THE PEP-14 DATA ACQUISITION SYSTEM

P. B. Madden, D. Bassett, S. J. Freedman, F. A. Harris,
C. P. Horne, M. J. Losty, J. Napolitano, V. Vuillemin

Lawrence Berkeley Laboratory, Berkeley, California 94720
Stanford University, Stanford, California 94305
and University of Hawaii, Honolulu, Hawaii 96822

ABSTRACT

We have developed a real time software system for the VAX 11/780 which employs ten processes cooperating concurrently in the task of data acquisition and online analysis for the "Free Quark Search" experiment at the new colliding beam storage ring facility (PEP) located at the Stanford Linear Accelerator Center. The system utilizes most of the resources available to the applications programmer under VAX 11/780 VMS including inter-process communication via mailbox, global event flag cluster, and mapped global section, and is written primarily in FORTRAN IV PLUS with time critical portions in MACRO-11. A discussion of the system architecture, implementation and performance comprises the body of this article.

INTRODUCTION

The Stanford Linear Accelerator Center (SLAC) is a national research facility operated for the U. S. Department of Energy by Stanford University for the purpose of performing experiments in high energy physics. Designed and constructed in a joint effort involving the Lawrence Berkeley Laboratory and SLAC, the "Positron Electron Project" (PEP) is nearing completion and several new experimental detectors are under development to fully exploit the research potential of this new machine. Specifically, "PEP-14" designates one of these detectors and the "Free Quark Search" it is designed to accomplish.

Developed by physicists from three institutions in a collaborative effort the software system has proven its effectiveness after several months of use, and represents one of the first full implementations at PEP of an online system designed specifically for the VAX 11/780. Although our discussion here is necessarily application dependent, the architecture described below is quite flexible and conforms readily to many sets of system requirements.

HARDWARE CONFIGURATION

Detector

The PEP-14 detector is especially designed to look for elementary particles produced in electron-positron collisions at PEP which have an electric charge less than that of the electron - one distinguishing feature of the free quark. The apparatus consists of two identical sections, arranged on opposite sides of the interaction region, each containing five multi-wire proportional chambers (MWPC's), nine layers of scintillation counters, and one layer of Cerenkov counters. It employs 456 photo-multiplier tubes to measure the energy deposited by the charged particles traversing the scintillators and roughly 1800 channels of anode readout in the ten MWPC's. The MWPC cathode readout is via delay line (four per chamber). Several layers of counters on each side of the interaction region are also used to measure time of flight information.

CAMAC

The CAMAC system required to support the detector configuration described above includes about 800 channels of analog to digital converters (ADC's) and 250 channels of time to digital converters (TDC's). Additionally, there is specialized electronics associated with the anode wire and delay line readout in the MWPC's. The system requires two CAMAC branches which interface via SLAC built branch drivers through the SLAC Trunk UNIBUS Driver (STUD) to the VAX. (1) The cycle time for the CAMAC interface is 4 microseconds and a full readout of data from the detector (about 1700 32 bit words) requires about 10 milliseconds.

VAX 11/780

The configuration of the VAX employed by PEP-14 for software development, data acquisition, hardware testing, and offline analysis is given below.

VAX 11/780 cpu
Floating point accelerator
One megabyte of physical memory
One RP06 disk drive
Two TU-77 magnetic tape drives
One VERSATEC line printer
Two GRINNELL visual display units

The PEP-14 VAX is one of six purchased by PEP to support the online computing needs of experimentalists. Since VAX 11/780 VMS provides a facility for concurrent programming in FORTRAN IV and a full set of software development tools, the VAX 11/780 is ideally suited to this task. Additionally, with its 32 bit word size and virtual memory operating system the VAX 11/780 is also capable of handling much of the offline scientific computing required by experimenters at SLAC.

DESIGN CONSIDERATIONS

The computing requirements for most high energy physics experiments fall into three categories: First, the online computer must handle the data acquisition phase of the experiment; Secondly, the online machine is typically needed to monitor the state of the experimental apparatus; Finally, computing support is usually required for a full offline analysis of data collected during the acquisition phase. In the past, "acquisition" and "monitoring" were usually handled by a dedicated minicomputer while offline analysis was accomplished on a larger mainframe. In this section we discuss the impact of each of the above mentioned categories on the PEP-14 software design.

Data Acquisition

Any experiment of an exploratory nature presents special problems for the online software. Since we do not know beforehand exactly what to expect in terms of event rate or relative frequency of event type we must design a system which can deliver optimum performance over a wide range of system input parameters. We will show in the next section that this requirement leads naturally to a multiprocessing architecture. In preparation for that discussion, an abbreviated list of the system requirements is given below.

We begin with those requirements which generate real time critical code.

R1 - The goal of the PEP-14 experiment is to make a quantitive determination of the probability of quark production at PEP. The accuracy of this determination depends on the total amount of data collected by the experiment. Since the most valuable resource involved in an experiment at PEP is the storage ring itself, the online software must be designed to collect as much data as possible while PEP is running. Consequently, the highest priority item involves collecting the raw data from the CAMAC system and moving it to the output tape.

R2 - Since the experiment requires the collection of a large amount of data it becomes important to sample the data online, convert some of it to physical quantities and perform a preliminary analysis of a subset of the events to ensure that the apparatus is functioning properly.

R3 - We know that most of the events collected during the experiment will not be crucial to our results (except for the accuracy determination mentioned above). Consequently, as many events as possible should be evaluated online in order to see which ones are potential quark candidates.

R4 - A portion of the converted data should be collected and put into histograms and scatterplots (tallied) so detector performance can be studied and preliminary physics results can be obtained over the duration of a data acquisition run.

R5 - If there is cpu time remaining, some fraction of the events (particularly quark candidates) should be subjected to a complete online analysis.

