module (Figure 6) the two independent charge digitizing channels are shown along with the readout control logic. Each channel had individual front panel connectors for gate and input signals. The front panel also had connectors for the output data bus and the control bus via standard IDC flat ribbon cable. This permitted the daisy-chained readout of several modules at speeds up to 40MB/s. No CAMAC features were used other than the power provided by the crate.

At the beginning of data collection and after the readout of each event, the modules had to be reset by the control bus to be ready to accept a new event. The input PMT signal was typically a negative voltage pulse having a peak amplitude of 1.5V which was integrated for the duration of the gate (80ns). The trailing edge of the integration gate was used (after a short delay to allow for settling) to strobe a 14-bit 3 MHz analog-to-digital converter [18]. In response the ADC sent an End-of-Conversion pulse which was used to set the Data Ready latch. The output of this latch inhibited further gates from being accepted until the module was reset from the control bus. Several modules were read out in "daisy-chain" fashion (serially) as shown in Figure 7. The OR of the two Data Ready latches in a QDC module was wire-OR'ed with all modules on the control bus to form the REQ line. REQ was used to inform the DAQ system that data were ready to be read out. The DAQ system commenced readout by asserting the Read line of the control bus. The Read Enable input of each QDC module had two jumpers, J1 and J2, which allowed the module to be enabled from either the control bus Read line or from a front panel auxiliary input. Only the module to be read first took its enable from the control bus; the other modules received the enable from the preceding module and passed it on to the next module in line after readout.

When a module received Read Enable, its two-bit address counter was initially reset. The gate G1 asserted the data of channel 0 onto the data bus. The DAQ system, upon storing this data, generated a clock pulse to advance the counter which closed G1 and opened G2, thereby asserting the data of channel 1 onto the bus. Although all QDC modules received the clock pulse on the control bus, only the enabled module responded. After a second clock pulse, the second bit of the counter was set, which was interpreted as Read Enable by the next QDC module in line. The address counter remained locked in this state until all subsequent modules were read out and the DAQ system sent a reset pulse on the control bus.

[1] each QDC dissipates 1.8W

The QDCs were read out to the DAQ system through a PWC latch crate employing a special programming of the Buffered Encoder (mentioned above) in which the data were buffered but not encoded by address. Since the QDCs produced only 14 bits of data the modules were grouped in two parallel chains to take full advantage of the 32-bit-wide backplane and thereby reduced the number of readout clock cycles by two. An Unencoded Source Module (USM, see Figure 8) implemented the interface between the readout protocols of the QDC control bus and latch crate backplane.

4.2 Readout Speed and Livetime

Sixteen physics and calibration triggers were used during the 1999 run of HyperCP, most of which were prescaled. To prevent the mixing of independent, closely spaced events, triggers were inhibited during the latch system readout. Consequently, the experiment was “dead” to new events during the readout process. The deadtime for each event lasted from a trigger pulse until the arrival of the trigger release pulse. This process was managed by the Trigger Inhibit Module (TIM). After latching a “live” trigger from any of the 16 trigger pre-scalers, the TIM issued a short output pulse which was subsequently fanned out, delayed, or shaped (as the case may be) for use as latch gates, Start pulses, and generic trigger pulses. Triggers lost to deadtime were tracked through the use of inhibited and uninhibited trigger scalers.

The latch gates were shaped and delayed to meet the individual requirements of each detector system. The cable delays for the MWPC latch gates varied from 32 to 132 ns depending on the length of the signal cable and position of the detector within the spectrometer. The Start pulses for these crates were timed to arrive at least 50 ns after the gate to allow for signal propagation and settling of the backplane levels. On average, 10 clock cycles were then required to record all MWPC signals within the BE FIFOs with small variations event to event. Consequently, the MWPC readout was complete (on average) after 1100 ns (including an average of 42 ns to synchronize to the readout clock after Start).

Signal formation for the proportional tubes of the muon station was substantially slower than for the MWPCs and required a gate width of 300 ns (as compared to 20 ns) to achieve high latching efficiency. This increased overhead was compensated by reduced hit densities which required fewer readout clock cycles. Readout of the muon crates was complete (on average) after 1200 ns.

Of the two crates with no sparsification, one read out the calorimeter ADCs while the other read out trigger bits and other event-related, non-sparse data.

The trigger-bit crate produced 7 "slots" worth of data for every event. Including gate delays, this crate required 1000 ns to complete readout into the FIFOs of the buffered encoder. The ADC crate also produced 7 slots worth of data; however, additional delay was required to accommodate setup and conversion times. Following a trigger, 1550 ns was required to complete the readout of the ADC data into the FIFOs of the BE.

When the Carry-Out line on the backplane of the latch crate went low (i.e. no more data), the buffered encoder generated a Done pulse. These pulses were collected by the Trigger Release Module (TRM) (Figure 9). Each active channel of the TRM corresponded to a particular buffered encoder. The TRM generated a System Busy logic level by ORing individual BUSY lines for each crate. A pulse from the trigger system set all BUSY lines within the TRM. These lines were individually reset by the Done pulses from the BEs. A trigger release pulse was issued to the TIM, following Done pulses from all BEs. Including all cable delays and propagation times, triggers were released 150 ns after the last Done pulse issued by a BE. In general, the total deadtime per event was fixed by the ADC crate at 1.7 μs. Fewer than 5% of events possessed a deadtime greater than 1.7 μs. The observed livetime distribution is shown in Figure 10. The long tail toward small livetimes was attributed to system errors requiring initialization of the electronics (e.g. ESN sequence error or hardware errors).

4.3 VDAS Spill Buffer and Interfaces

The digitized data from the detector and trigger systems were accumulated in the VDAS buffers [10]. These accepted 32-bit words at speeds up to 100 ns per word using the RS-485 signal protocol. For the 1997 run of HyperCP, to interface between the VDAS system and the Processor Bus, we built P-Bus to VDAS Interfaces for each of the latch system data paths [9]. The interfaces were reused with slight modification during the 1999 run and served two main functions:

(1) stack pairs of 16-bit words from the Processor Bus into 32-bit words;
(2) precede each super-event fragment with a 32-bit header word by storing the data temporarily in an 8192-word FIFO and counting bytes until Complete.

The interface accepted input data from the compressor module via the Processor Bus. The data were then latched, converted to TTL logic levels, and stacked into 32-bit words inside 8192-deep CMOS FIFOs. Upon latching an LNC, the interface generated a 32-bit header word containing the path number, the ESN (from the complete word), a data flag (distinguishing 1999 data from 1997 data), and the total byte count for the super-event fragment. The header was put in front of the stacked data, which were then output at one 32-bit word every other clock cycle, while the interface accepted data from the next super-event fragment. The LNC word was not passed to the output stream. The interface control logic was programmed into three fast CMOS PALs.

Due to the long distance (≈ 150 meters) between our electronics room and our control room, the data from the P-Bus/VDAS interfaces were transmitted to the VDAS via high-speed optical links [19]. Five optical fibers (one per data path) transmitted data from the electronics room to the control room, and one returned the Near Full condition (the logical OR of the Near Full signals from the five VDAS buffers) from the VDAS to the electronics hall. The Near Full signal was used to veto triggers.

5 Data Collection and Run Control

The super-event fragments from the VDAS were assembled into complete super-events and written to tape by the VME event-building and tape-writing system (Fast DAQ or FDA). In addition to the event data stored within the VDAS, spill information, such as scaler data, was collected and recorded by the Spill Monitor process (Slow DAQ or SDA). These systems were managed by the DART Run Control (DRC) program running on a Silicon Graphics Indy workstation. The DRC program employed a user-friendly Tcl/Tk interface, accepting commands in the form of a mouse click on the command button of interest. Experimental operations were streamlined such that only five user commands were needed at the DRC level: Initialize, Start, Pause, Resume, and Stop. Underlying these commands were a series of functions and script files that relayed commands and information to the FDA, SDA, trigger system, and latch system. To further simplify DAQ operations, and protect the user from inadvertent errors, the DRC program controlled which commands were available at any given moment. For example, the Start command was not available until after the Initialize command had been executed.

