Nuclear Instruments and Methods in Physics Research A 455 (2000) 424–432

NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A

www.elsevier.nl/locate/nima

A high-throughput data acquisition system for the HyperCP experiment

Y.C. Chen[a,*], K.C. Cheng[a], W.-S. Choong[c], E.C. Dukes[e], P. Gu[c], C. Ho[a], C. James[d], D.M. Kaplan[b], W.R. Luebke[b], K.B. Luk[c], K. Nelson[e], H.A. Rubin[b], J.P. Sheng[a], C.G. White[b], C.S. Yu[a]

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

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

[c] *Department of Physics, University of California and Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*

[d] *Fermilab, Batavia, IL 60510, USA*

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

Received 28 March 2000; accepted 29 April 2000

Abstract

The data acquisition system of the HyperCP experiment at Fermilab recorded about 50 TB of data on 12 000 tapes in 1997. The system recorded data at a sustained throughput of 12 MB/s typically and was capable of a maximum rate of 16 MB/s. The front-end electronics systems read 20 000 channels and achieved a typical readout dead time of about 3 μs per event, allowing operation at a trigger rate of 75 kHz with less than 30% dead time. © 2000 Elsevier Science B.V. All rights reserved.

PACS: 29.90. + r

Keywords: Acquisition; HyperCP; E871; Fermilab; Hyperon; CP violation

1. Introduction

The HyperCP experiment (Fermilab E871) [1,2] is searching for CP violation in hyperon decays by comparing the decay distributions of Ξ^- and $\bar{\Xi}^+$ decays. To achieve a sensitivity of 10^{-4}, which is better than the current limit [3] by two orders of magnitude, HyperCP needs to reconstruct 10^9 cascade decays in the $\Xi^- \rightarrow \Lambda\pi^- \rightarrow p\pi^-\pi^-$ mode and the corresponding $\bar{\Xi}^+$ mode. To reduce potential biases caused by on-line event selection, the HyperCP collaboration chose to collect events based on loose trigger constraints, and then use tight selection requirements off-line. Consequently, HyperCP required a data acquisition (DAQ) system capable of high data throughput to tape.

The Fermilab Tevatron delivered 800 GeV/c protons in a spill of 23 s with a 34 s inter-spill period. Due to the beam structure within the 23-s spill, the beam was useful to HyperCP for about 19 s. During the spill, triggered events were stored

* Correspondence address. Fermilab, MS 122 E 871, Batavia, IL 60510, USA. Fax: 1-630-840-3867.

E-mail address: chenyc@fnal.gov (Y.C. Chen).

PII: S0168-9002(00)00509-X

in a memory buffer, and the DAQ system wrote them to tape both during and between spills.

2. Overview of the HyperCP DAQ

The design of the HyperCP DAQ system was based on a similar system used in the E791 experiment at Fermilab [4]. The schematic of the HyperCP DAQ system is shown in Fig. 1. The front-end readout system, which resided in the electronic hall, consisted of two major components, the Nevis system[1] and the Front End Module system (FEM).

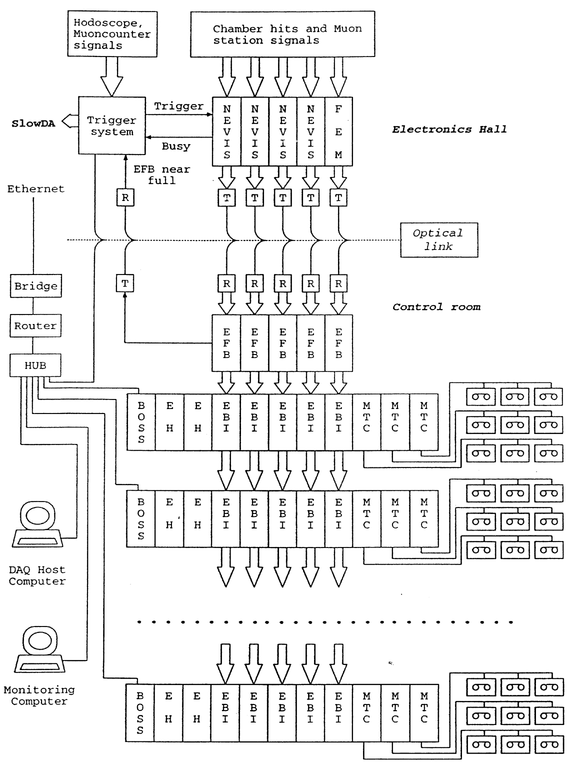

Fig. 1. Block diagram of the HyperCP data acquisition system.