Monitoring

As can be seen from the discussion in the previous section the PEP-14 detector and associated electronics is sufficiently complex to require a set of computer assisted monitoring procedures. Although seemingly unrelated to the data acquisition software, these procedures do require access to the CAMAC system. In addition, a subset of these procedures must be executed just prior to every run of the acquisition software. It seems reasonable therefore to incorporate a consideration of these monitoring procedures into the design of the online system. This requirement will be referred to in later discussions as "M1".

Offline Analysis

As mentioned in the introduction to this section a provision must be made for a full analysis of the raw data tapes offline. Since the VAX 11/780 has the potential to handle this phase of the computing effort as well as the online work discussed above, it would be very advantageous if the online software were designed in such a way that it could be employed in offline analysis. This possibility was incorporated as a requirement into our design considerations for the PEP-14 system and will be referred to in later discussions as requirement "O1".

Overall System

The following is a list of requirements which are less real time critical, but equally important from the point view of a sound overall system design.

S1 - Apart from PEP another valuable resource in any experimental effort is the time of the physicists involved. Consequently, we require a good man/machine interface which will allow the experimenters to control and monitor the operation of the online system.

S2 - In keeping with the requirement just mentioned, we need a flexible graphics display facility which allows the experimenters to see at a glance the effect of a given event on the detector and to quickly evaluate any of the histograms and scatterplots accumulated by the system.

S3 - A centralized alarm message facility is required to ensure accurate reporting and logging of system problems throughout the experiment.

S4 - Since the requirements already considered imply a multiprocessing architecture, a procedure should be provided to automate as much as possible of the mechanics of process creation, initialization and direction.

S5 - Finally, a strictly applications dependent requirement is that the status of the PEP storage ring must be checked periodically and relevant data logged to the output tape.

SYSTEM ARCHITECTURE

A schematic representation of the system architecture and data flow is shown in Fig. 1. Each process in the system (representated by a circle) is labeled by a process name and the requirement it is

designed to satisfy, while each of the mapped global sections is represented by a rectangle and labeled with the section name. This figure represents the configuration of the online software when operating in the "acquisition" mode. There are two other modes of operation for the software system. One of these, "monitoring", is much simpler architecturally and is shown in Fig. 2. The final mode, "replay", is designed to handle the work of the offline analysis. As we will see in our discussion of this mode the architecture of the online software permits a rather elegant solution to the problem of offline analysis. A brief description of each process involved in the PEP-14 system is given below.

The PEP-14 Processes

EVENTSIN - The EVENTSIN process is designed to function as the application software interface to the CAMAC system. Controlled by parameters placed in a global section by other processes, it reads all or some subset of data from the CAMAC when an interrupt has occurred and places the data at predefined locations in the raw data buffer. This process must be very fast to allow for high event rates.

TAPEOUT - During data acquisition this process is responsible for moving the data from the raw data buffer and onto the output tape. Very little reformatting of data is involved here, but several event records must be blocked into a single physical record on tape.

PREANALYSIS - This process contains the minimal amount of analysis required to determine if an event is a potential quark candidate and must sample as many events as possible online. It gets data from the raw data buffer, converts it to physical quantities, does a preliminary reconstruction of the event, and moves its results to a preanalysis output buffer.

TALLY - The TALLY process is designed to look at selected locations in the various data buffers and collect the data into histograms and scatterplots. The locations whose contents are to be tallied are designated by the operator and can be changed while the system is running. Tally must also be fast in order to gather sufficient statistics over the course of a run.

ANALYSIS - The ANALYSIS process is responsible for the complete analysis of an event. It takes the data from a Preanalysis output buffer, makes the best possible evaluation of each event, and moves its results to an analysis output buffer. Particularly interesting events are written to the disk. Considering the size of its job this process is expected to look at only a few events during the data acquisition phase of the experiment.

ONE EVENT DISPLAY - This process is responsible for displaying information on single events for the operator. It interfaces with the GRINNELL visual display units and produces a drawing of the experimental apparatus with hits and tracks (based on data in the preanalysis or analysis buffers) superimposed. The operator can see immediately which sections of the counters and chambers were effected by the event. Additionally, an operator may request a dump from any of the three data buffers be directed to a user terminal or to the VERSATEC.

DISPLAY - This process will cause any of the histograms or scatterplots accumulated by the TALLY process to be displayed on the GRINNELL, a terminal or the VERSATEC. Any display can be called upon demand by the operator or he may request that they be shown automatically at some time interval.

COMMAND - The COMMAND process represents the user interface to the online software. Through this process the user can communicate with the other processes to control the operation of the online system. (ie, which varisbles are to be tallied, which displays are active, when to start and stop the run, etc.)

ALARM - This process provides a uniform alarm reporting scheme for the system. Any process which encounters an alarm condition simply writes a unique prioritized alarm number into the ALARM mailbox. This number is matched with the corresponding ASCII string in the ALARM process and the full alarm message is output to the console. In addition to the identifying alarm number, any process in the system may also pass associated parameter values to ALARM which will be output with the error message to aid the operator in diagnosing the alarm condition. Alarm messages may be suppressed or the severity of the alarm may be changed by the operator via the COMMAND process.

MASTER - Although the MASTER process is not directly involved in the event driven execution of the system it plays an important role during system initialization and at the beginning and end of each run. MASTER is responsible for the creation of each of the subprocesses, for notifying them when they are supposed to participate in a particular mode of operation, and for deleting them at the end of a run. It also keeps track of experimental run and tape numbers and monitors the performance of the system by computing statistics at the end of each run and receiving any termination mailbox messages from VAX 11/780 VMS.

PERIODIC MONITORING - This process is responsible for checking the status of the PEP storage ring at periodic intervals (via a network from the PEP control computer) and notifying TAPEOUT to log the results to the data tape.