5.1 VME Event-Building and Tape Writing

The goal for the event-building and tape-writing system (FDA) was a sustained data-to-tape rate of at least 23 MB/sec, a 75% increase over the 1997 rate. Stability and rapid error recovery were also desirable characteristics. The re-design goals mandated efficient use of the VME hardware combined with fast yet robust MVME167 processes. Except where mentioned, the DAQ software was completely re-written using the C programming language and compiled on the SGI host computer.

5.1.1 Hardware

The FDA hardware consisted of three identical VME 6U crates, each housing three MVME167 processors and five Event Buffer Interface (EBI) modules. Mounted along the backplane of each crate were three MVME712 communication modules, one for each CPU. The SCSI port on the MVME712 was used to connect each CPU to three Exabyte 8705 tape drives (Figure 11).

The MVME167 processors were purchased commercially, while the Event Buffer Interface (EBI) was a custom VME design. The EBI provided VME access to the VDAS data. Each VDAS output was bused to an EBI in each of the VME crates. The bus protocol was RS-485. Bus access was restricted to one EBI at a time through the use of an electronic token passed from one EBI to the next through front-panel lemo connectors. If in control of the bus and addressed by a CPU for a DMA transfer, the EBI sent a strobe pulse to the VDAS requesting data. Data on the VDAS bus were latched by the EBI 192 ns later. The EBI converted the data to VME protocol and output the data along the VME backplane to be stored by the CPU within the event building buffer. This cycle continued until the DMA request had been satisfied. The DMA setup time was approximately 10 μs; consequently, small DMA transfers were inefficient. Event aggregation improved overall efficiency by increasing the size of the average event fragment from 100 bytes to 1000 bytes. Aggregation also decreased the relative variation between super-event fragments, contributing to increased efficiency.

The FDA software structure was designed around modular VME crates, allowing the system to work with from one to five crates. Figure 12 displays the processes running on the CPUs. The Data-Handler and Tape-Spinner processes were run on each CPU, with one Tape-Spinner process for each tape drive. The Data-Handlers and Tape-Spinners were loaded and started during the CPU boot sequence, along with various DART objects used to facilitate communication with the run control process running on the host SGI computer. The Listener processes, one per Data-Handler, were spawned from the DRC Start command. The Listener processes managed event building within the crate and facilitated communication with the DRC program. Communication between the Listener and Data-Handler was conducted through a socket connection, while the Listeners communicated with DRC via DART processes running on CPU 1. The Data-Handler and Tape-Spinners communicated through global variables. The Sampler process provided packets of raw data to the DART Hoist program for use by on-line monitoring programs.

5.1.2 Event-Building

Data were transferred between the EBI and Data-Handler across the VME backplane. The Data-Handler sent a strobe to the EBI causing the EBI to transfer data from the VDAS to the VME backplane. The header of the super-event fragment was read first. The fragment size and ESN number were extracted and a DMA transfer was initiated. The super-event fragment was written directly into one of three circular buffers. The process then repeated for the second, third, fourth, and fifth EBI in the crate. The assembled super-event was now complete with the data from all paths brought together. The ESNs from all path-headers were compared to verify that the individual event fragments originated from a common event. Event building proceeded as long as all VDAS buffers were non-empty (i.e. at least 47 kilobytes of data remained within each buffer).

Each circular buffer consisted of twenty-five 64-kilobyte blocks. The block size was stipulated by offline software, while the number of blocks per ring was set by the available memory on each CPU module. A block was tagged as full if at the end of a complete super-event it contained more than 52 kilobytes. Should an unusually large super-event attempt to exceed the 64-kilobyte block size, the block was discarded, along with the current super-event. Events were never written across block boundaries. Blocks were filled sequentially; however, the Data-Handler switched among circular buffers depending upon which buffer had the greatest number of free blocks.

Backplane access was restricted to a single process at a time through the use of a VxWorks semophore. A Data-Handler with available buffer space requested backplane access for event building via the Listener. The Listener took the semophore (if available) which acted as the event building key. The semophore was passed on a first-come, first served basis. The Listener and Data-Handler sat idle until the key (semophore) was received. After receiving the key, the Listener notified the Data-Handler which proceeded with event building. The key was released by the Listener following a message from the Data-Handler that all buffers were full. The DRC Pause and Stop commands were implemented, in part, by inhibiting the passing of the key.

Multi-crate operations were organized using EBI token passing. After completing a DMA transfer, the Data-Handler signaled the EBI to pass the VDAS token to the next crate. In this fashion, event building proceeded sequentially from crate to crate (super-event 1 was assembled by crate 1, super-event 2 was assembled by crate 2, etc.). Since only one EBI was addressed per crate at any given moment, the VME crates built events in parallel.

5.1.3 Tape-Writing

Each circular buffer had an associated Tape-Spinner process which ran in parallel with the Data-Handler. Each Tape-Spinner worked sequentially around its circular buffer writing data to a single tape drive. If the current block was full, the Tape-Spinner wrote that block to tape and marked the block as empty. It then proceeded to the next block. If the current block was not full, the Tape-Spinner waited indefinitely until the block was full. If a tape drive or Tape-Spinner failed, the associated buffer simply filled, and was thereafter ignored.

During normal running, the Data-Handlers assembled events at greater than 12 MB/s per crate. The Tape-Spinners were limited by the tape drives to no more than 1 MB/s. Under these conditions, event building and tape writing were evenly distributed among the CPUs and tape drives; however, at sufficiently low data input rates, it was possible for one CPU to hold the event building key indefinitely. Also, the average instantaneous data rate during beam extraction exceeded the throughput capabilities of tape writing, thus necessitating the accumulation of data within the spill buffer, as described above.

Tape-Spinners maintained a counter of the number of bytes written to tape, setting a global variable should the counter exceed 4.5 gigabytes. During event processing, the Data-Handler checked the value of this variable, reporting to the Listener when a tape was full. The Listener in turn reported to DRC, which initiated a DRC Stop command. Following a brief cleanup procedure, the Listener processes then exited.

Complex error correction code was deliberately excluded in order to maintain simplicity and speed. The Data-Handlers checked only for ESN sequence errors and header-word format errors. Tagging other forms of data corruption was left to the on-line monitoring programs. The Data-Handler notified the Listener concerning ESN or header-word errors. The Listener notified DRC, which then initiated an automatic cleanup procedure. After clearing all buffers and initializing the appropriate hardware, event processing resumed.

Much of the software was automated. The Tape-Spinner and Sampler processes were restarted (if necessary) as a part of the DRC Init command. Most data errors trapped by the on-line monitoring programs were cleared by executing the DRC Pause command followed by the DRC Resume command. CAMAC function calls tied to DRC commands were used to hold triggers, reset critical hardware, and release triggers. Buffers were automatically cleared at every DRC Initialize or Resume. The software was designed to require a CPU reboot only if a Data-Handler crashed, or if the DRC crashed.

Each spill cycle consisted of a 40-second beam-extraction period (slow spill) followed by 41 seconds of recovery. Data related to each spill cycle were monitored and recorded using the SlowDA (SDA) process. This process was a stand-alone program written using the C programming language and run on the host SGI computer. Data were read into the program [21] through a JORWAY [22] serial CAMAC to SCSI interface, as well as through a client/server application called Simple Acquisition of Data [23] (SAD) connected to the accelerator VAX cluster. The SDA hardware configuration is detailed elsewhere [9]. The DART run control program was used to send commands to the SDA. Design considerations included robustness and consistency in performance and operation. Speed was not an issue due to the long spill cycle.

