Fermi National Accelerator Laboratory

FERMILAB-Conf-97/259

High Speed Data Acquisition

Peter S. Cooper

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

July 1997

Presented at the *VII ICFA School on Instrumentaion in Elementary Particle Physics,* Leon, Guanajuato, Mexico, July 7-19, 1997

Operated by Universities Research Association Inc. under Contract No. DE-AC02-76CH03000 with the United States Department of Energy

High Speed Data Acquisition

Peter S. Cooper

Fermi National Accelerator Laboratory
MS 122 P.O. Box 500 Batavia, Il 60510 USA

Abstract. A general introduction to high Speed data acquisition system techniques in modern particle physics experiments is given. Examples are drawn from the SELEX(E781) high statistics charmed baryon production and decay experiment now taking data at Fermilab.

INTRODUCTION

Data acquisition systems [DAQ] actually "do" modern particle physics experiments. We program and "train" them to select the events we want to keep. They respond on nanosecond to millisecond time scales to select the interesting data, put those data together into an event and ultimately save those events on a data tape. They also provide the command and control functions to allow the physicists to monitor what is happening on the detector and make changes.

DAQ SYSTEM FUNCTIONS

DAQ functions naturally split into three categories based upon the required response time. High speed functions are associated with the rate of events in the apparatus. Slow control functions are in real-time but with times of order seconds. On-line monitoring functions are near real-time. These operate from seconds to minutes after data is available. The sections below enumerate some of the functions required of each system.

High Speed DAQ Functions

* Select interesting interactions to keep for further study
  - With fast electronics [Trigger]
  - With fancy software [Filter]
  - With hybrids of the two

* Digitize analog detector signals into digital bytes of data
  - Event data is digitized if it is not born that way
  - Data is sparsifed - suppress the channels with zero

* Collect data for each of the detector systems
  Sub-events from each detector system (or systems) are packaged and usually "pipelined" to the higher levels of the DAQ.
  - Usually in high speed systems this level of data handling is happening in parallel in all sub-systems at the same time.

* Build events
  - Collect all the sub-events from the *same* trigger together
  - Add the necessary structures to correctly format events

* Analyze events in software
  - Some experiments run their off-line analysis code on some or all of the events.
  - If this step rejects significant numbers of events this is a software filter (Level 3 in some experiment's jargon).
  - Summaries (histograms, etc.) are built to monitor filter properties.

* Permanently record the selected events
  - Split-up up events by type (trigger?) into different files
  - usually on magnetic tape of some type
  - Newest trend is toward direct network transfer of data files to the Computer Center

A schematic of DAQ components is shown in Figure 1 below. Analog detector signals are digitized when a trigger arrives. The digitized data is "piped" to an event builder where all fragments of an event from several front end/digitizer systems come together. Built events are sent to an on-line computer where the may be filtered with software to further select interesting events. Selected events are written to data tape and/or over the network to the computer center.

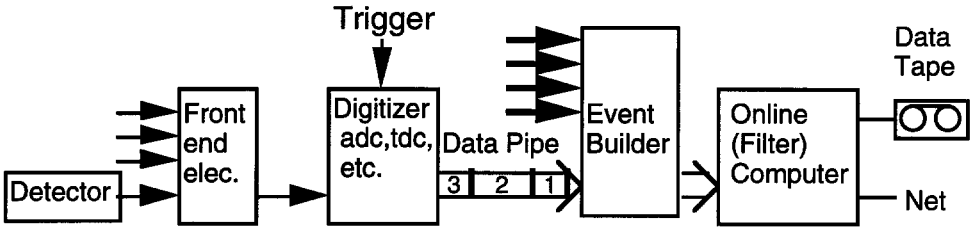


FIGURE 1. Schematic DAQ components.

Slow Control System Functions

* Control the experiment
  - Start and stop runs
  - Reload triggers, electronics, etc.

* Cold Start - reboot all or part of the DAQ itself

Monitoring Functions

* Monitor the experiment status
  - Readout non-event data (scalers, rates, etc.)
  - Log these data

* Complete sufficient analysis to display the status of the running experiment
  - Trigger rates
  - Detector status - e.g. MWPC wire maps, number of hits/plane, etc.
  - Events - Single event display (usually from the off-line)
  - Filter - pass rates, stability

TECHNOLOGIES

Data acquisition systems are built using a set of standard technologies for electronics, data transfer, computers and programming. In all of these technologies there are standards which have been developed so that all components don't have to be engineered from first principles. Very often experiments must design and engineer their own components because the functionality they require isn't available in a standard module or program. Most the manpower expended in building DAQ system is invested in this area. It is almost always advisable to use standards to the maximum degree possible, even when engineering specialized systems. The amount of effort required to replicate, in a *working* system, what already exists in an existing standard is always underestimated.