As can be seen from the above descriptions there is essentially a one to one correspondence between processes and requirements. One exception to this statement is requirement R1 which is embodied in two processes, EVENTSIN and TAPEOUT. There are two reasons for this split: First, the interarrival time between interrupts for our application is not uniform but is more closely described by a Poisson distribution. By allowing the tape writing to operate concurrently with the CAMAC reading, and by placing multiple buffers between the two processes (currently five) we can withstand a higher mean event rate without losing events; Secondly, the EVENTSIN process can be replaced by a process which reads data from magnetic tape. This not only permits testing of the online software with Monte Carlo data, but also allows the online software to be used for offline analysis (Replay mode).

The second exception occurs in the process PREANALYSIS which combines requirements R2 and R3. Both of these requirements are so crucial to the experiment that there is no reason to prefer one over the other. Since there also is considerable overlap between the data required for both requirements, a single process is clearly the best solution.

Mapped Global Sections

As shown in Fig. 1 there are six mapped global sections employed in the PEP-14 system. Two of these are used to pass control information to the various processes while the remaining four are used for data storage. Among these last four there is still another distinction to be made. Three of them are used to contain experimental data from events in various stages of analysis while the fourth contains histograms and scatterplots accumulated over the duration of the run. Finally, those global sections which contain events are partitioned into several "buffers" each of which contains the data from only one event. Throughout this discussion we will use the term "buffer" to refer to that portion of a global section which holds the data from a single event. A brief description of each section and its use is given below.

MASSYNC - This is the control global section for the event driven operation of the software system. It contains information that the real time critical processes need to synchronize their operation and is used by the MASTER process to communicate the mode of operation, run number and other information characterizing the state of the run.

COMGLOBAL - This global section is used by the COMMAND process to communicate control information from the operator to the various processes.

BUFRAW - This global section contains the five raw data buffers currently provided for data read from the CAMAC. This placement of multiple event "buffers" between processes is an important feature of the online architecture since it allows the higher priority processes to decouple themselves from the downstream processes while buffers are available.

BUFPNL - This global section contains the five buffers currently provided for the preliminary results of PREANALYSIS.

BUFANL - Two buffers are provided in the BUFANL global section for fully analyzed results of the ANALYSIS process.

HBOOK - This global section holds the results accumulated by the TALLY process. In order to handle the definition, accumulation and editing of histograms and scatterplots we have chosen a general package, "HBOOK", which is available from the Data Handling Division of the European Organization for Nuclear Research (CERN).

Mailbox Communication

The mailboxes employed by the online system are shown in Fig. 3. The mailbox allows a simple implementation of a FIFO queue for interprocess communications that are not real time critical. Consequently, it is employed by the COMMAND process to communicate information from the operator to TALLY and DISPLAY as well as by the ALARM subprocess to receive messages from every other subprocess in the online system. Additionally, the mailbox can perform specialized functions under VAX 11/780 VMS. It is used by COMMAND in conjunction with terminal I/O to trap unsolicited input from the operator. MASTER also specifies a mailbox at process creation time to receive messages from VAX 11/780 VMS about the status of a process which terminates abnormally.

We now turn to a discussion of each mode of operation seperately.

Data Acquisition

The operation of the online software in the data acquisition mode is best illustrated if we make the following assumptions.

1. Assume a uniform interarrival time for events and denote the time between interrupts by I.

2. Assume each process consumes the same amount of cpu time per event which we denote by P.

3. Finally, we ignore the effects of context switching and I/O requests.

With these assumptions the number of events analyzed by each process for four values of I over a period NI is shown in Fig. 4. As can be seen from the figure, when I=2P there is only time to write the raw data to output tape. As I becomes larger each downstream process comes closer to analyzing all events until finally for values of I>5P every process has time to analyze every event. This simple example is somewhat misleading since it also demonstrates that a multiprocessing architecture is not the best solution for a situation in which the event rate is uniform. If we relax assumption 1 and assume a Poisson distributed interarrival time with mean <I> we get a closer approximation to the true system performance. This case is shown in Fig. 5 for several values of <I> and is obtained by forming a weighted average of the graphs in Fig. 4 with the Poisson distribution as the weighting factor. It can be seen from Fig. 5 that even for values of <I> near 2P the downstream processes still have time to analyze a subset of events. Additionally, the system adjusts its response dynamically to the value of <I> as was mentioned in the previous section.

Monitoring

A detailed description of the monitoring procedures required for the experimental apparatus is not relevant to our discussion of the system architecture and as can be seen from Fig. 2 they are all incorporated into a single process. The central point here is that the MONITORING process is able to use the EVENTSIN process to obtain data (by placing control information into a global section shared between the two processes) thereby saving considerable duplication of code required to read the CAMAC. Additionally, the MASTER process automatically invokes the MONITORING system at the request of the operator and places those processes

not involved in the monitoring procedures in a hibernating state.

Replay

As mentioned in the introduction to this section, the Replay mode is designed to accomplish a full analysis of the experimental results offline. The input to this mode of operation is a raw data tape produced by the "acquisition" mode. Architecturally, the replay mode is almost identical to the system shown in Fig. 1. The only differences are that the EVENTSIN process is replaced by TAPEIN and that PERIODIC MONITORING is no longer required.

Though the architecture is similar this mode differs significantly from data acquisition in its operation. While the acquisition mode allows for concurrent operation the replay is completely synchronous. No new event is placed in the raw data buffer until all the downstream processes have fully completed their tasks.

IMPLEMENTATION

Until now we have been using the term "process" to denote both subprocesses and detached processes. For the implementation of the architecture discussed above, we chose to use subprocesses rather than detached processes. This choice is not critical to the following discussion. The use of subprocesses allows for some simplification in the allocation of various devices to the online system and has the advantage during the debugging phase that one can usually recover from a hung system by simply logging out and starting over.