The SDA maintained a series of control and log files that recorded the run number and run status, scaler data, wire chamber high voltages and currents, and beamline-element information. The SDA process was launched from the DRC startup script. It began by opening a connection to the DRC, opening a branch to the serial CAMAC chain, and initializing all the CAMAC crate controllers. The process then waited for commands from DRC. A DRC Initialize command started an initialization process which included the following:

- reading configuration and parameter files;
- clearing and initializing CAMAC modules;
- resetting the optical receivers and transmitters;
- initializing internal variables;
- initializing DAMP (described below);
- initializing the SAD configuration.

Following initialization, no further action was taken until a DRC Start command was received. Following a Start, SDA wrote the current run number (as reported by DRC), run polarity, and run status (active) into a log file read by the on-line monitoring code. SDA then opened a file (spill log) used to store data collected during the beam spill and, for normalization reasons, a second data file (interspill log) used to store scaler data collected between spills. Execution then passed to the primary spill loop, executed once each spill. The spill loop began at the start of beam extraction and terminated approximately 72 seconds later. Data were collected at specified points along the loop based on the value of a 48-bit CAMAC scaler incrementing at a rate of 13 kHz. This clock scaler was reset by a hardware pulse called T5, which was generated immediately before the beginning of beam extraction. All SDA times were therefore referenced to T5. The clock scaler was read by SDA via CAMAC whenever a time value was desired.

The information collected and written to log files included:

- 155 48-bit CAMAC scalers;
- wire chamber voltages and currents;
- Hall probe data from sensors positioned within the spectrometer magnets;
- set points, currents, and read-back values for various beamline elements;
- SWIC information at five locations along the beamline;
- miscellaneous information such as date, time, and spill number.

The chamber voltages were measured using a LeCroy scanning ADC, while the chamber currents were measured by directing the HV-supply's current-monitor output into a custom voltage-to-frequency converter whose output was fed into a scaler. The scaler output represented the integrated chamber current, with each count corresponding to 50 μC. The Hall probes were read via CAMAC five times during the spill at six-second intervals. The beam positions and profiles were scanned and digitized nine times across the 40-second spill.

Data for beamline devices were collected and distributed by a program called Epicure [23] running on the Fermilab Beams Division's VAX/VMS cluster. The SAD application program ran alongside Epicure and provided communication between Epicure and the HyperCP SlowDA program. Following appropriate hand-shaking, Epicure data were obtained by the SDA through SAD function calls.

Forty-one seconds after T5 (i.e. after the end of the slow spill), the integrals of the individual chamber currents across the 40-second spill were read and written to the spill log. SDA then read all scalers (as defined in the configuration files) via serial CAMAC and wrote their values to the spill log, followed by the Epicure data reported through SAD, along with a time stamp. All scalers were then reset. After 71 seconds, the scalers were read again, these results being written to the interspill log file. Again, the scalers were zeroed.

In order to monitor the performance and operation of the experiment, key data were displayed at the end of each spill cycle using the DAMP [24] (DA Monitoring Program) software package. DAMP provided a versatile platform for displaying DAQ statistics, Hall probe values, and scaler data by way of online strip charts and static displays. Configuration files defined the display layout, the data being provided dynamically to DAMP by SDA.

The DRC Stop command inhibited future spill loops, closed the spill and interspill log files, terminated the SAD/Epicure connection, and stopped data reporting to DAMP. SDA terminated only if DRC terminated.

6 Online Monitoring

The online monitoring software package for HyperCP was written using the Fermilab DART programming package HOIST [25] to supply events for checking in real time, and a combination of the Fermilab HISTO (Histo-Scope) [26] and CERN HBOOK [27] packages for display and archiving. The HOIST package ran in a server-client mode with the servers running on the MVME167 data-acquisition computers, keeping a shared buffer on the DAQ SGI Indy full on a first-in/first-out basis. A program written mostly in FORTRAN running as a HOIST client on a second SGI Indy (or on offline DEC Alpha computers) could then request a record of events, essentially a "tape record" at a time, and then unpack and histogram any individual signal in the system. A unique aspect of the monitoring software was that except for the actual code for requesting the data record, the program was written entirely using our standard offline processing package, and thus was able to serve not only as a test of the hardware, but also as the most sensitive and demanding possible checker of our offline software algorithms.

Each event was unpacked using standard unpackers and passed to a histogramming routine which then used HISTO to display hardware profiles and occupancy plots for essentially every signal in the system. Because of HISTO's limited printing capabilities, we simultaneously filled HBOOK histograms which were then output either on demand or at the end of the run for archival purposes. A standardized PAW [28] macro was run on the HBOOK output to produce a set of printed histograms which could easily be compared to others on a run-by-run basis for each element in the system.

The software was initially developed during the 1997 run. Modifications were subsequently made to accommodate the hardware changes implemented for the 1999 run. Basically all configuration was handled by data files read in at execution time. Although this had a minor impact on the program startup time (actually less than the time necessary to establish the client-server connection and to initialize the HISTO package), it had the advantage that the hardware configuration could be changed almost in real time by changing an entry in a data file.

The Fermilab HISTO package was used for the online display because its user interface is extremely powerful and flexible. Users could easily choose which plots to examine closely, and had interactive control over both ordinate and abscissa, allowing them to expand or contract the display as desired.

Although HISTO was designed to enable users to change interactively almost any feature of the plots being displayed, we also utilized a feature whereby a "canned" set of histograms was displayed. During a typical run, for example, users had available a profile of each of the wire-chamber planes which enabled them to immediately identify either "hot" or "cold" wires – or more often, groups of wires which failed due to a bad cable or preamplifier, for example – and take appropriate action. Also displayed was a set of occupancy plots (number of occurrences of events as a function of the total number of hits in a chamber or hodoscope, or of a path or crate) which enabled the DAQ system to be load-balanced to reduce deadtime.

Offline studies after the 1997 run indicated that the track reconstruction efficiency was improved if "hot" wires were removed before processing (due to the elimination of some ghost tracks). Therefore a modification was made to the 1999 program to produce automatically a list of hot wires while the run was underway, and those data files were then immediately available for offline computing.

In addition to the monitoring software described above, users also had an online version of our offline event display program available. This program ran as a HOIST client, interactively displaying reconstructed tracks on a scale outline of the detector. Again, this program was running our actual offline reconstruction package, and thus served as a test of both the hardware and the software algorithms.

7 Summary

The performance of the DAQ met or exceeded all expectations. The observed throughput to tape was 27 MB/s, limited by the number and speed of the Exabyte tape drives. Data collection efficiency was improved with respect to Run I due to the robust nature of the new DAQ software, decreased startup time, and automated error recover procedures. The observed deadtime per event was 1.7 μs which resulted in a typical livetime of 80% at trigger rates near 100 kHz. Between June 1999 and January 2000, HyperCP collected more than 75 terabytes of data written to over 19,000 8mm tapes.

8 Acknowledgements

We thank A. Baumbaugh, T. Carter, S. Hansen, K. Knickerbocker, C. James, J. Meadows, D. Northacker, and M. Vittone for help and useful advice. We would also like to thank A. Chakravorty, C. Durandet, D. Rajaram, and S. White for their assistance. This work was funded in part by the U.S. Department of Energy and the National Science Council of the Republic of China.

References

[1] A. Alavi-Harati *et al.*, Phys. Rev. Lett. **83**, 22 (1999), V. Fanti *et al.*, Phys. Lett. B, **465**, 335 (1999), and B. Winstein and L. Wolfenstein, Rev. Mod. Phys. **65**, 1113 (1993).

[2] X. G. He *et al.*, Phys. Rev. D**61**, (2000).