Standard Electronics and Readout Busses

Most experiments use standard electronics in different families. Most of these are busses with interfaces to computers and or other busses for readout and control. The CAMAC standard is the archetype. The big issue with these standards, and their abuses, is software support. Usual more effort is usually spent on programming a new module or system and all the things around it which aren't there, or don't work, than engineering it.

The list below gives most of the common electronics standards using in particle physics experiments. These are defined as real engineering standards so that different designers can engineer components which will work together under the definitions of a standard. (1)

- NIM — Nuclear Instrumentation modules (late 1960's)
- CAMAC — Computer Automated Measurement and Control (1970)
- Fastbus — Better, faster (harder) CAMAC (~1980)
- VME — Computer industry standard (particle physics ~1985)
- Home-brew — Some people who build their own electronics don't believe in standards (e.g.: Transport bus(es) from Nevis Labs) Some systems use a standard bus, like CAMAC, in non-standard ways (PCOS and FERA from Lecroy, among many others)

Data Pipes - Systems for Long Distance (>10m) High Speed Transfers

Moving large volumes of data around is challenging. No bus *ever* achieves its maximum theoretical bandwidth. Most can't achieve 50% of the maximum in

realistic applications. Some, like ethernet, can provide up to 10% to several simultaneous users but no one user can get more than 10%.

The big issues in moving data through pipes are blocking and software overhead. The maximum bandwidth is determined for one, infinitely large, block of data. The time to start and stop the transfer of a block usual dominates unless block sizes are very large and great care is taken to optimize a system.

To achieve the maximum average data transfer rates through a data pipe it is highly advisable to send data in only one direction with the largest possible block sizes to minimize the overhead in starting and finishing blocks. There is a well known example of an experiment who used the same data pipe network to send their data from detector to DAQ and their control messages from DAQ to detector. The control messages had difficulty "swimming upstream" in the large data flow and would sometime arrive too late or not arrive at all. Performance in that system suffered greatly.

- RS232	serial terminal lines (more of a straw than a pipe) user to download types of controllers (e.g. HV systems)
- CAMAC	branch highway (~100 Kb/sec) Allows multiple crate CAMAC systems
- FASTBUS	segments (~1Mb/sec) - like a CAMAC branching highway
- Ethernet	computing industry standards
- SCSI	- small Computer System Interface - disk and tape systems
- FDDI	
- HPPI	
- ATM	
- RS 485	parallel ECL lines at 10MHz clock rate
- optical fiber	up to 1 Gbit/sec on the fiber but driving electronics is never this fast.

Real-time Computers

Real-time computing is fundamentally different than the general purpose batch or timeshared computing. It is much harder to write and debug real time programs. The basic issue is response time - how long before an interrupt can get to it's service routine. The problem come when two things try to happen at once. (The second interrupt comes while the first is still being serviced.) The difference in real time systems is in the operating system much more than the computer hardware itself. All "big" computers treat their disk subsystems as real-time devices. They just don't treat their user's jobs with that kind of priority.

There are many different types of real-time computers and operating systems. They are typically small, complicated and for experts only. Real-time programming is much harder than usual. If you hear about a real-time problem and you are not a DAQ expert the experiment is in trouble.

These systems are often used as embedded controllers in DAQs grafting some data stream into another system. Examples are Fastbus masters and the processors at the beginning and ends of data pipes like the Selex fiber optic data paths.

General Purpose Computers

The systems we are all familiar with from our desktops and computer centers are a backbone element of high performance DAQ systems. In many ways jobs which formerly were done only in the off-line analysis phase of an experiment have migrated on-line. These include filtering events before writing them to tape and certain monitoring, calibration and alignments. These are really just doing the off-line analysis and first levels of event selection cuts before writing the data to tape. Programs like the DAQ user interface are often written using the high level graphics and programming tools available on this type of system.

These systems are typically UNIX workstations or multi-processors. Programming is done in high level languages, FORTRAN, C and recently C++. There is a strong overlap with parts of the off-line analysis programs. The unpacking routines in the off-line have much more to do with details of the DAQ than the analysis. Conversely, the single event display is usually integrated into the on-line by stealing it wholesale from the off-line codes. A filter code is often just a version of the off-line analysis rejecting events before they have been written to tape.