Development Scheme

The portion of a multiprocessing architecture most prone to error is that which has to do with interprocess synchronization. Fortunately, for most processes in the system there is a clear line of demarcation between that portion of the code required to handle interprocess communication and the portion which is strictly applications related. Consequently, we chose a "top down" implementation scheme for the online software. A control "shell" was defined for each subprocess which contained the code required for interprocess protocol, initialization and finalization. It provides the mechanism for each process to transfer between modes and hooks for the strictly application related routines.

The first step in the implementation process was to code a version of the MASTER process and to set up a shell for each of the subprocesses. This skeletal system was then tested to make sure that the overall design philosophy was sound. Once the operation of the skeletal system was verified the applications routines were included.

Interprocess Synchronization

In addition to the event flags required for event driven execution of the system, we are also using global event flags to handle the handshaking required during the subprocess creation, initialization and finalization. VAX 11/780 VMS provides 64 global event flags, divided into two clusters,for applications users. Although this number is more than adequate for the job at hand, some care must be taken in organization of the flags since it is not possible to wait for a logical "or" of event flags in more than one cluster.

Subprocess Creation - The MASTER process is responsible for creating all the subprocesses and insuring that they have successfully completed their initialization sequence (associating event flag clusters, mapping global sections, etc.) The MASTER process creates the subprocesses in a sequential manner with calls to the $CREPRC system service routine. Creation of the next subprocess in line is not initiated until MASTER receives notification (via global event flag) that the previous subprocess has completed its initialization, or until a "timeout" occurrs in MASTER.

Mode Initialization - The initialization described above which occurrs at subprocess creation time is referred to as "one-time" initialization since it must be done only when the online software is brought up in preparation for a run. In addition to the "one-time" initialization there is also an initialization procedure involved at the beginning of each mode of operation. Since the online software must be able to switch between data acquisition, monitoring, or replay mode fairly quickly we prefer not to go through the creation sequence for each different mode. Consequently, a subprocess cannot depend on the loader to initialize variables. Additionally, rewinds must be issued to tape drives, run numbers must be updated, the CAMAC must be cleared as well as many other miscellaneous items depending on the mode chosen. Again the MASTER process handles this initialization in a sequential manner, notifying each subprocess via a unique global event flag to begin a mode, passing the mode information in the control global section, and waiting for successfull completion.

Mode finalization - As in the mode initialization sequence described above each subprocesses generally has one or more things to perform at the end of each mode of operation. Mode finalization is also handled in a sequential manner with a unique event flag for each subprocess.

Event Driven Execution - Execution of the online system during data acquisition is controlled by global event flags and buffer status arrays contained in the MASSYNC global section and by the priorities assigned to each subprocess under VAX 11/780 VMS. A schematic representation of the subprocess execution sequence is shown in Fig. 6. In this figure an event flag which initiates subprocess execution is representated by an arrow from the subprocess which sets it to the subprocess which receives it. Upon having received the event flag (from $WAITFR or $WFLOR) the subprocess, in general, reads the event flag cluster to determine the sender. This information directs the subprocess to the global section containing the data to be processed.

It is at this point that the buffer status arrays come into play. Since the global sections are shared by more than one subprocess (up to four in the case of BUFRAW and BUFPNL) and since we require asynchronous operation, some "interlock" mechanism must be provided. This is accomplished by declar-

ing an array in MASSYNC for each event buffer in the system. The number of locations in the array is equal to the number of subprocesses which require access to the corresponding buffer and the value contained in each location signifies the state of a buffer relative to the subprocess. A buffer can be in one of the three states listed below.

1. Not Done - This indicates that the buffer contains data which requires the attention of the subprocess.

2. Busy - Indicates that a buffer is currently locked by the subprocess and should not be accessed by others.

3. Done - Indicates that the subprocess has completed its work on the data in the buffer and that the buffer is available for a new event.

By appropriate manipulation of these buffer status arrays each subprocess in the system directs the execution of the downstream subprocesses. This point is crucial to the operation of the system since the buffer status arrays together with multiple buffers give the real time critical subprocesses the capability to bypass completely the slower downstream subprocesses if the data rate should justify it!

There is one special case not covered in the general scheme just described. If PREANALYSIS should decide that an event is particularly interesting, the capability exists to flag any buffer as a "special event". Each process in the system can then direct its attention immediately to that event giving it precedence over all others.

Implementation Details

Some details of our particular implementation are given below.

User Interface - The COMMAND process employs a table driven "action verb" parsing scheme allowing both "key word" and positional parameters. It also provides an extensive set of help files and a facility for processing indirect command files. This interface has the advantage of being very flexible and of closely approximating the command language available under VAX 11/780 VMS. COMMAND also employs the "unsolicited input" feature of the terminal driver. In this scheme, COMMAND simply associates a single mailbox with one or more terminals (with the $ASSIGN system service) and posts a read request to the mailbox. If any of the terminals receive input for which there is no read pending a message identifying the terminal is written to the mailbox. COMMAND can then issue a read directly to the terminal to obtain the input string. This feature has proven quite useful since more than one terminal can be assigned to control the operation of the online system. Additionally, since the terminals are allocated to the MASTER process they are also available for I/O from any of the subprocesses in the system.

Parameter Files - We have made every effort in the implementation to keep the system as "table driven" as possible. Toward this end, we have made extensive use of ASCII files, accessible with the text editor, to control the initialization of the system. The MASTER process, for example, determines from a file at the beginning of each mode of operation which processes are involved, what their priorities should be, and which event flags to use in the initialization sequence. The effort was well justified since the ability to change the configuration of the system at will without a recompilation has proven to be invaluable not only in the testing phase but in the day to day operation of the system as well.