[3] J. Antos *et al.*, Fermilab Proposal 871 (revised), Mar. 26, 1994, see also http://ppd.fnal.gov/experiments/e871/welcome.html

[4] K. B. Luk *et al.*, Phys. Rev. Lett. **85**, 4860 (2000) and P. D. Barnes *et al.*, Nucl. Phys. B (Proc. Suppl.) bf 56A, 46 (1997).

[5] T. D. Lee and C. N. Yang, Phys. Rev. **108**, 1645 (1957).

[6] See for example J. F. Donoghue, B. R. Holstein, and G. Valencia, Int. J. Mod. Phys. A, **2**, 319 (1987).

[7] C. G. White *et al.*, "Search for Direct CP Violation in Lambda and Xi Hyperon Decays", Nucl. Phys. Proc. Suppl. 71, (1999).

[8] Wong-Seng Choong, "A Search for Direct CP Violation in $\mathrm{K}^+ \rightarrow \pi^+\pi^+\pi^-$ Decays", Ph.D. Thesis, University of California at Berkeley (2000).

[9] "High Speed DAQ System for the HyperCP Experiment", Y. C. Chen *et al.*, accepted for publication in Nucl. Instrum. Meth. A (2000)

[10] A. E. Baumbaugh *et al.*, IEEE Trans. Nucl. Sci. **NS-33**, 903 (1985); K. L. Knickerbocker *et al.*, IEEE Trans. Nucl. Sci. **NS-34**, 245 (1986).

[11] R. Pordes *et al.*, "Fermilab's DART DA System," presented at Computing in High Energy Physics (CHEP 94), San Francisco, CA, 21–27 Apr. 1994, FERMILAB-CONF-94-103.

[12] Wind River Systems, 500 Wind River Way, Alameda, CA 94501.

[13] Original design ca. 1970 by F. W. Sippach, Columbia University Nevis Laboratories, P.O. Box 137, Irvington, NY 10533, updated for fast readout by F. W. Sippach and H. Cunitz, Columbia University Nevis Laboratories, in 1981. The latch card design was updated from ECL II to ECL 10,000 by J. A. Crittenden and D. M. Kaplan in 1984.

[14] http://p25ext.lanl.gov/e789/e789.html

[15] Manufactured originally by Ansley Co. (later Thomas & Betts/Ansley) during the 1970s and 1980s but no longer commercially available.

[16] J. A. Crittenden *et al.*, IEEE Trans. Nucl. Sci. **NS-31**, 1028 (1984).

[17] W. Sippach, G. Benenson, and B. Knapp, IEEE Trans. Nucl. Sci. **NS-27**, 578 (1980); Y. B. Hsiung *et al.*, Nucl. Instr. and Meth. **A245**, 338 (1986).

[18] Datel model ADC-943

[19] J. T. Meadows *et al.*, FERMILAB-CONF-95-110, presented at *9th Conference on Real-Time Computer Applications in Nuclear, Particle and Plasma Physics*, East Lansing, Michigan, 22–26 May 1995.

[20] Exabyte Corp., 1685 38th St., Boulder, CO 80301.

[21] D. Slimmer and J. Streets, "CAMAC Control in DART and Test Beam DA," Fermilab Computing Division note, DS0240 (1994).

[22] JORWAY Co., 27 Bond St. Westbury, NY 11950.

[23] "RD Controls Software Release 143.3, SAD Services for Remote Acquisition of Epicure Data," E. Dambik, Fermilab Programming Note PN0528 (1997).

[24] "DAMP User's Guide," C. Moore *et al.*, Fermilab Programming Note PN0522 (1997).

[25] "DFM_HOIST," G. Oleynik *et al.*, Fermilab Programming Note PN0482 (1998).

[26] "Histo-Scope Version 4.0," M. Edel *et al.*, Fermilab Programming Note SP0034 (1997).

[27] "HBOOK Reference Manual, Version 4.22," CERN Program Library Long Writeup Y250 (1994).

[28] "PAW – Physics Analysis Workstation," CERN Program Library Long Writeup (1990).

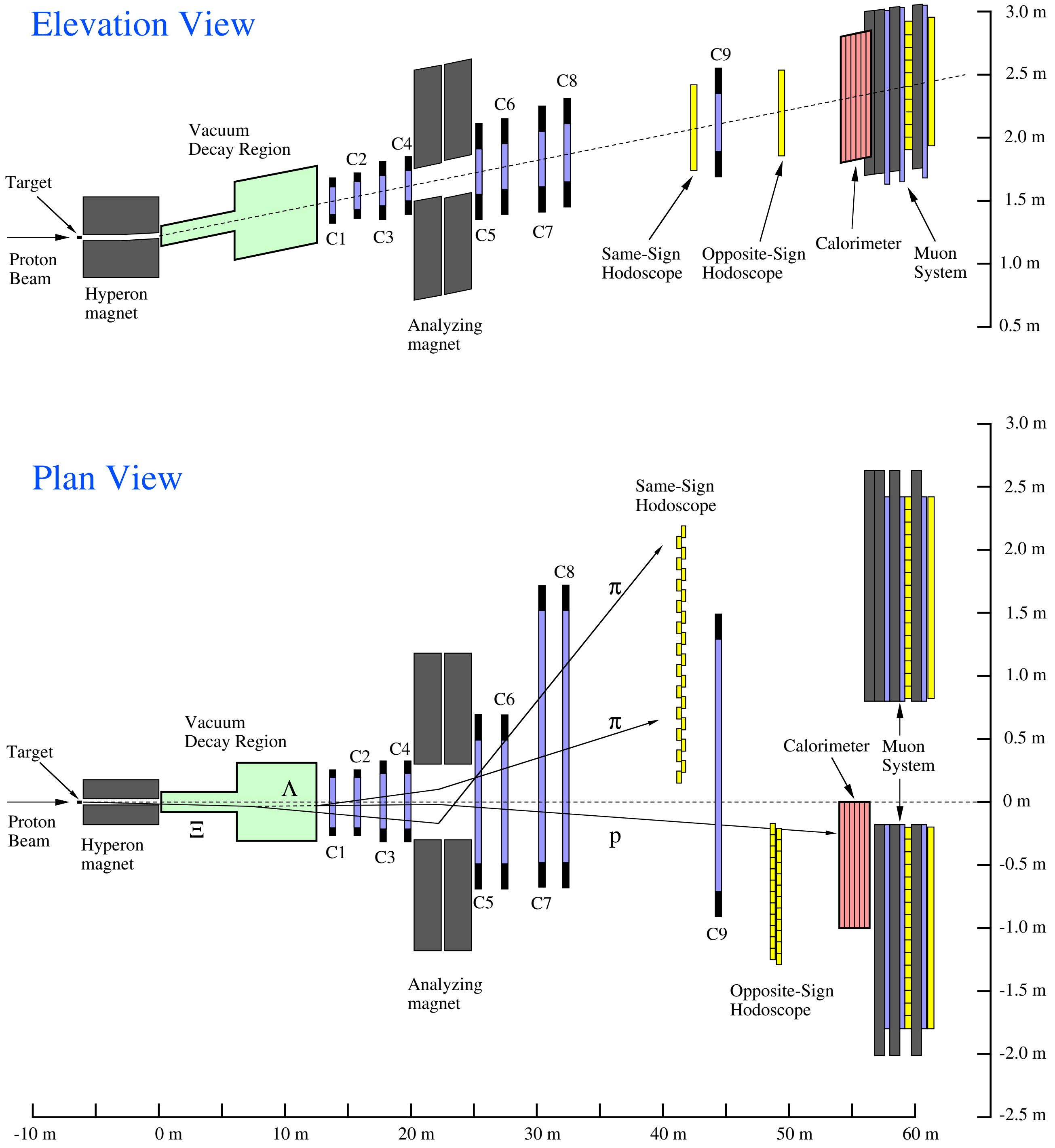

Fig. 1. The HyperCP spectrometer.