We describe these systems below. Event fragments from the front-end readout system were accumulated and stored in five event FIFO (first in first out) buffers (EFB) [5,6] through five pairs of optical-link modules [7]. The EFB and optical-link modules were built by Fermilab. Four sets were used for the Nevis system and one set for the FEM system.

Since the distance between the electronics hall and the control room was 160 m, optical-fiber links were used to maintain excellent data integrity at high frequency. The optical transmitters converted 32-bit RS-485 data words to optical signals while the optical receivers converted the data back to RS-485. Each transmitter/receiver pair was capable of operation at a rate of 1 Gbit/s.

Each EFB contained 192 MB of memory; there were 960 MB in total. The EFBs stored events during the spill and allowed the DAQ system to write data to tape for the entire spill cycle.

Data from the EFBs were sent to five 6U VME crates working in parallel. Each VME crate housed three Motorola MVME167 computers, five Event Buffer Interfaces (EBI) and three magnetic tape controllers (MTC). The Ciprico RF 3573 SCSI interface[2] was used as the MTC. Each MTC handled three Exabyte 8505 tape drives,[3] which could run at a maximum rate of 500 KB/s in the uncompressed mode.

Each MVME167 had 8 MB of memory on board. The first computer in each crate was the Boss of the crate while the other two were Event Handlers (EH). The Boss was not only the master of the VME bus but also the controller and monitor of the DAQ crate. It scheduled the EH to take data in turn, sent data to tape, and monitored the throughput of the DAQ crate.

At any moment there could be only one EH in action. The function of the EH was to bring data from the EFBs into its own local buffers, to form event block by joining event fragments together and adding event header, and finally to complete the buffer by updating information in its own header block. Details are given in Section 4.2.2.

[1] 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 CR design was updated from ECL II to ECL 10,000 by J.A. Crittenden and D.M. Kaplan in 1984.

[2] Ciprico Inc., 1161 Homestead Ln., Chanhassen, MN 55317.

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

When all buffers of one EH were full, the Boss of the crate was notified. It then sent the address and size of these buffers to the MTCs which then transfered these buffers from the EH and wrote them to tape.

The real-time operating system, VxWorks 5.0.1,[4] was used on all MVME167 computers. The DAQ host computer, an SGI Indy, held the kernel of VxWorks and served all MVME167 computers. It was also the platform for developing the DAQ program and was the control center during data taking.

An additional MVME167 located in a crate near the front-end readout systems received commands from the DAQ host computer and served as a run control signal converter. It could initialize the front-end, stop or enable the trigger system via a signal generator, and also generated an eight-bit event synchronization number (ESN) for the FEM system for each triggered event.

To monitor data taking and to record the running condition a number of CAMAC crates were chained together in series to the DAQ host computer with a JORWAY branch driver,[5] which was a SCSI device and an interface to the CAMAC controller. Data from the scalers, hall probes, and beamline elements were read and recorded on disk by a separated program, the SlowDA, running on the DAQ host computer.

In the following sections we describe the structure of both the hardware and software of this system, and its performance.

3. Front-end readout systems

The task of the front-end readout systems was to latch digitized data from about 20 000 channels of multi-wire proportional chambers (MWPC), two trigger hodoscopes, a calorimeter, and a muon system. Upon receiving the trigger, it latched signals from the detectors, encoded them and sent them to the EFBs through the optical links. The read out time of a single triggered event varied depending upon which crate had the largest number of hits, and was typically about 3.5 μs. The HyperCP front-end readout system was composed of the Nevis and the FEM systems.

3.1. The Nevis system

Components of the Nevis system have been used in several experiments, such as E605 [8,9] and E789 [10] at Fermilab. As shown in Fig. 2, the Nevis system was composed of CAMAC card cages with custom-made backplanes. Each crate contained one Encoder (ENC) board, one FIFO-buffer module and up to twenty-two 32-channel Coincidence Register (CR) cards. There were a total of 35 Nevis crates instrumented and grouped into four data paths. Care was taken to balance the loads among the data paths.