Global Sections - Except for considerations of speed and efficiency, there are only two things which prohibit an implementation entirely in FORTRAN IV. The first of these has to do with the fact that files created for use as mapped global sections must have the "User File Open" bit set in the "File Options" field of the File Access Block. This can be done by setting up the File Access Block in MACRO-11 with the $FAB macro and issuing $OPEN within the MACRO-11 routine or by providing a "USEROPEN" routine for use with the FORTRAN IV "OPEN" statement. Since the section files are also used as paging files for the global section pages the "Contiguous" bit should also be set in the File Access Block. Finally, by setting the "Create If" bit in the same field one need not worry about whether one is creating or opening files since the $CREATE macro will handle this automatically. (2) The second consideration has to do with the fact that global sections can only be mapped into the virtual address space on page boundaries. This presents a problem because the address space to be mapped to the section file appears in FORTRAN IV code as a COMMON block and the FORTRAN IV compiler currently aligns these blocks on a long word boundary. Since each COMMON block appears to the linker as a PSECT with the same name as the labelled common, one can force page alignment by including a dummy MACRO-11 routine in the link which contains a program section of the same name as the labelled common, but redefines the alignment attribute to "PAGE". The linker issues a warning diagnostic, but fortunately takes the correct course and starts the program section on a page boundary. (3)

Two other minor problems were encountered with the use of mapped global sections. Early in our development we had to increase the major system parameter GBLSECTIONS from its default value of 40 to a new value of 60. This increase was prompted by the "SECTBLFUL" error message from VAX 11/780 VMS when the $CRMPSC system was invoked. The second problem is due to the fact that when an image exits some pages of the section file may remain in physical memory. If another image tries to map to this file or if one tries to delete it the message "FILE LOCKED BY ANOTHER USER" is received. This persists until normal system activity forces the section pages back to the section file. The solution is to explicitly invoke the $UPDSEC system service just before the image exits which forces the section pages to be written back to the section file and allows the file to be closed normally.

Mailboxes - So far we have encountered only one relatively subtle problem with mailboxes related to their use with "unsolicited input". The problem is connected with the distinction between a "process" and an executable image under VAX VMS 11/780 and

manifests itself in the following way. Consider the case of an image which creates a temporary mailbox, associates it to a terminal and then exits normally. If one again executes that image within the context of the same process (ie. without logging out) the message SS$_SUPERSEDE is received from the $CREMBX system service routine. This is because the previously created mailbox was not deleted when the image exited. When the image then associates the newly created mailbox with the terminal it will not receive any messages since they are all apparently sent to the previously created mailbox. One solution is to associate a logical name with the mailbox and have the image check to see if the mailbox already exists before calling $CREMBX.

Quotas and Privileges - In order to accomodate an online system involving many subprocesses we created a special account with the AUTHORIZE program to provide the context in which to execute the MASTER image. The minimal set of privileges required for this account to support our configuration is given below.

1. GRPNAM - This is required because the subprocesses are using logical names placed in the group logical name table to refer to I/O devices.

2. ALTPRI - This privilege is needed because we require real time priorities for some subprocesses. The MASTER process sets these priorities at creation time and during mode initialization.

3. TMPMBX - The MASTER process requires this privilege to create the mailboxes used for interprocess communication.

We also provided the account with the maximum allowed value for each quota. Very early in the development process it became obvious that the default quota list supplied at process creation time was not adequate for our purposes. Consequently, the quota list is set up explicitly for each subprocess in MASTER based on parameter file input. Since each subprocess in our system has a rather specialized function, this has proven quite useful. The penalty for a poorly constructed quota list can be a few hours of puzzlement. Either the subprocess will not be created at all or it will wind up in a "Mutex Wait" state. The latter shows up as an MWAIT state with either the DCL SHOW SYSTEM command or with the USER feature of the DISPLAY utility. In principle, the problems with MWAIT can be solved with the $SETRWM system service, although we have so far not had to resort to this remedy. The problems encountered at subprocess creation time are typically more subtle and require the specification of a "termination mailbox" in the $CREPRC system service routine. If the subprocess is not successfully created or if it terminates abnormally during execution VAX 11/780 VMS will write a termination message into the mailbox which can be read by the MASTER process to diagnose the problem.

MACRO-11 Implementation - As mentioned in the abstract the online system is implemented primarily in FORTRAN IV PLUS. The two exceptions to this rule occur in PREANALYSIS and TALLY. Both of these subprocesses are in the real time loop and consequently, need to be as efficient as possible. Additionally, the specialized nature of their tasks lends itself to a much more efficient implementation in MACRO-11. Specifically, several levels of indirect addressing are required for TALLY while PREANALYSIS requires management of linked list data structures for the tracking algorithm. Although TAPEOUT is implemented completely in FORTRAN IV PLUS it does make use of a MACRO-11 routine to move data between buffers. This routine is essentially just the MOVC3 instruction and gains about a factor of two over the standard FORTRAN do loop. As will be seen in the next section this use of MACRO-11 has been well justified. It should also be mentioned here that the enhanced instruction set of the VAX 11/780 brings MACRO-11 much closer to higher level languages in ease of implementation.

SYSTEM PERFORMANCE

Performance Measurement

In order to measure the performance of the online system the following test was performed.

1. A version of EVENTSIN was set up to read data from magnetic tape rather than from the CAMAC system.

2. The $SETIMR system service routine was employed to provide a uniform event rate.

3. Finally the $GETJPI system service routine was used to get the total cpu time consumed during the simulated run.

The "run" described above was performed for four uniform event interarrival times, I, with 1000 events in each run. After subtracting the cpu time required for "initialization" the cpu time per event for each of the real time subprocesses in the system was computed. Since the performance of the TALLY subprocess depends on the number of histograms and scatterplots accumulated, a typical indirect command file was executed before each run which defined 21 histograms and 16 scatter plots. This cpu time per event (in milliseconds) as well as the number of events, N, handled by each subprocess is given in TABLE 1 for each value of I (also given in milliseconds).