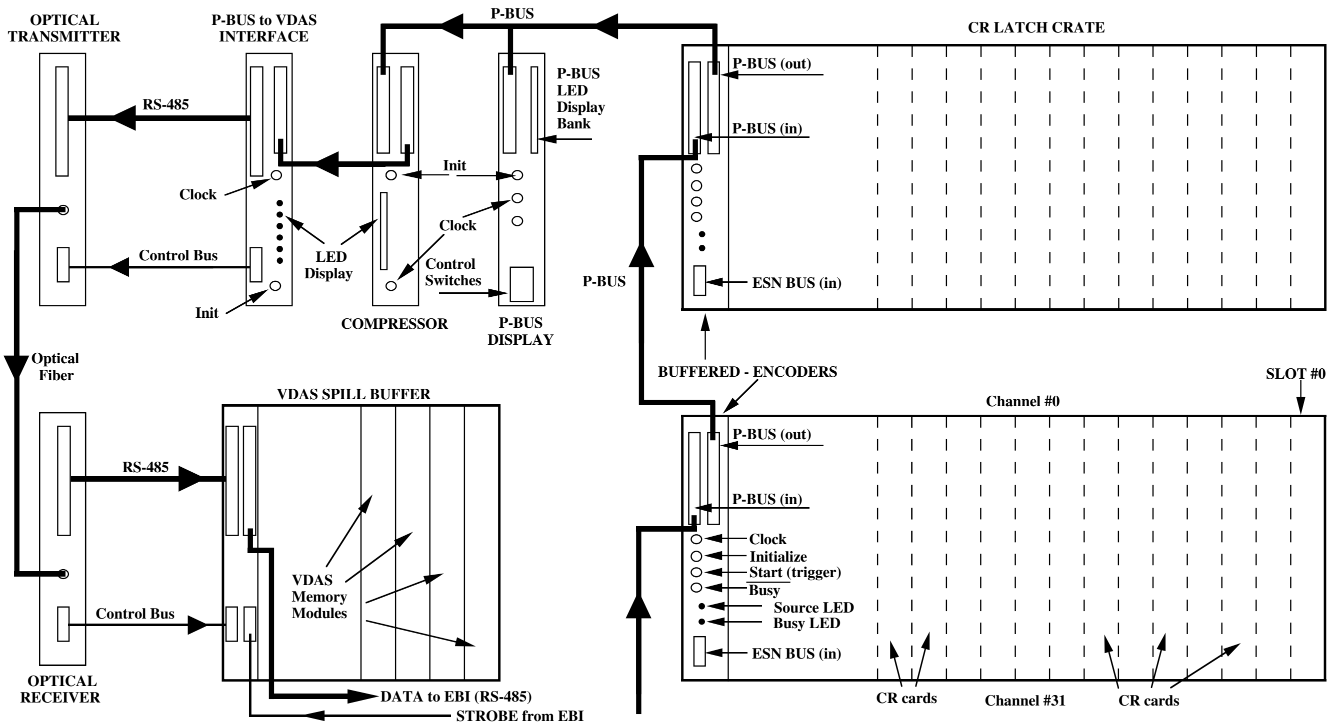

Fig. 2. Schematic of a data path.

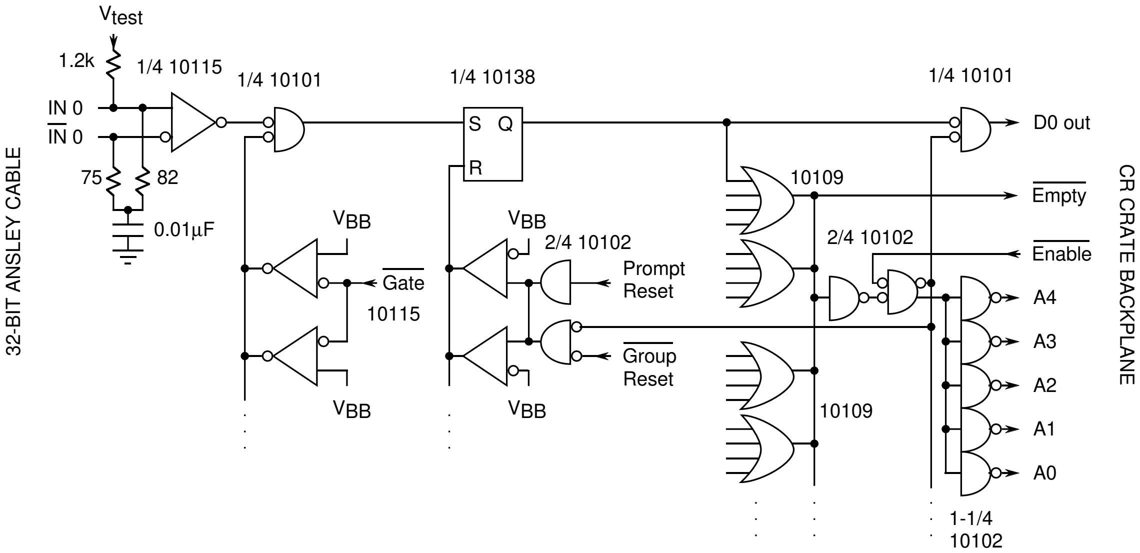

Fig. 3. Schematic diagram of a latch card (one channel of 32 shown).

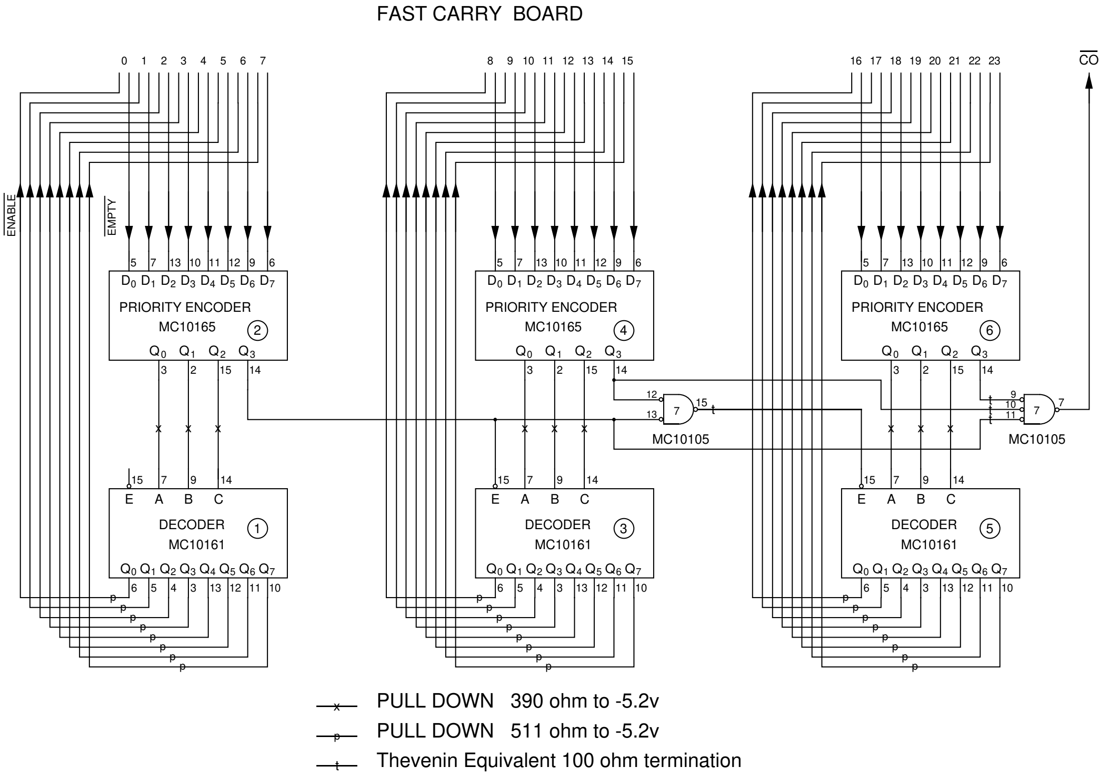

Fig. 4. Schematic of the Fast Carry Card.