The digitized signals from the multiwire proportional chambers, hodoscopes and trigger system were latched in CR cards whenever an event satisfied at least one of the trigger conditions. Channel latching resulted when a coincidence occurred between the digitized detector signal and a gate generated by the trigger system. Each 4-slot (128 channels) grouping within the latch crate required a separate gate input which would be timed-in according to the specifics of the detector and signal cables. The width of the gate was 20 ns for signals from chambers up-stream of the analyzing magnet, and was 40 ns for down-stream chambers. Gate

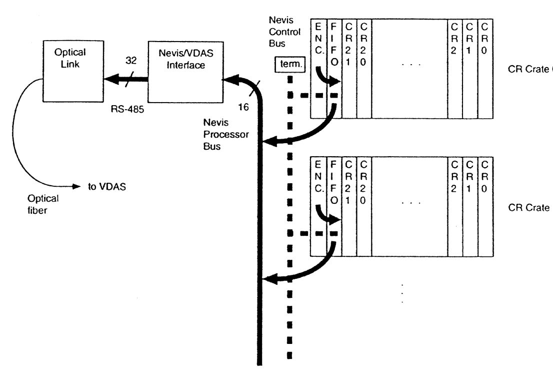

Fig. 2. Block diagram of a Nevis data path.

---

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

[5] JORWAY Co., 27 Bond Street Westbury, New York 11590.

cable timing was done with care to insure maximum latching efficiency.

The Encoder in each crate encoded the hit channels as 16-bit P-Bus protocol data words which were then stored in the buffer module. A four-bit ESN was incremented by each Encoder and encoded into the data stream upon receiving the read-out signal generated by the trigger system. The ESN was checked first by the hardware and subsequently by event-building software to verify that all event fragments were from the same triggered event.

Each data path was instrumented with a "Nevis/VDAS" interface module that read data serially from the FIFO-buffer modules. This module checked the ESN from each crate and held-up the readout in case of error. The interface module would stack and convert pairs of 16-bit P-Bus data words into 32-bit RS-485 data words required by the EFB. A 16-bit filler word was inserted at the end of an event fragment should the number of hits be odd. In addition, the interface module inserted a header word at the beginning of each event fragment identifying the ESN for the event, the data path, and type of data, and the number of bytes in the event fragment.

3.2. The FEM system

The fifth data path was based on the FEM system, shown in Fig. 3. This system digitized signals from the muon proportional tubes and the muon hodoscopes. The FEM crate was a 9U VME crate holding two interface modules, FEMIA and FEMIB, and up to 20 FEM boards. Each FEM board could latch 128 channels.

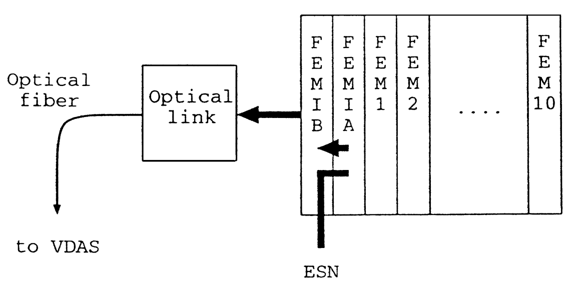

Fig. 3. Block diagram of FEM data path.

The FEMIA module read out data words from those FEM boards which had hits. The FEMIB had the capability of daisy-chaining several FEM crates together. Since there were about 1000 channels in the muon station, only 10 FEM boards in one crate sufficed for Hypercp, so the daisy-chaining feature was not used.

At the input end of each FEM board, ECL signals from discriminators were converted to TTL signals and latched if coincident with a gate signal. The cable length for gate and gate width were adjusted to maximize the efficiency of signal latching. The gate width was set at 500 ns. Upon receipt of the trigger signal, data were pushed into an internal FIFO buffer and the board was ready for taking next event. This made the system "deadtimeless". On board encoding logic sequentially processed the data in the FIFO buffer and stored the encoded data in the dual-port memory. When the FEMIA requested data, only FEM boards with encoded data responded. To allow readout of a given event, an ESN was required. Upon receipt of a trigger, this number was broadcast to all FEM boards. The FEM boards used the ESN as part of the address in storing the encoded data in the dual-port memory. When reading out the encoded data, the FEMIA read all data words with the same ESN by sending the ESN to FEM boards on the address lines. The FEMIB received data from the FEMIA, formatted them to meet the RS485 protocol, and sent them to the EFB through an optical-link module. An eight-bit ESN was required by the FEM subsystem and was generated by the run control signal converter but only the last four bits were compared to the four-bit ESN of the Nevis subsystem.