TABLE 1

Subprocess	I=200 N	I=100 N	I=70 N	I=50 N	ms/ev
TAPEOUT	1000	1000	1000	1000	7.6
PREANALYSIS	1000	997	967	683	55.8
TALLY	1000	967	252	17	4.3
ANALYSIS	248	64	10	7	510.5

A few words of explanation are in order about the results in TABLE 1.

1. The interarrival times listed in the table do not include the time required to actually read the data from the tape into the raw data buffers.

2. All the subprocesses listed in the table were run at real time priorities. The highest (TAPEOUT) was set at 21 with priorities for the other subprocesses decreasing by one in the order they are shown in the table.

3. Since the run was made with a uniform event rate in principle the downstream processes should have gotten no events at all as the interarrival times became smaller. The small number of events processed by ANALYSIS and TALLY for the smaller values of I is due to the fact that there was a small amount of terminal I/O done while the tape was being read.

4. Since the ANALYSIS subprocess is not fully implemented at this point its applications portion was replaced with a simple loop designed to consume about ten times the cpu time of PREANALYSIS.

Although not specifically measured in this test, the cpu time per event for EVENTSIN when reading data from the CAMAC system is roughly 7 milliseconds.

VAX 11/780 VMS considerations

So far, the problems encountered with tuning VAX 11/780 VMS to run the online system have been relatively minor and are discussed below.

Subprocess working set size - At subprocess creation time the system checks the values in the quota list supplied to the $CREPRC system service routine against its own values set up by the SYSGEN utility. At our installation the value for the minor system parameter PQL_DWSQUOTA was originally set to 100 pages which imposed a limit on the size of the working set for each subprocess. One can solve this problem either by giving each subprocess the EXQUOTA privilege or by running SYSGEN to change the value of PQL_DWSQUOTA. We took the latter course and set PQL_DWSQUOTA to 256 pages which is more than adequate for our real time critical subprocesses.

Global Section Page Fault Cluster Size - The performance characteristics of the online system can also be quite sensitive to the page fault cluster size parameter specified to the $CRMPSC system service. Apparently a subprocess pays a double penalty for swapping when a significant portion of the working set consists of shared pages. Since the pages of a mapped global section are used by other processes they are not written to the swap file when a given process is swapped out. Consequently, if these pages are forced back to the section file while the process is swapped out it must generate page faults to reaccess them when it is again ready for execution. With our current full megabyte of physical memory swapping is not a significant factor in the operation of the online system. Most of our development work, however, was done with one half megabyte so the opportunity to observe the effects discussed above was readily available.

CONCLUSIONS

As can be seen from the results of the the preceding section the performance of the online system conforms closely to our original design specifications. Our data rate is limited only by the speed of the tape drive (currently about 145,000 bytes per second) and we expect this to be the case even when we go to faster 6250 BPI drives. Additionally, we should be able to preanalyze and tally roughly half the events online even with a mean event rate of 20 events per second. Our satisfaction with the performance of the online system is due, in large part, to an extensive design effort undertaken by members of the software development group which produced a complete design document before implementation began. (4) The importance of a well defined design phase in the development of any real time system has long been recognized and our experience again confirms this belief.

In retrospect, our experience with the development of a real time system on the VAX 11/780 has been a pleasant one. We have found VAX 11/780 VMS to be an extremely well documented and flexible system and we recommend it without hesitation for other applications with comparable computing requirements.

REFERENCES

1. REAL-TIME DATA COLLECTION USING MULTIPLE VAX/VMS SYSTEMS, Proceedings of the Digital Equipment Users Conference, Vol. 5, No. 4. C. Granieri, K. Johnson.

2. A FORTRAN callable MACRO-11 routine to handle the creation of files suitable for mapped global sections was originally supplied by the Digitel Equipment Corporation.

3. The solution to this problem was proposed by Art Leino of the SLAC Data Analysis Group.

4. SOFTWARE DESIGN FOR PEP-14, Lawrence Berkeley Laboratory, Group A Programming Note 259, October 1978, F. A. Harris, C. P. Horne, S. J. Freedman, M. J. Losty, V. Vuillemin.

ACKNOWLEDGEMENTS

The software architecture described in this article was adapted from a skeletal model for the PEP online systems originally suggested by the SLAC Data Analysis Group to whom we gratefully acknowledge our debt.

We would also like to thank the members of the Computer Science and Applied Mathematics Department at the Lawrence Berkeley Laboratory who graciously provided assistance on their PDP 11/70 based UNIX operating system to aid in the preparation of this article for publication.

This work was supported by the High Energy Physics Division, Office of Basic Energy Sciences, of the U. S. Department of Energy under Contract no. W-7405-ENG-48.