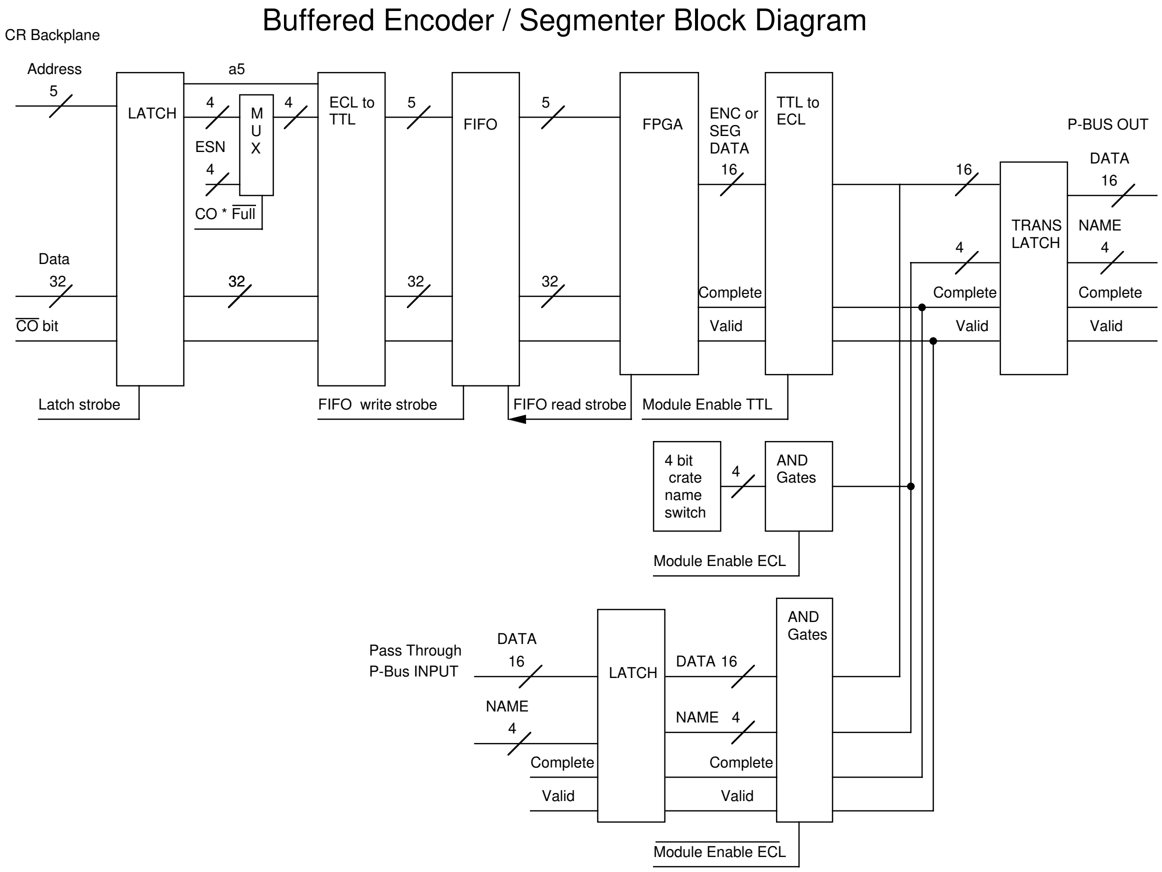

Fig. 5. Block diagram for the Buffered Encoder module.

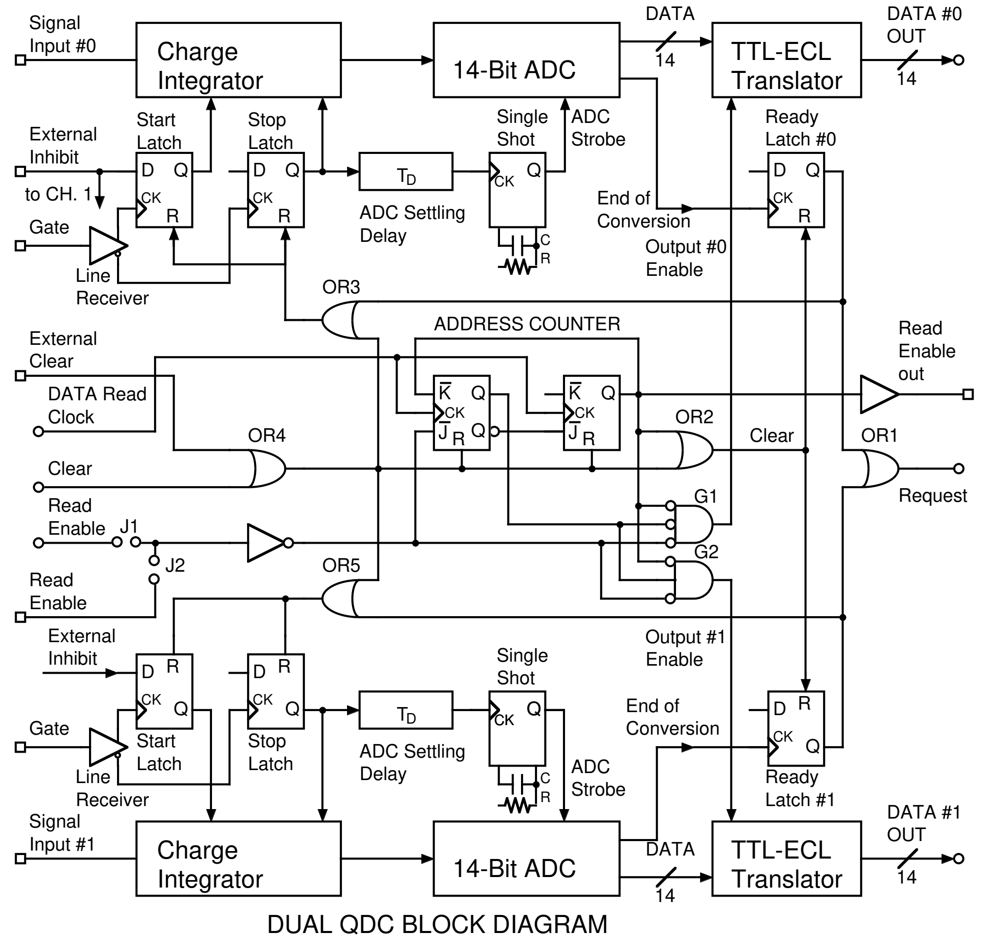

Fig. 6. QDC block diagram. refers to front panel lemo connections and ◦ refers to front panel ribbon cable connections