4. Software architecture

There were two essential components in the DAQ software, FastDA and SlowDA. On top of them were the Run Control and DAMP (Data Acquisition Monitoring Program) software products [11] provided by the DART (Data Acquisition in Real Time) group of the Fermilab Computing

Division [12]. A separate package, DFM_HOIST [13], was used to transfer a sample of events from the Boss to the monitoring computer.

4.1. DART Run Control

The DART Run Control program provided a user-friendly interface. The user could initialize, start, pause, resume or stop data taking by clicking the appropriate icons.

When all MVME167 computers finished booting, the user could start DART Run Control from the DAQ host computer. DAMP was brought up at the same time. Connections between the Run Control and all Boss were also established. The user could click on the INIT button to initialize the system, then click START to begin data taking. One could pause the run at any moment and resume later. When the tapes were near full, the DAQ system stopped automatically. One could also click on STOP to terminate data-taking if desired.

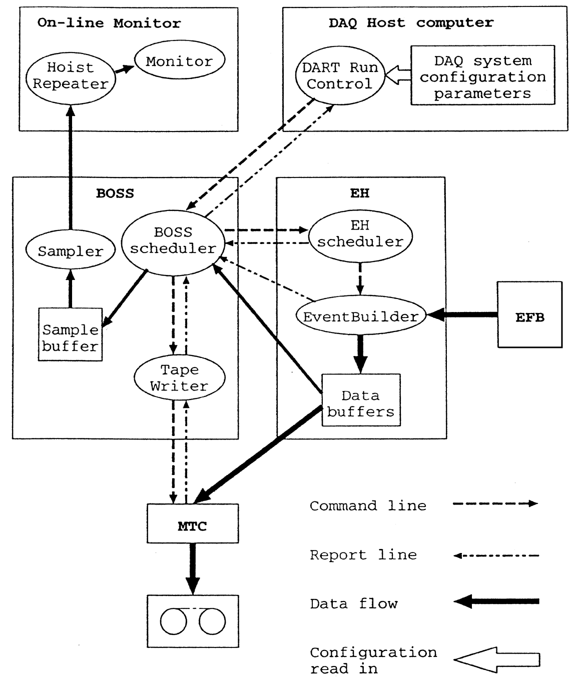

Fig. 4. Block diagram of FastDA.

4.2. FastDA program

4.2.1. Overview

The FastDA program took charge of logging data to tapes. A block diagram of FastDA is shown in Fig. 4. The FastDA was composed of two groups of programs. The programs in the first group ran on Boss were for controlling, monitoring and sending data to tape. The second group ran on EH were the routines for reading and formatting events. After the MVME167 computers were booted up the two groups of programs were automatically downloaded.

On the Boss the Boss Scheduler took the responsibility for receiving and executing commands from DART Run Control. It scheduled EHs to build events, and sent buffer addresses and sizes to the MTCs so that they could read event blocks from the EH and write them to tapes. It also monitored the running conditions, and sent a small fraction of events to the on-line monitor as well as reporting the status of tape drives, throughput rate of each event building crate, and warning or error messages to the DAQ monitor program running on the host computer.

The routines running on the EHs included EH Scheduler, Event Builder, Data Block Formatter. They received and executed commands from the Boss, built events, formed event blocks, and reported to the Boss about the status of event building or transferring data to tapes. If an error occurred, such as ESN inconsistency between event fragments, the EH notified the Boss and the information was passed to DART Run Control as well as the computers in the other event building crates. The system was then stopped smoothly. All events left inside the EFB were cleared when recovering from an error.

4.2.2. Event handling

In each EH, 50 static event buffers were allocated, each 64 kB in size. A block header was inserted at the beginning of each event buffer with some fields left empty such as number of events, and number of words used. The addresses of all event buffers were passed to the Boss when the system was initialized. (Since these were static buffers the addresses never changed.)