Programming

The majority of effort put into building a DAQ is in programming. The techniques are advanced and complicated. (Lots of queuing theory.) No system can be really tested and debugged in isolation; problem isolation is an enormous challenge. A typical problem is a filter job crashing in one event in a million because some hardware module in the experiment is dropping or corrupting bits. You cannot isolate this kind of problem without knowing a great deal about all the hardware and software between the particle detectors and your piece of analysis code.

DAQ *SYSTEMS*

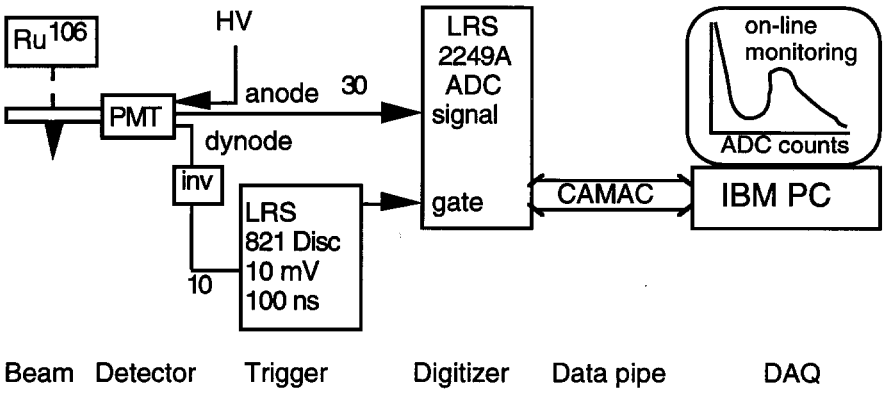
The critical issue that is often missed in designing and building a new DAQ is that it is the system which has to be optimized, not a small part of the system. All new DAQ systems are, by definition, beyond the state of the art. No one designs a new one when an old system can be adapted or upgraded to do the job. DAQ's are large projects involving many people; programmers, engineers, technicians and physicists. Getting everyone to understand the big picture is hard. Getting them to all agree is impossible.

As an example, the Selex DAQ is built out of hardware and software components produced by the DART collaboration at Fermilab (2). This is a group of about 25 professional programmers, electronics engineers, and technicians plus one or more physicists from each of eight experiments. This team has produced part or all of the DAQ systems running on nearly all the Fermilab fixed target experiments now taking data. Several years were required to bring this project to its present point with many different, stable, DAQ's taking data everyday.

DEAD TIME

The most important parameter controlling the design and performance of high speed DAQ systems is dead time. Dead time occurs whenever a given step in the processing of the data in a DAQ takes a finite amount of time. If a particular step takes 100 nsec and the second event comes 50 nsec after the first there is a problem. In the simplest case the second event is just lost - the system is dead for that event. This is always enforced in electronics at the trigger level. When the DAQ cannot accept another event it asserts a logic level usual called "busy" which prevents the trigger electronics from accepting another event. When the busy is reset DAQ can continue.

One of the simplest DAQ systems possible is shown in Figure 1 below. The beam is a radioactive source and the detector a single scintillation counter consisting of a plastic scintillator and a photo multiplier tube (PMT). The trigger electronics consists of a NIM discriminator which fires whenever the inverted dynode pulse from the PMT exceeds 10 mV. With the PMT high voltage properly adjusted the trigger rate will be dominated by the rate at which beta particles from the source pass through the scintillator. There will also be a small but finite counting rate when the source is removed due to noise in the PMT.



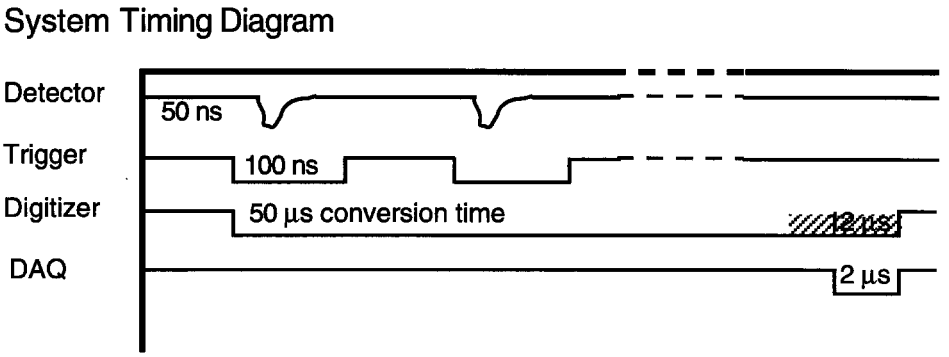


FIGURE 2. A simple DAQ system

The goal of this system is to measure the performance of the detector, specifically the pulse height spectrum of the scintillator. This is a real example in

the sense that the all the parts are real devices which your experiment already has. This is an example of a "test stand"; a small and simple DAQ system used to checkout parts of an experiment.