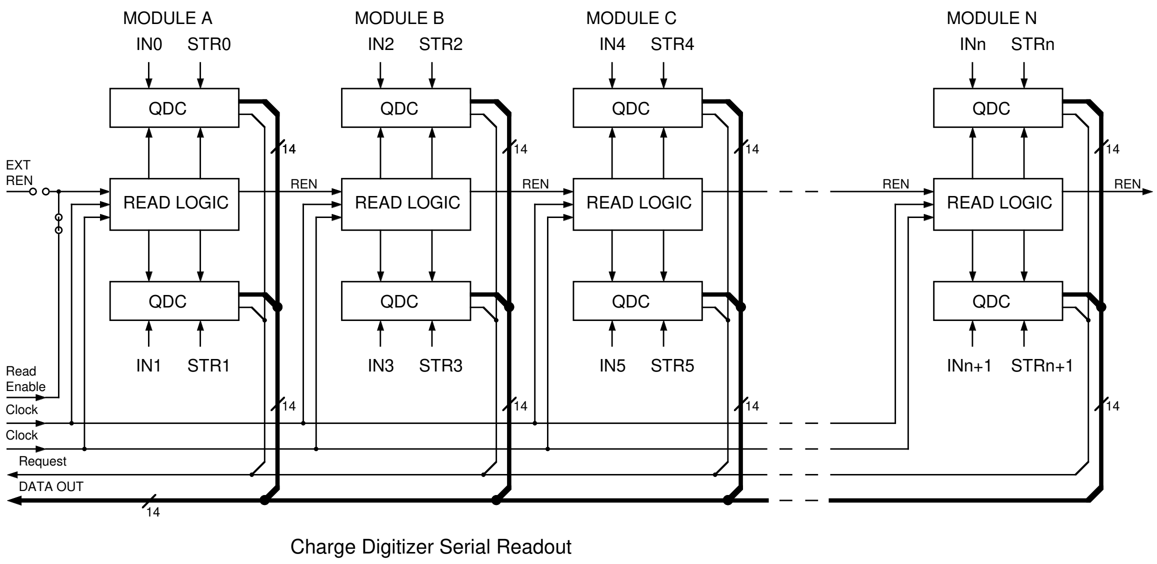

Fig. 7. Connections for serial readout of several QDC modules

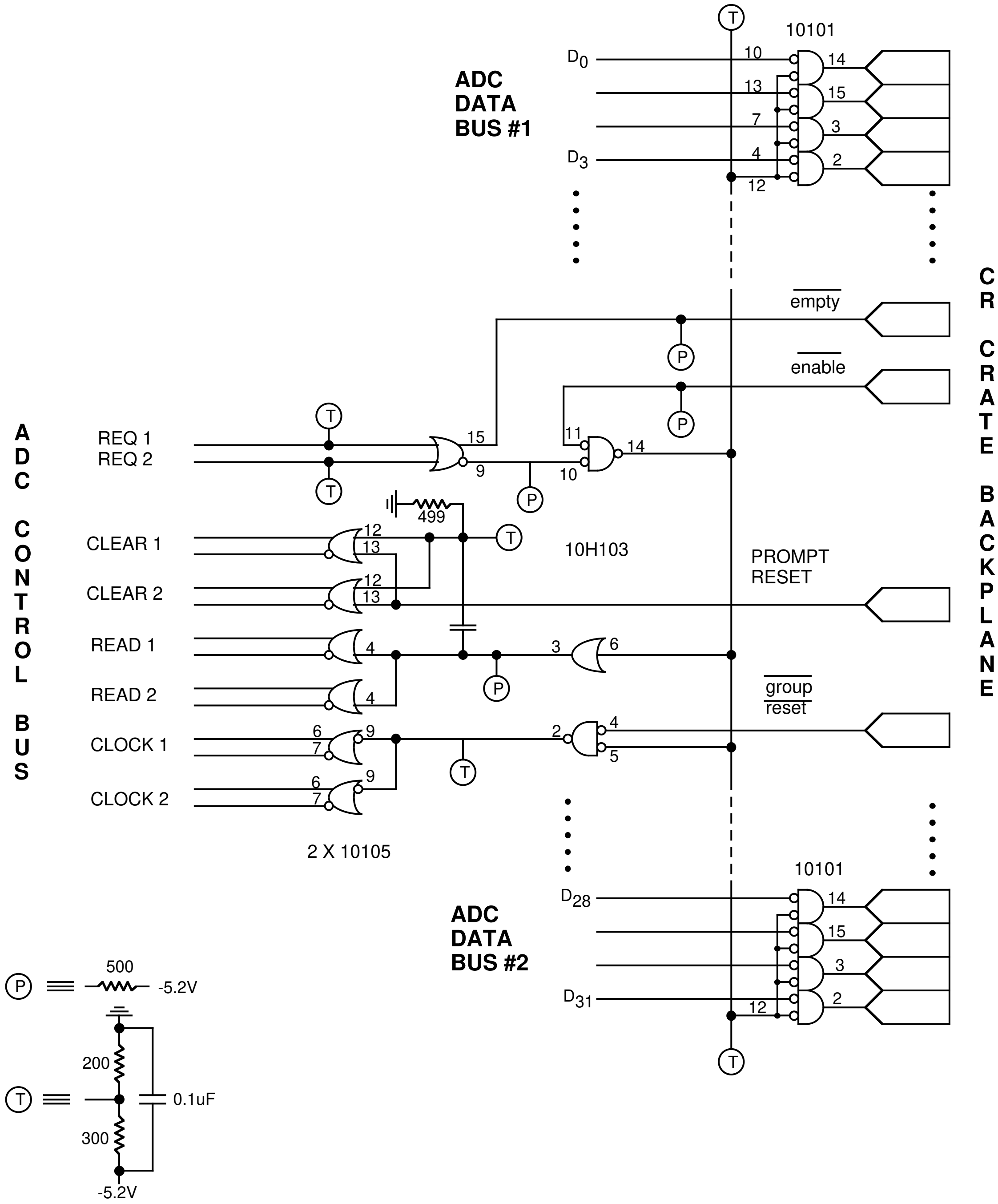

Fig. 8. Unencoded Source Module for interfacing the QDC control and data bus to a PWC latch crate backplane

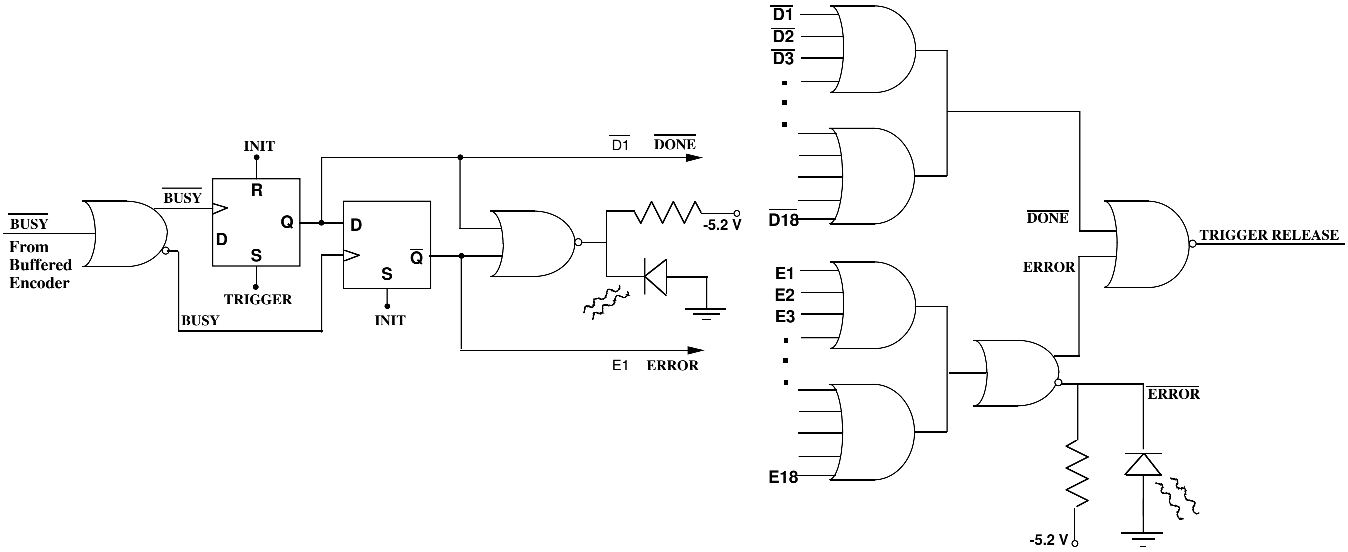

Fig. 9. Schematic diagram of Trigger Release Module (only one channel is shown).