The Boss, having obtained the addresses of all data buffers in both EH, built up a table for EH status tracking. It also asked the MTCs to initialize the tape drives and checked for status. A special buffer was also created in the Boss to hold the pool of sample events for monitoring the performance of the spectrometer.

There were five EFBs, five data paths, and five DAQ crates working in parallel, each containing five EBIs (one for each data path, see Fig. 4). At any moment only one EBI on each data path could be granted access to its EFB. Upon start up of data taking, all EBIs in the first DAQ crate received the grant. Once data became available in the EFB, the EH of the first crate took the first event fragment of the first event from the first data path. It then passed the grant to the next EBI in the chain, and moved on to the second data path to read the second event fragment of the first event. At this point, the EH in the second crate read the first event fragment of the second event from the first data path. The EBI grant continued to be passed as each EH was finished with its current event fragment. For example, when the EH of the first crate was on path three, the EH of the second crate was on path two, the EH of the third crate was on path one, and so on. Finally, when the EH of the first crate was done with the fifth data path, it had obtained all the fragments of the first event. It then moved back to the first data path to collect the first event fragment of the sixth event. If the EBI of that data path was waiting for a grant the EH would be idle until the grant arrived. While waiting in the idle loop, it also checked with the Boss to see if there was any command to be processed, such as STOP.

The ESN was checked by the EH during event fragment reading. In case of an ESN mismatch, it sent an error message to the Boss, which propagated the message to DAMP and posted an error message on the screen to warn the user. Usually this meant something was wrong in the hardware and the system had to be reinitialized.

The EH kept filling the event buffer until the buffer was near full. It then inserted the number of events and the actual buffer size into the block header and went on to fill the next buffer. The header block recorded the number of events in this buffer as well as the identification number for the Boss and EH responsible for the handling of this buffer. A buffer was closed when it was more than 77% full to minimize the risk of losing events due to insufficient memory. This threshold was set based on the typical event size multiplied by a safety factor of 30 to take into account event-size fluctuations.

When all buffers were filled, it sent a message to the Boss Scheduler. The Boss Scheduler then activated the next EH to begin event building. In the mean time it also sent addresses and actual buffer sizes of all event buffers from the full EH to the MTCs, which copied all event buffers onto tapes. Attention was paid to balance the load among the tape drives and the amount of data on tapes. The EH stopped reading data when the EFB showed near empty and waited for more data to come in. Again, it checked with the Boss to see if there was any other command while waiting.

4.2.3. Data stream for monitoring

When the Boss Scheduler found that the active EH was full, it copied the contents of one event buffer into the sample buffer if it was empty. The buffer was marked as full once the copying was completed.

A customized DFM_Hoist program was created for the HyperCP experiment. It was composed of two parts, the sampler and the Hoist Repeater. At the request of the on-line Monitor program, the Hoist Repeater asked the Sampler to transfer events in the sample buffer. The Monitor program analyzed these events on-line. Once events in the sample buffer were transfered, the sample buffer was marked as empty so that the buffer would be refilled by the Boss Scheduler.

4.2.4. Operation of this DAQ system

To start data taking and start a new run, all buffers in the Boss and EH computers were marked as empty. All EFBs were cleaned up. The UNHOLD signal was then sent to the trigger system to start triggering. The EH computers started to read when all EFBs were no longer near empty. The Boss started to request MTCs to write data to tape when the first active EH was full.

In case of pausing the data taking the only action taken was to disable the trigger system by initiating

Table 1
E871 DAQ user commands for FastDA

User command	Meaning
INIT	Initialize the DAQ system
START	Start taking data
PAUSE	Pause taking data
RESUME	Resume taking data after PAUSE
STOP	Stop taking data

Table 2
E871 DAQ internal command signals for FastDA

Command signal	Meaning
INIT	Initialize the hardware and software
HOLD	Enable the trigger system
UNHOLD	Disable the trigger system