The ADC has a finite conversion time of 50 μsec and will digitize the charge in a pulse contained with the gate. The width of the trigger pulse is set to 100 nsec to contain all the charge from a large PMT pulse. The relative timing of the gate and the anode signal are adjusted with cable delays to arrive the ADC with the trigger from a given pulse reaching the gate input just before the anode signal arrives at the signal input.

Once the ADC receives a gate it digitizes the charge within the gate to a 10 bit number with a conversion gain of 1/4 pC/count. The digitization takes 50 μsec. This type of ADC ignores subsequent gates until it is cleared with a CAMAC command (like F2 - read and clear).

The ADC interrupts the DAQ computer when the conversion is complete and the interrupt service routine reads and clears the ADC (dropping the internal ADC busy). This interrupt response time is 10 μsec plus 2 μsec for the actual CAMAC read and clear operation.

The background routine in the DAQ computer adds the new event to the pulse height histogram and updates the display of that histogram on the screen. This is on-line monitoring happening in near real time.

This DAQ system is already complicated enough to exhibit dead time. A second trigger with 62 μsec of the first is ignored. The fraction of triggers not lost to dead time is the lifetime ratio given by:

$$R / R_T = 1 / [1 + R_T T_d]$$

R	DAQ rate
R_T	Trigger rate
T_d	Dead time [62 μsec in this case]

The maximum DAQ rate is $R = 1 / T_d$ independent of the trigger rate. At this trigger rate; $R_T = 1 / T_d = 16$ KHz and the livetime ratio is 50%. Half of the events are lost to dead time at this trigger rate.

Minimizing Dead Time

Consider what happens if we used two ADC's in above example instead of one with odd events going to one ADC and even events to the other. Digitization and DAQ could proceed in parallel and the dead time would be reduced. However this system would have to be more complicated in order to route odd and even events correctly and decide when both ADCs were busy. This is an example of the first technique for reducing dead time - parallelism.

A second approach is pipeline processing of the data flow. A complicated operation can be broken into several simple steps each operating on subsequent events. The dead time goes down because the time to perform each step is less

than for the whole operation. Using first in first out (FIFO) buffer memories between step allows for variable times per step. The buffers will fill up while a long event is processed and empty during shorter events. A general schematic of a pipeline is shown in Figure 3 below.

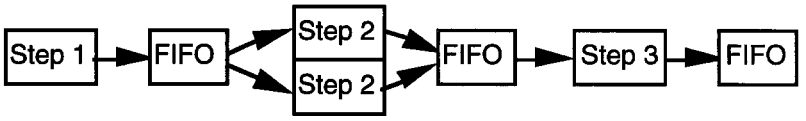


FIGURE 3. A pipeline processor.

These two techniques can be combined in ways like that shown above.

Dead time can never be completely eliminated. In the example of Figure 2 even if all dead time associated with the ADC were eliminated the trigger circuit itself has a 100 ns dead time due to the output pulse width. Since synchrotron beams are modulated with a maximum frequency by the accelerating RF a DAQ with a dead time smaller that the RF period is effectively dead timeless - it clears before the next beam particle can arrive.

The CDF and D0 collider experiments at Fermilab had DAQ systems designed for the 3.5 μsec time between bunch crossings in the Tevatron. The Tevatron collider upgrades now in progess reduce this time between crossings to ~400 nsec. All the front-end electronics in both of these experiments which exploited the 300 KHz maximum crossing rate are now being re-engineered. They are not dead timeless anymore.

A REAL DAQ SYSTEM

The schematic of my presently running Fermilab experiment is shown in Figure 4. Selex(E781) is a fixed target experiment designed to study the production and decay of charmed baryons (3). It is presently taking data in a 1MHz 600 GeV/c secondary beam in the proton center beam line at Fermilab.