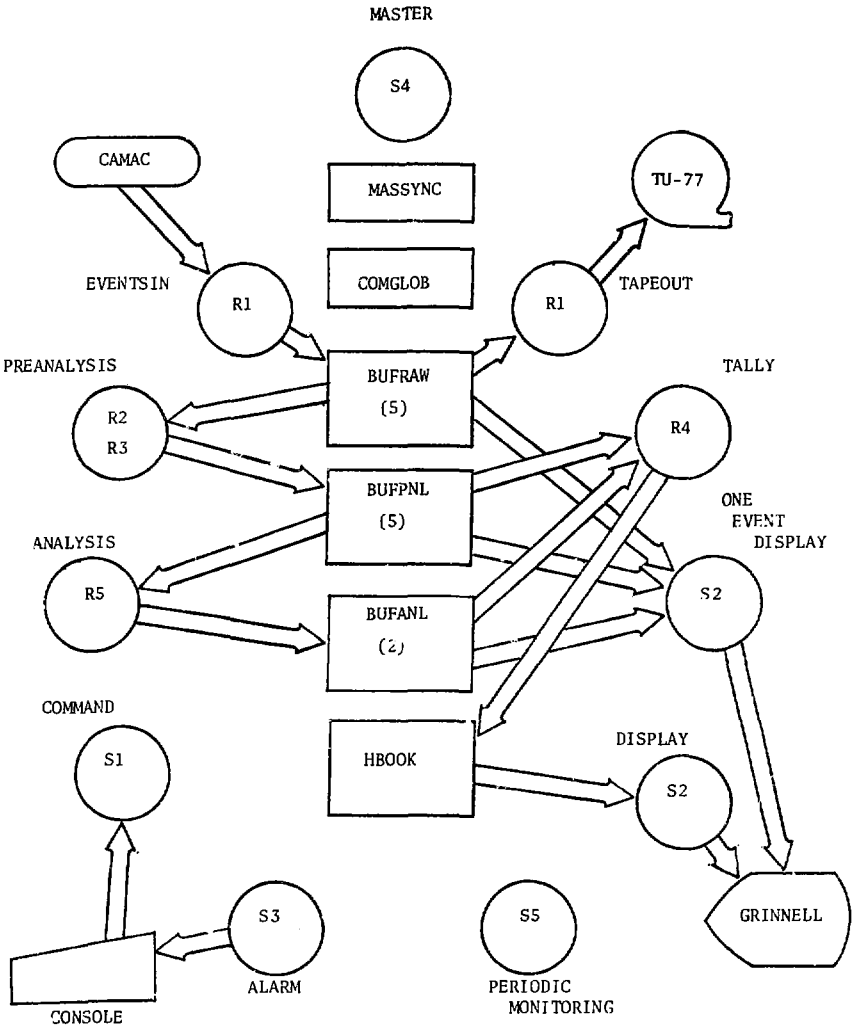


Figure 1. Architecture and Data Flow

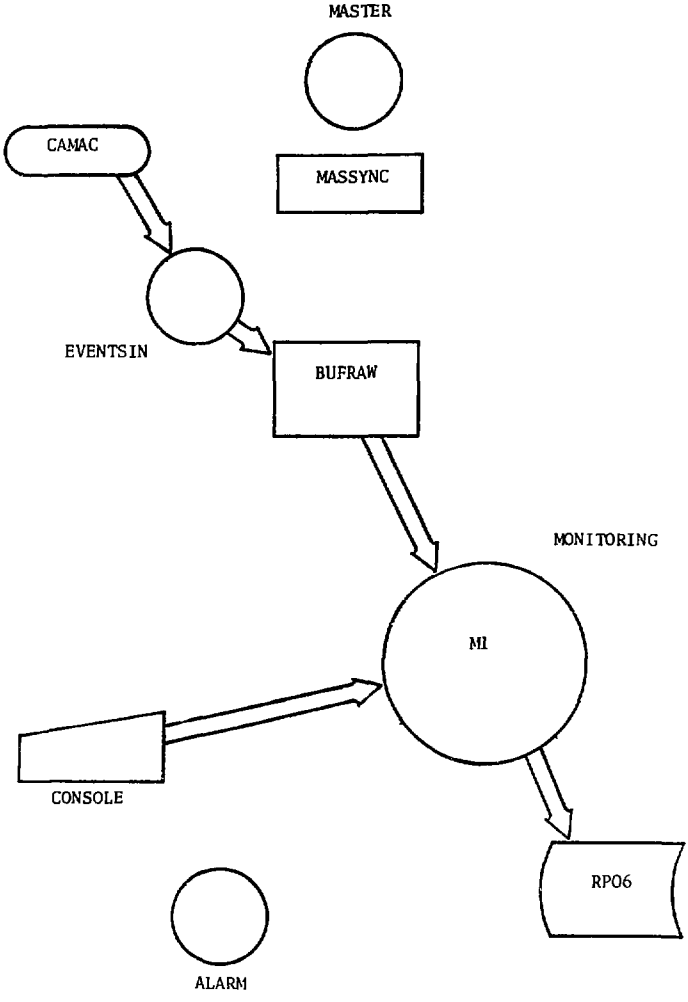


Figure 2. Monitoring Architecture

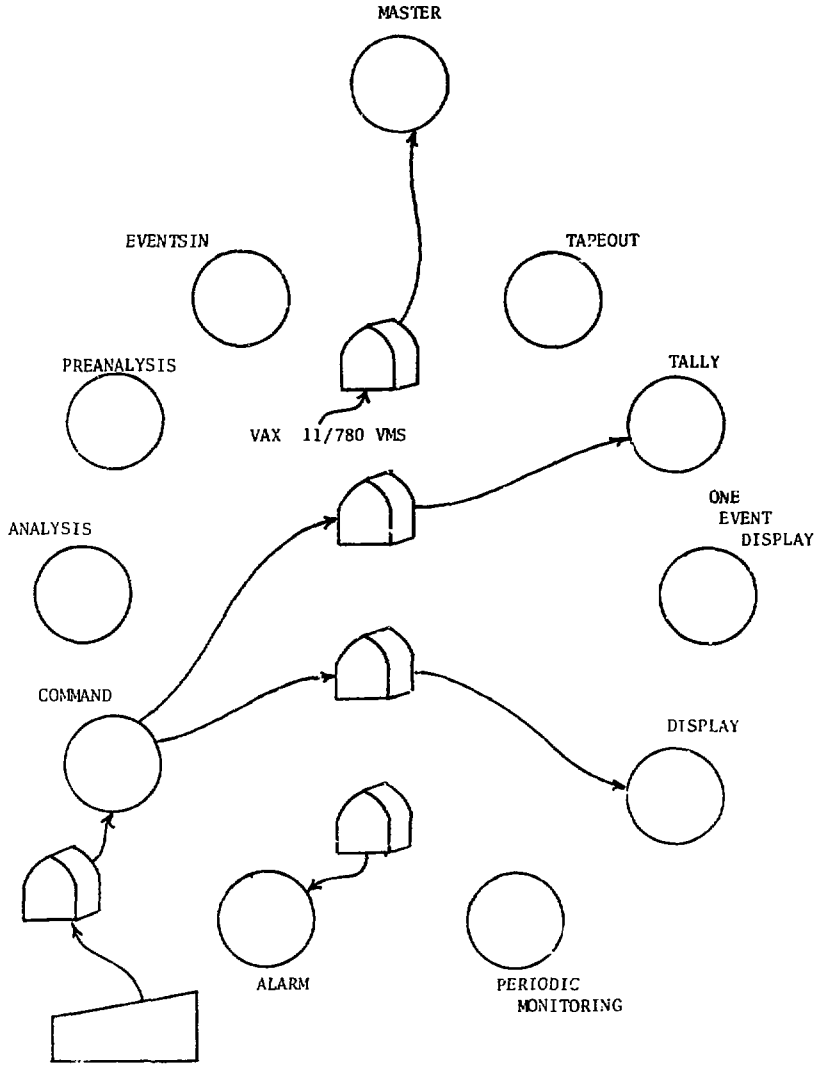
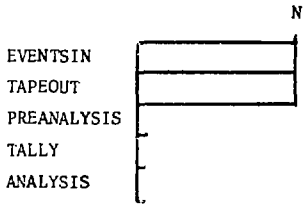
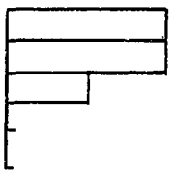


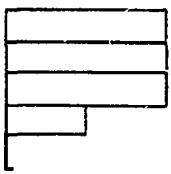
Figure 3. PEP-14 Mailboxes



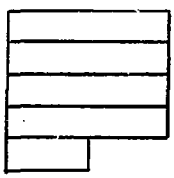
I = 2P



I = (5/2) P



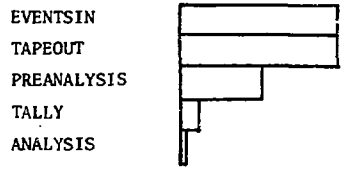