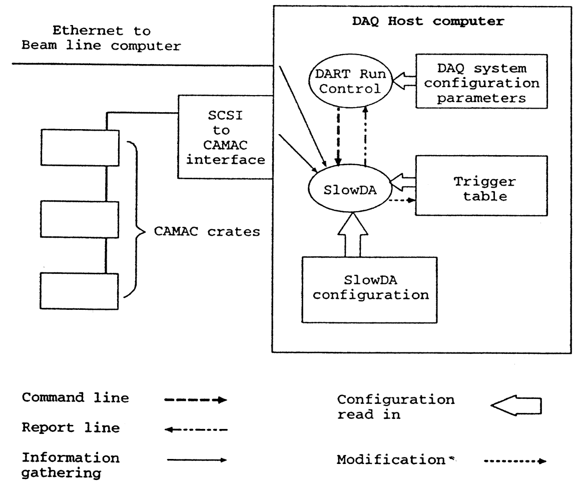

Fig. 5. Block diagram of SlowDA.

a HOLD command. When data taking was resumed, an UNHOLD command was sent to the trigger system to resume triggering.

To stop data taking and end a run, HOLD was first sent to the trigger system to stop triggering. A STOP command was then sent to all Boss and EH. However, the system continued to take data until the near empty signal showed up on the EFB of any data path. The system then wrote double End-of-File marks to all tapes and automatically ejected all tapes.

All FastDA commands available to user are listed in Table 1. The command signals are listed in Table 2.

4.3. SlowDA program

Running conditions were monitored by a special program called SlowDA. Information such as the number of triggers taken in each condition, the strength of magnetic fields and temperatures of magnets, as well as information from the beam position monitors were recorded spill by spill. A block diagram of SlowDA is shown in Fig. 5.

This system used standard CAMAC and GPIB instruments. It consisted of multiple CAMAC crates with the longest distance between two crates over 60 m which necessitated the use of the CAMAC Serial Highway. The control program of SlowDA ran on the DAQ host computer which was connected to the CAMAC crates through a JORWAY interface supported by the FNAL Computing Division [14].

In addition to CAMAC and GPIB instruments, the SlowDA obtained the beam line information from the beam line server known as Epicure through a client/server application called Simple Acquisition of Data (SAD) provided by the Fermilab Beam Division.

As with the FastDA package, the SlowDA program was integrated into the DART Run Control. When the DAQ system was initialized, SlowDA initialized all modules in the CAMAC crates, reset counters, and set pre-scale factors of triggers from the trigger table. The user could change these factors from the DART Run Control.

During data taking SlowDA read out information spill by spill, appended it to a log file and displayed it on the monitor screens through the DAMP program. The log file was closed when the DAQ system stopped.

The SlowDA had another responsibility besides logging the run condition. As mentioned in Section 2, there was a MVME167 located by the readout system to receive commands from the DAQ host computer. The procedure of passing commands was completed by the SlowDA through an Ethernet connection to this MVME167.

4.4. On-line monitoring

For users to understand how the system was performing, the percentage of each tape written, and the throughput rate of every VME crate were monitored. The Boss of each VME crate kept track of the progress of each EH. From the amount of data built per unit time, it calculated the throughput rate of each crate. The Boss also monitored the amount of data sent to each tape drive. The throughput of each crate and the amount of data written to tape by each tape drive were sent to the DAQ host computer and were displayed on screen by the DAMP program.

To monitor the performance of the spectrometer, sample events were stored in every Boss. Upon request, the DFM_HOIST package transfered data to the Monitor program. The Monitor program analyzed these events and displayed relevant information on screen, such as distributions of hits in MWPCs, muon stations, and hodoscopes. The event display was also available for dislaying the raw data and reconstructed information on an event-by-event basis.

5. System performance

This DAQ system was used by the HyperCP experiment in its 1997 run. The distribution of the event size is shown in Fig. 6. On average, the event size was about 600 bytes. In Fig. 7 the live time is shown with respect to the number of accepted triggers per spill. Typically, the experiment took data at a trigger rate of about 1.2 million per spill or 63 kHz and the average live time was around 75%.

The maximum number of triggers that the system could handle was around 1.6 million per spill, corresponding to a maximum trigger rate of 84 kHz. At the maximum trigger rate, the maximum sustained throughput of the DAQ system was 16 MB/s, with a live time of 65% or so. However, in certain situations, such as poor spill quality or a tape drive failure, the system could reach only 1.5 million triggers per spill and the corresponding live time was only 60%.