The physical layout of the experiment presents a challenge to the design of the DAQ system. All of the trigger electronics and most of the DAQ systems are required to be placed close to the detectors in the radiation area. The back end of the DAQ and the Physicists are located in an upstairs counting area 100m away. Due to the requirements of radiation safety it is not possible to work on the electronics with the beam on. All changes must be remotely programmable or require a beam off access.

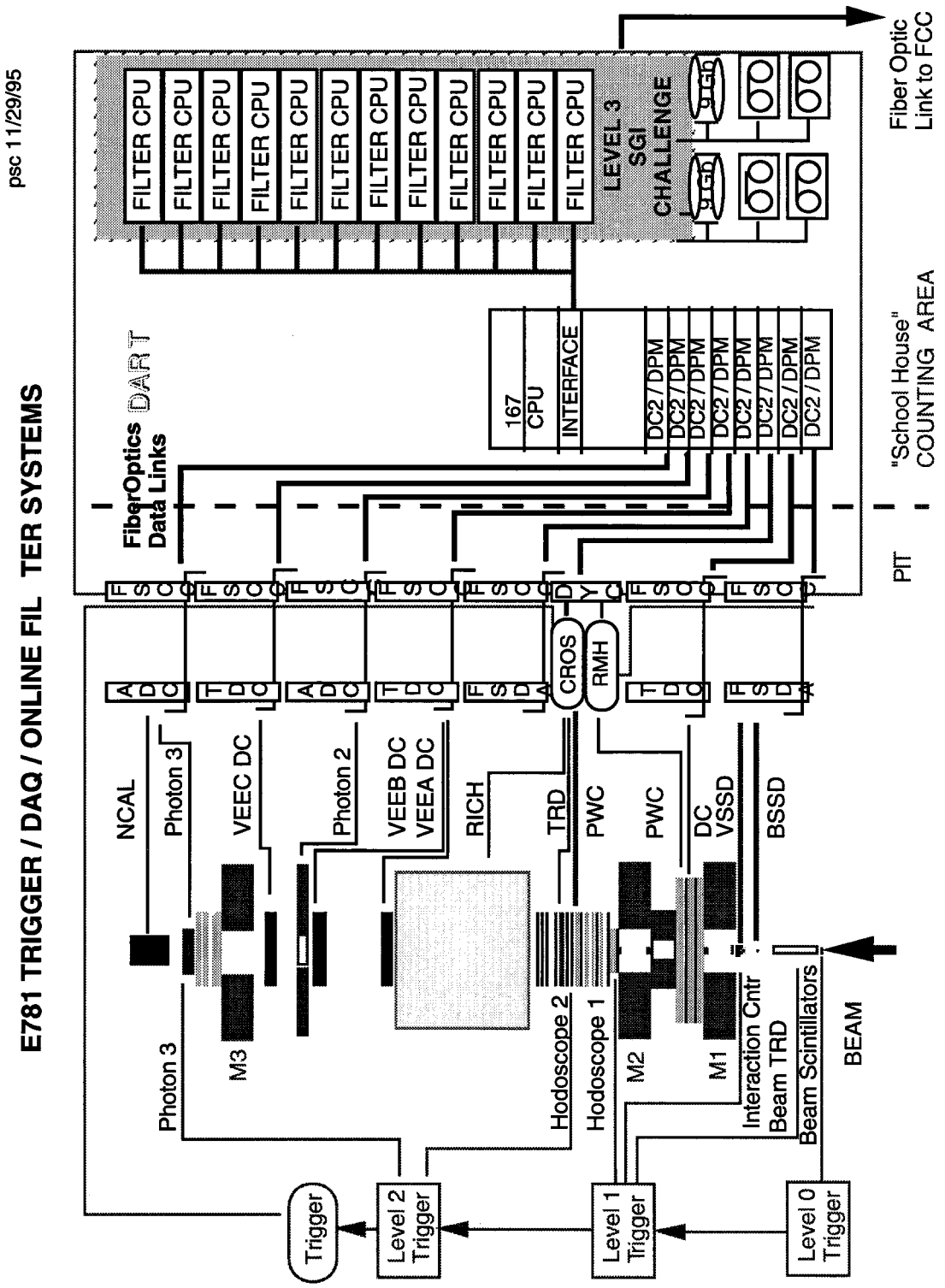


FIGURE 4. Selex (E781) DAQ system

Selex DAQ Parameters

Detector Channels

74K	Si Strips	ADC
17K	MWPC wires	Latches
5K	DC wires	TDC
2K	TRD wires	"Latches"
2K	PBG blocks	ADC
3K	RICH pmts	Latches
~100	Misc.	
~105K	Total channels	