I = (7/2) P



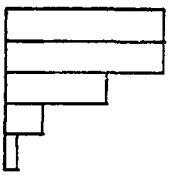
I = (9/2) P

Figure 4.

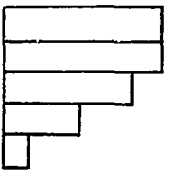
Uniformly Distributed
Interarrival Times



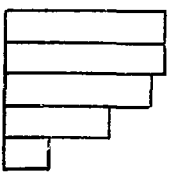
$\langle I \rangle = 2P$



$\langle I \rangle = (5/2)\ P$



$\langle I \rangle = (7/2)\ P$



$\langle I \rangle = (9/2)\ P$

Figure 5.

Poisson Distributed
Interarrival Times

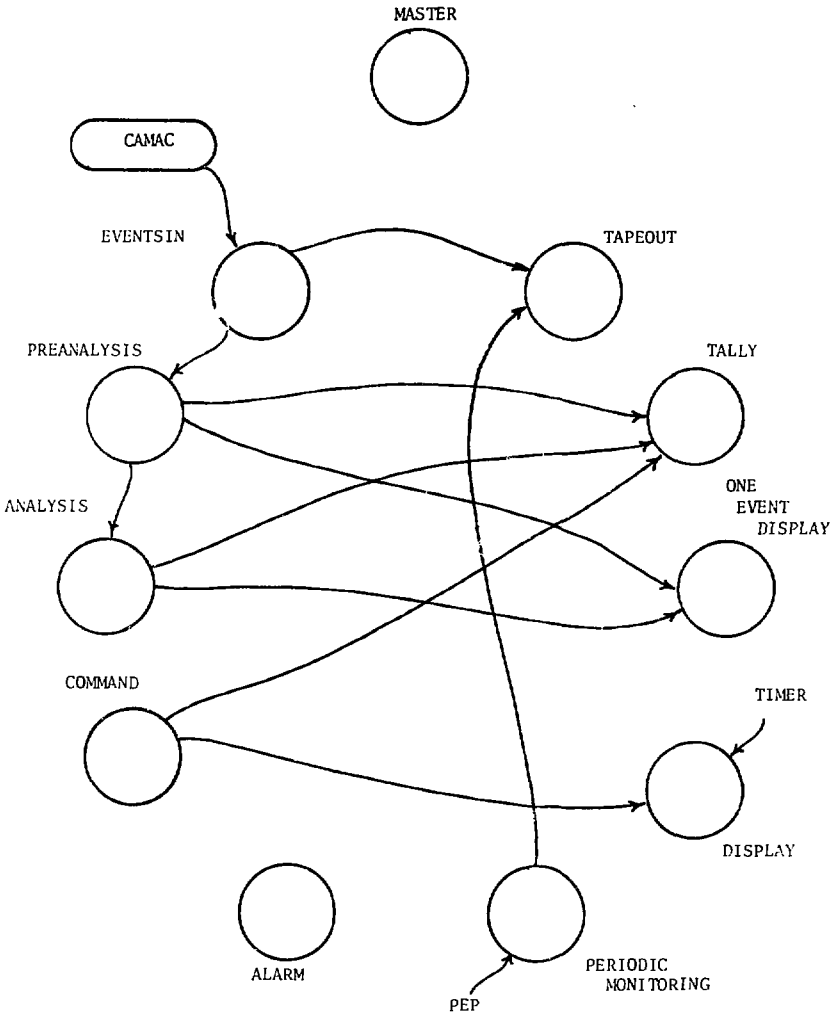


Figure 6. Event Driven Execution