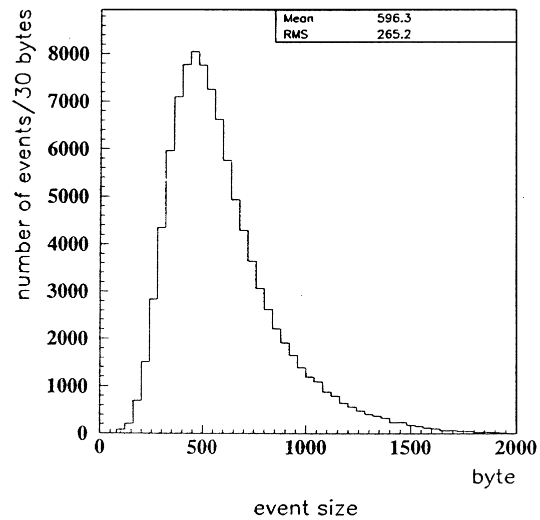

Fig. 6. Distribution of event size.

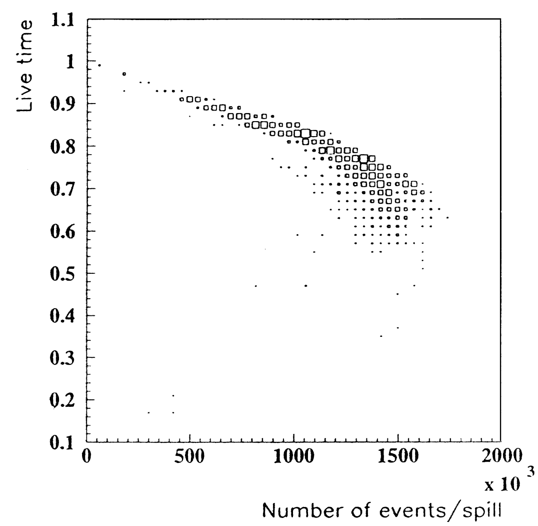

Fig. 7. Live time of the E871 DAQ system in 1997 run.

Between April and September of 1997, this DAQ system provided HyperCP the capability of collecting 50 TB of data on 12000 8 mm tapes. The projected sensitivity in searching for CP violation in hyperon decays is about 2×10^{-4} with this data sample.

Acknowledgements

We thank A. Baumbaugh, T. Carter, S. Hansen, K. Knickerbocker, and J. Meadows for help and useful advice, and the DART group of the Fermilab Computing Division for their support. This work was funded by the US Department of Energy and the National Science Council of the Republic of China.

References

[1] J. Antos et al., Fermilab Proposal 871 (revised), March 26, 1994.
[2] C. White et al., Nucl. Phys. B (Proc. Suppl.) 71 (1999) 451.
[3] P.D. Barnes et al., Phys. Rev. C 54 (1996) 1877.
[4] S. Amato et al., Nucl. Instr. and. Meth. A 324 (1993) 535.
[5] A.E. Baumbaugh et al., IEEE Trans. Nucl. Sci. NS-33 (1985) 903.
[6] K.L. Knickerbocker et al., IEEE Trans. Nucl. Sci. NS-34 (1986) 245.
[7] J.T. Meadows et al., FERMILAB-CONF-95-110, presented at Nineth Conference on Real-Time Computer Applications in Nuclear, Particle and Plasma Physics, East Lansing, Michigan, 22–26 May 1995.
[8] J.A. Crittenden et al., IEEE Trans. Nucl. Sci. NS-31 (1984) 1028.
[9] J.A. Crittenden et al., Phys. Rev. D 34 (1986) 2584.
[10] D.M. Jansen et al., Phys. Rev. Lett. 74 (1995) 3118.
[11] C. Moore et al., "DART DA Monitoring", Fermilab Computing Division note, DS243, 5 August 1998.
[12] R. Pordes et al., Fermilab's DART DA System, Presented at Computing in High Energy Physics (CHEP 94), San Francisco, CA, 21–27 April 1994, FERMILAB-CONF-94-103.
[13] Gene Oleynik et al., "DFM_HOIST" Fermilab Computing Division note, P5029 (used to be referenced as PN-0482), 4 August 1998.
[14] D. Slimmer, J. Streets, "CAMAC Control in DART and Test Beam DA", Fermilab Computing Division note, DS240, 9 September 1994.