DAQ Trigger Rate (T_2)	5000 Hz
Events Size	6500 bytes (average; heavy sparsification)
DAQ data rate	33 Mb/sec
On-line Software Filter	

Computers	20 SGI R4400 200 MHz, 144 MIP processors = 2880 MIPS
Algorithm	Reject events with high momentum (>15 GeV/c) tracks consistent with a single vertex. Off-line tracking and vertexing code written in FORTRAN.
Beam duty factor:	1/3 - 20 sec of beam every 60 sec
CPU time	<12 msec/event average
Rejection	90% - 1 trigger in 10 is written to tape.
Taping rate	1 Mb/sec (2-8505 8mm tape drives in parallel).

Selex DAQ Architecture

The Selex DAQ has 10 parallel systems each reading out about 10% of the detector channels. The trigger strobes each of the digitizers shown in the center column of figure 4. The dead time is about 100 μsec per trigger and the livetime ratio is 50%. This is an example of parallelism on a large scale. Each data pipe is a fiber optic cable which carries one stream of event fragments from the digitizers in the radiation area to a dual ported memory upstairs in the counting area. data flow on these pipe are unidirectional and pipelined hardware inserts event headers and byte counts so that data lose or corruption can be detected. At the end of a 20 sec beam spill the ~650 Mb of data from that spill reside in those 10 memories. There are dedicated real-time processors in the data flow between the end of the optical data pipe and each memory which build a directory in each memory. This is a list of the addresses where each of the ~100K event fragments begins in that memory.

Data are transferred to the filter computers in blocks of 400 events using the event directories for each memory. The filter computers are actually two SMP (symmetric multi-processors) with 12 and 10 processors respectively. These two machines run 11 and 9 filter jobs which filter the events in parallel selecting about one in ten triggers based upon a reconstruction algorithm which looks for events with high momentum tracks which are inconsistent with a single vertex. Only some of the ten data streams are unpacked and used by this filter algorithm. The time available is about 12 CPU-msec/event. We typically require most of this time

on average. Occasionally we are not done computing before then next accelerator spill comes (in 60 sec). In this case we must inhibit triggers for the next accelerator spill. This is an example of dead time at a very high level in a DAQ system.

When a trigger passes the filter criteria all its event fragments are copied to an output buffer (event building) which is written to a file on disk. These disk files are copied by a separate job to 8mm tape and renamed to a different directory on disk. There is enough disk to hold about 4 hours of data. The least recently used file is deleted to make space for new data. This allows time to recovery from tape drive problems and to look at the recent data before it is deleted.

A typical run last two hours and has about 20 such 200 Mb files. Shortly after the first files is closed during a run a set of jobs are run against this file on the on-line computers to produce on-line monitoring histograms. These allow the physicists on shift to determine that all the detectors and filters are working properly. One of the data files is copied over the network to a tape robot in the Feynman Computing Center. This makes a sample of each run available for off-line analysis on the Fermilab central computers with having to remount and read the raw data tape.

This system has worked quite well. The amount of experiment downtime due to DAQ problems has been about normal for a system of this level of complexity. We lose, at most, a few hours a week to DAQ problems.

CONCLUSIONS

We spend most of our time as experimentalists doing everything but particle physics. The ultimate goal of a DAQ system, like all the other HEP technologies, is to enable the experiment to do physics. Students, particularly, tend to forget, in the middle of all the battles to make an experiment really work, that all these techniques are merely means to and end. If the physics experiment didn't work it doesn't matter whether the DAQ did or not.

In the case of Selex there is little doubt that the experiment works. We have been taking data for about 5 months after 7 months commissioning a new apparatus. We have analyzed, off-line, about 10% of the data taken thus far and have already reconstructed the decays of about 1000 charmed particles. This is a long way from the $\sim 10^5$ charm decays we expect when a full analysis of the data is completed. This effort demonstrates that the experiment works and particularly that the relatively complicated filter algorithms succeed in keeping much of the desired signals. In this sense the Selex DAQ is a success.

REFERENCES

1. An example is the FASTBUS standard - ANSI/IEEE Std 960-1986.
2. More information on the DART collaboration can be found on the web at http://fndaub.fnal.gov:8000
3. More information on the SELEX experiment can be found on the web at http://fn781a.fnal.gov