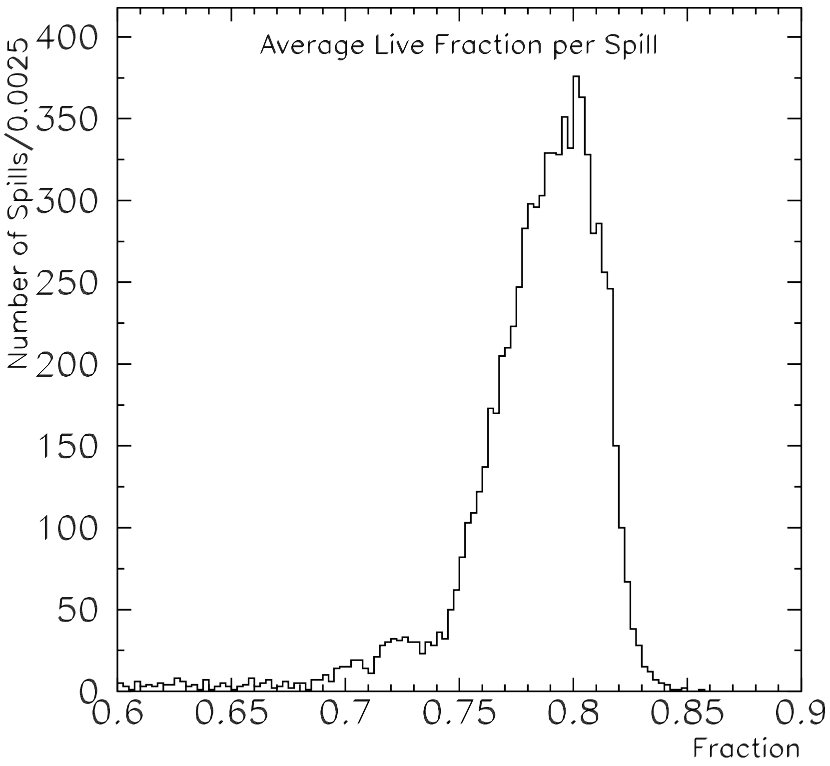

Fig. 10. Average live fraction for spills with trigger rates between 95 and 105 kHz.

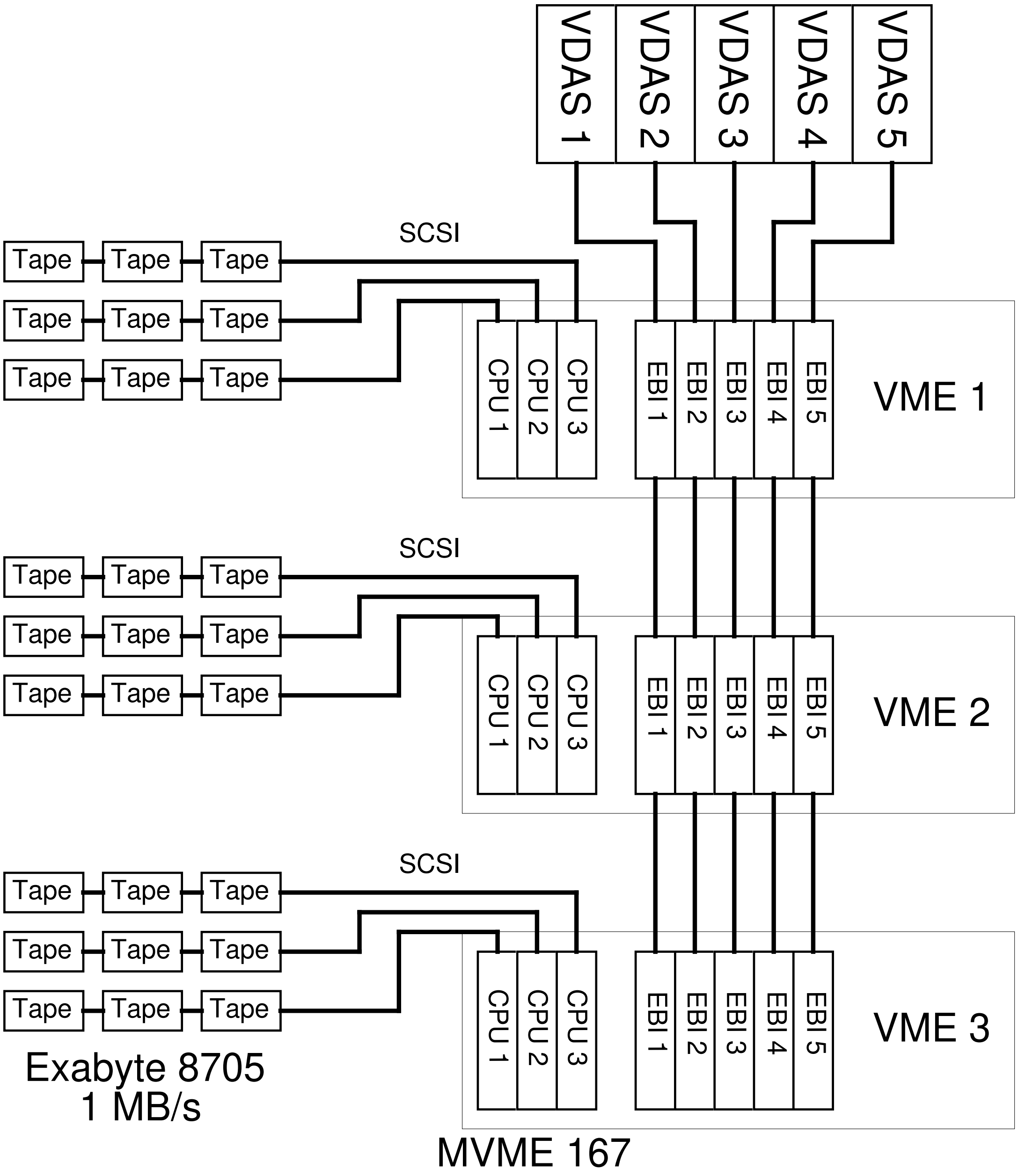

Fig. 11. VME hardware configuration.

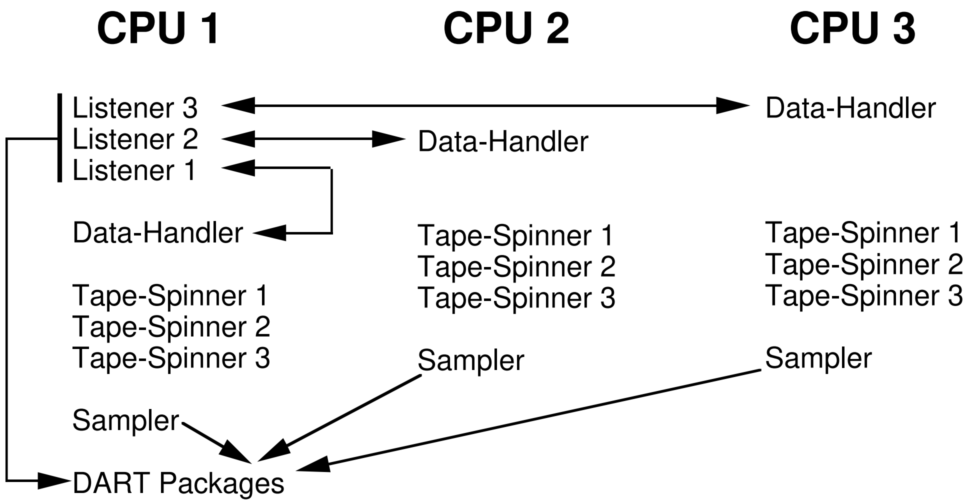

Fig. 12. Configuration of the processes running on each VME CPU.